SAND2014-5014
Unlimited Release
July 2014
Updated November 7, 2014

Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.1 Developers Manual

Brian M. Adams, Mohamed S. Ebeida, Michael S. Eldred, John D. Jakeman, Laura P. Swiler, J. Adam Stephens, Dena M. Vigil, Timothy M. Wildey
Optimization and Uncertainty Quantification Department

William J. Bohnhoff
Radiation Transport Department

Keith R. Dalbey
Mission Analysis and Simulation Department

John P. Eddy
System Readiness and Sustainment Technologies Department

Kenneth T. Hu
Validation and Uncertainty Quantification Department

Lara E. Bauman, Patricia D. Hough
Quantitative Modeling and Analysis Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185

Abstract

The Dakota (Design Analysis Kit for Optimization and Terascale Applications) toolkit provides a flexible and extensible interface between simulation codes and iterative analysis methods. Dakota contains algorithms for optimization with gradient and nongradient-based methods; uncertainty quantification with sampling, reliability, and stochastic expansion methods; parameter estimation with nonlinear least squares methods; and sensitivity/variance analysis with design of experiments and parameter study methods. These capabilities may be used on their own or as components within advanced strategies such as surrogate-based optimization, mixed integer nonlinear programming, or optimization under uncertainty. By employing object-oriented design to implement abstractions of the key components required for iterative systems analyses, the Dakota toolkit provides a flexible and extensible problem-solving environment for design and performance analysis of computational models on high performance computers.

This report describes the Dakota class hierarchies. It is derived from annotation of the source code and provides detailed class documentation, including all member functions and attributes.

Contents

1 Dakota Developers Manual **11**
- 1.1 Introduction . . . 11
- 1.2 Overview of Dakota . . . 11
- 1.3 Services . . . 16
- 1.4 Development Practices and Guidance . . . 16
- 1.5 Additional Resources . . . 17

2 Coding Style Guidelines and Conventions **19**
- 2.1 Introduction . . . 19
- 2.2 C++/c Style Guidelines . . . 19
- 2.3 File Naming Conventions . . . 21
- 2.4 Class Documentation Conventions . . . 22
- 2.5 CMake Style Guidelines . . . 22

3 Instructions for Modifying Dakota's Input Specification **25**
- 3.1 New XML specification . . . 25
- 3.2 Deprecated: Modify dakota.input.nspec . . . 27
- 3.3 Rebuild generated files . . . 28
- 3.4 Update NIDRProblemDescDB.cpp in Dakota/src . . . 28
- 3.5 Update ProblemDescDB.cpp in Dakota/src . . . 29
- 3.6 Update Corresponding Data Classes . . . 30
- 3.7 Use get_<data_type>() Functions . . . 31
- 3.8 Update the Documentation . . . 31

4 Understanding Iterator Flow **33**

5 Interfacing with Dakota as a Library **35**
- 5.1 Introduction . . . 35
- 5.2 Basic Dakota library instantiation . . . 36
- 5.3 Configuring Dakota operation . . . 37
- 5.4 Creating a simulator plugin interface . . . 40
- 5.5 Retrieving data after a run . . . 43
- 5.6 Linking against the Dakota library . . . 43

6 Performing Function Evaluations **45**
- 6.1 Synchronous function evaluations . . . 45
- 6.2 Asynchronous function evaluations . . . 45
- 6.3 Analyses within each function evaluation . . . 46

7 Working with Variable Containers and Views **47**
7.1 Storage in Variables . . . 47
7.2 Storage in SharedVariablesData . . . 48
7.3 Active and inactive views . . . 49

8 Namespace Index **51**
8.1 Namespace List . . . 51

9 Hierarchical Index **53**
9.1 Class Hierarchy . . . 53

10 Class Index **59**
10.1 Class List . . . 59

11 File Index **69**
11.1 File List . . . 69

12 Namespace Documentation **71**
12.1 Dakota Namespace Reference . . . 71
12.2 SIM Namespace Reference . . . 306

13 Class Documentation **307**
13.1 ActiveSet Class Reference . . . 307
13.2 Analyzer Class Reference . . . 309
13.3 ApplicationInterface Class Reference . . . 316
13.4 Approximation Class Reference . . . 330
13.5 ApproximationInterface Class Reference . . . 337
13.6 APPSEvalMgr Class Reference . . . 342
13.7 APPSOptimizer Class Reference . . . 344
13.8 BaseConstructor Struct Reference . . . 347
13.9 callback_data Struct Reference . . . 347
13.10 COLINApplication Class Reference . . . 348
13.11 COLINOptimizer Class Reference . . . 351
13.12 CollabHybridMetaIterator Class Reference . . . 355
13.13 CommandLineHandler Class Reference . . . 356
13.14 CommandShell Class Reference . . . 358
13.15 ConcurrentMetaIterator Class Reference . . . 359
13.16 CONMINOptimizer Class Reference . . . 362
13.17 ConsoleRedirector Class Reference . . . 369
13.18 Constraints Class Reference . . . 370
13.19 DataEnvironment Class Reference . . . 379
13.20 DataEnvironmentRep Class Reference . . . 380
13.21 DataFitSurrModel Class Reference . . . 382
13.22 DataInterface Class Reference . . . 393
13.23 DataMethod Class Reference . . . 394
13.24 DataMethodRep Class Reference . . . 395
13.25 DataModel Class Reference . . . 408
13.26 DataModelRep Class Reference . . . 409
13.27 DataResponses Class Reference . . . 413
13.28 DataResponsesRep Class Reference . . . 414

13.29DataVariables Class Reference . . . 418
13.30DataVariablesRep Class Reference . . . 419
13.31DDACEDesignCompExp Class Reference . . . 432
13.32DirectApplicInterface Class Reference . . . 436
13.33DiscrepancyCorrection Class Reference . . . 440
13.34DOTOptimizer Class Reference . . . 443
13.35JEGAOptimizer::Driver Class Reference . . . 447
13.36EffGlobalMinimizer Class Reference . . . 449
13.37EfficientSubspaceMethod Class Reference . . . 451
13.38EmbedHybridMetaIterator Class Reference . . . 455
13.39Environment Class Reference . . . 457
13.40NomadOptimizer::Evaluator Class Reference . . . 461
13.41JEGAOptimizer::Evaluator Class Reference . . . 463
13.42JEGAOptimizer::EvaluatorCreator Class Reference . . . 468
13.43ExecutableEnvironment Class Reference . . . 469
13.44ExperimentData Class Reference . . . 470
13.45ForkApplicInterface Class Reference . . . 471
13.46FSUDesignCompExp Class Reference . . . 473
13.47GaussProcApproximation Class Reference . . . 476
13.48GetLongOpt Class Reference . . . 481
13.49Graphics Class Reference . . . 484
13.50GridApplicInterface Class Reference . . . 486
13.51HierarchSurrModel Class Reference . . . 487
13.52Interface Class Reference . . . 491
13.53Iterator Class Reference . . . 500
13.54IteratorScheduler Class Reference . . . 514
13.55JEGAOptimizer Class Reference . . . 519
13.56LeastSq Class Reference . . . 525
13.57LibraryEnvironment Class Reference . . . 529
13.58LightWtBaseConstructor Struct Reference . . . 531
13.59MatchesWC Struct Reference . . . 531
13.60MatlabInterface Class Reference . . . 532
13.61MetaIterator Class Reference . . . 533
13.62Minimizer Class Reference . . . 535
13.63MixedVarConstraints Class Reference . . . 547
13.64MixedVariables Class Reference . . . 548
13.65Model Class Reference . . . 549
13.66MPIManager Class Reference . . . 580
13.67MPIPackBuffer Class Reference . . . 581
13.68MPIUnpackBuffer Class Reference . . . 583
13.69NCSUOptimizer Class Reference . . . 585
13.70NestedModel Class Reference . . . 588
13.71NIDRProblemDescDB Class Reference . . . 599
13.72NL2Res Struct Reference . . . 604
13.73NL2SOLLeastSq Class Reference . . . 604
13.74NLPQLPOptimizer Class Reference . . . 607
13.75NLSSOLLeastSq Class Reference . . . 611
13.76NoDBBaseConstructor Struct Reference . . . 613
13.77NomadOptimizer Class Reference . . . 613

13.78 NonD Class Reference . . . 616
13.79 NonDAdaptImpSampling Class Reference . . . 626
13.80 NonDAdaptiveSampling Class Reference . . . 629
13.81 NonDBayesCalibration Class Reference . . . 634
13.82 NonDCalibration Class Reference . . . 636
13.83 NonDCubature Class Reference . . . 638
13.84 NonDDREAMBayesCalibration Class Reference . . . 641
13.85 NonDExpansion Class Reference . . . 644
13.86 NonDGlobalEvidence Class Reference . . . 651
13.87 NonDGlobalInterval Class Reference . . . 652
13.88 NonDGlobalReliability Class Reference . . . 655
13.89 NonDGlobalSingleInterval Class Reference . . . 658
13.90 NonDGPImpSampling Class Reference . . . 659
13.91 NonDGPMSABayesCalibration Class Reference . . . 662
13.92 NonDIncremLHSSampling Class Reference . . . 665
13.93 NonDIntegration Class Reference . . . 667
13.94 NonDInterval Class Reference . . . 670
13.95 NonDLHSEvidence Class Reference . . . 672
13.96 NonDLHSInterval Class Reference . . . 673
13.97 NonDLHSSampling Class Reference . . . 674
13.98 NonDLHSSingleInterval Class Reference . . . 676
13.99 NonDLocalEvidence Class Reference . . . 678
13.100 NonDLocalInterval Class Reference . . . 679
13.101 NonDLocalReliability Class Reference . . . 681
13.102 NonDLocalSingleInterval Class Reference . . . 691
13.103 NonDPOFDarts Class Reference . . . 692
13.104 NonDPolynomialChaos Class Reference . . . 695
13.105 NonDQuadrature Class Reference . . . 699
13.106 NonDQUESOBayesCalibration Class Reference . . . 703
13.107 NonDReliability Class Reference . . . 705
13.108 NonDSampling Class Reference . . . 707
13.109 NonDSparseGrid Class Reference . . . 714
13.110 NonDStochCollocation Class Reference . . . 717
13.111 NonlinearCGOptimizer Class Reference . . . 719
13.112 NPSOLOptimizer Class Reference . . . 722
13.113 OptDartsOptimizer Class Reference . . . 724
13.114 Optimizer Class Reference . . . 727
13.115 OutputManager Class Reference . . . 732
13.116 OutputWriter Class Reference . . . 735
13.117 ParallelConfiguration Class Reference . . . 736
13.118 ParallelDirectApplicInterface Class Reference . . . 738
13.119 ParallelLevel Class Reference . . . 739
13.120 ParallelLibrary Class Reference . . . 742
13.121 ParamResponsePair Class Reference . . . 751
13.122 ParamStudy Class Reference . . . 755
13.123 partial_prp_equality Struct Reference . . . 760
13.124 partial_prp_hash Struct Reference . . . 760
13.125 PecosApproximation Class Reference . . . 761
13.126 ProblemDescDB Class Reference . . . 765

13.127 ProcessApplicInterface Class Reference . . . 775
13.128 ProcessHandleApplicInterface Class Reference . . . 779
13.129 ProgramOptions Class Reference . . . 784
13.130 PStudyDACE Class Reference . . . 788
13.131 PSUADEDesignCompExp Class Reference . . . 790
13.132 PythonInterface Class Reference . . . 793
13.133 RecastModel Class Reference . . . 795
13.134 RelaxedVarConstraints Class Reference . . . 804
13.135 RelaxedVariables Class Reference . . . 805
13.136 Response Class Reference . . . 806
13.137 ResponseRep Class Reference . . . 810
13.138 RestartWriter Class Reference . . . 817
13.139 ResultsDBAny Class Reference . . . 818
13.140 ResultsEntry< StoredType > Class Template Reference . . . 820
13.141 ResultsID Class Reference . . . 821
13.142 ResultsManager Class Reference . . . 822
13.143 ResultsNames Class Reference . . . 824
13.144 RichExtrapVerification Class Reference . . . 825
13.145 ScilabInterface Class Reference . . . 827
13.146 SensAnalysisGlobal Class Reference . . . 828
13.147 SeqHybridMetaIterator Class Reference . . . 831
13.148 SerialDirectApplicInterface Class Reference . . . 834
13.149 SharedApproxData Class Reference . . . 835
13.150 SharedPecosApproxData Class Reference . . . 840
13.151 SharedSurfpackApproxData Class Reference . . . 843
13.152 SharedVariablesData Class Reference . . . 845
13.153 SharedVariablesDataRep Class Reference . . . 850
13.154 SingleModel Class Reference . . . 855
13.155 SNLLBase Class Reference . . . 857
13.156 SNLLLeastSq Class Reference . . . 859
13.157 SNLLOptimizer Class Reference . . . 863
13.158 SOLBase Class Reference . . . 869
13.159 SpawnApplicInterface Class Reference . . . 871
13.160 SurfpackApproximation Class Reference . . . 872
13.161 SurrBasedGlobalMinimizer Class Reference . . . 875
13.162 SurrBasedLocalMinimizer Class Reference . . . 876
13.163 SurrBasedMinimizer Class Reference . . . 883
13.164 SurrogateModel Class Reference . . . 888
13.165 SysCallApplicInterface Class Reference . . . 892
13.166 TANA3Approximation Class Reference . . . 895
13.167 TaylorApproximation Class Reference . . . 897
13.168 TestDriverInterface Class Reference . . . 898
13.169 TrackerHTTP Class Reference . . . 903
13.170 UsageTracker Class Reference . . . 905
13.171 Var_icheck Struct Reference . . . 906
13.172 Var_rcheck Struct Reference . . . 906
13.173 Variables Class Reference . . . 907
13.174 Verification Class Reference . . . 916
13.175 VLint Struct Reference . . . 918

13.176 VLreal Struct Reference . . . 918
13.177 VLstr Struct Reference . . . 919
13.178 WorkdirHelper Class Reference . . . 919

14 File Documentation **925**
14.1 dakota_dll_api.cpp File Reference . . . 925
14.2 dakota_dll_api.h File Reference . . . 926
14.3 dakota_tabular_io.hpp File Reference . . . 927
14.4 dll_tester.cpp File Reference . . . 929
14.5 JEGAOptimizer.cpp File Reference . . . 929
14.6 JEGAOptimizer.hpp File Reference . . . 930
14.7 library_mode.cpp File Reference . . . 930
14.8 library_split.cpp File Reference . . . 934
14.9 main.cpp File Reference . . . 934
14.10 restart_util.cpp File Reference . . . 935

Index **937**

Chapter 1

Dakota Developers Manual

Author

Brian M. Adams, Lara E. Bauman, William J. Bohnhoff, Keith R. Dalbey, John P. Eddy, Mohamed S. Ebeida, Michael S. Eldred, Patricia D. Hough, Kenneth T. Hu, John D. Jakeman, Laura P. Swiler, J. Adam Stephens, Dena M. Vigil, Timothy M. Wildey

1.1 Introduction

The Dakota (Design Analysis Kit for Optimization and Terascale Applications) toolkit provides a flexible, extensible interface between analysis codes and iteration methods. Dakota contains algorithms for optimization with gradient and nongradient-based methods, uncertainty quantification with sampling, reliability, stochastic expansion, and interval estimation methods, parameter estimation with nonlinear least squares methods, and sensitivity/variance analysis with design of experiments and parameter study capabilities. (Solution verification and Bayesian approaches are also in development.) These capabilities may be used on their own or as components within advanced algorithms such as surrogate-based optimization, mixed integer nonlinear programming, mixed aleatory-epistemic uncertainty quantification, or optimization under uncertainty. By employing object-oriented design to implement abstractions of the key components required for iterative systems analyses, the Dakota toolkit provides a flexible problem-solving environment for design and performance analysis of computational models on high performance computers.

The Developers Manual focuses on documentation of Dakota design principles and class structures; it derives principally from annotated source code. For information on input command syntax, refer to the `Reference Manual`, and for more details on Dakota features and capabilities, refer to the Users Manual.

1.2 Overview of Dakota

In Dakota, the *environment* manages execution modes and input/output streams and defines the top-level iterator. This top-level iterator may be either a standard iterator or a meta-iterator. In the former case, the iterator identifies a model and the environment executes the iterator on the model to perform a single study. In the latter case, iterator recursions are present and sub-iterators may identify their own models. In both cases, models may contain additional recursions in the case of nested iteration or surrogate modeling. In a simple example, a hybrid meta-iterator might manage a global optimizer operating on a low-fidelity model that feeds promising design points into a local optimizer operating on a high-fidelity model. And in a more advanced example, a surrogate-based optimization under uncertainty approach would employ an uncertainty quantification iterator nested within an

optimization iterator and would employ truth models contained within surrogate models. Thus, iterators and models provide both stand-alone capabilities as well as building blocks for more sophisticated studies.

A model contains a set of *variables*, an *interface*, and a set of *responses*, and the iterator operates on the model to map the variables into responses using the interface. Each of these components is a flexible abstraction with a variety of specializations for supporting different types of iterative studies. In a Dakota input file, the user specifies these components through environment, method, model, variables, interface, and responses keyword specifications.

The use of class hierarchies provides a mechanism for extensibility in Dakota components. In each of the various class hierarchies, adding a new capability typically involves deriving a new class and providing a set of virtual function redefinitions. These redefinitions define the coding portions specific to the new derived class, with the common portions already defined at the base class. Thus, with a small amount of new code, the existing facilities can be extended, reused, and leveraged for new purposes. The following sections tour Dakota's class organization.

1.2.1 Environment

Class hierarchy: Environment.

Environments provide the top level abstraction for managing different execution modes and managing input and output streams. Specific environments include:

- ExecutableEnvironment: the environment for execution of Dakota as a stand-alone application.
- LibraryEnvironment: the environment for execution of Dakota as an embedded library service.

1.2.2 Iterators

Class hierarchy: Iterator. Iterator implementations may choose to split operations up into run-time phases as described in Understanding Iterator Flow.

The iterator hierarchy contains a variety of iterative algorithms for optimization, uncertainty quantification, nonlinear least squares, design of experiments, and parameter studies. The hierarchy is divided into MetaIterator, Minimizer, and Analyzer branches.

The MetaIterator classes manage sequencing and collaboration among multiple methods with support for concurrent iterator parallelism. Methods include:

- SeqHybridMetaIterator: hybrid minimization using a set of iterators employing a corresponding set of models of varying fidelity. The sequential hybrid passes the best solutions from one method in as the starting points of the next method in the sequence.
- CollabHybridMetaIterator: hybrid minimization employing collaboration and sharing of response data among methods during the course if iteration. This class is currently a placeholder.
- EmbedHybridMetaIterator: hybrid minimization involving periodic use of a local search method for refinement during the iteration of an outer global method. This class is currently a placeholder.
- ConcurrentMetaIterator: two similar algorithms are available: (1) multi-start iteration from several different starting points, and (2) pareto set optimization for several different multi-objective weightings. Employs a single iterator with a single model, but runs multiple instances of the iterator concurrently for different settings within the model.

The Minimizer classes address optimization and deterministic calibration and are grouped into:

- Optimization: Optimizer provides a base class for gradient-based (e.g., CONMINOptimizer and SNLLOptimizer) and derivative-free (e.g., NCSUOptimizer, JEGAOptimizer) optimization solvers. Most of these are wrappers for third-party libraries that implement the optimization algorithms. Classes APPSEvalMgr and COLINApplication provide the function evaluation interface for APPSOptimizer and COLINOptimizer, respectively.

- Parameter estimation: LeastSq provides a base class for NL2SOLLeastSq, a least-squares solver based on NL2SOL, SNLLLeastSq, a Gauss-Newton least-squares solver, and NLSSOLLeastSq, an SQP-based least-squares solver.

- Surrogate-based minimization (both optimization and nonlinear least squares): SurrBasedMinimizer provides a base class for SurrBasedLocalMinimizer, SurrBasedGlobalMinimizer, and EffGlobalMinimizer. The surrogate-based local and global methods employ a single iterator with any of the available SurrogateModel capabilities (local, multipoint, or global data fits or hierarchical approximations) and perform a sequence of approximate optimizations, each involving build, optimize, and verify steps. The efficient global method, on the other hand, hard-wires a recursion involving Gaussian process surrogate models coupled with the DIRECT global optimizer to maximize an expected improvement function.

The Analyzer classes are grouped into:

- Uncertainty quantification: NonD provides a base class for non-deterministic methods in several categories:

  - Sampling: NonDSampling is further specialized with the NonDLHSSampling class for Latin hypercube and Monte Carlo sampling, and a number of other classes supporting incremental and adaptive sampling such as NonDAdaptImpSampling for multi-modal adaptive importance sampling.
  - Reliability Analysis: NonDReliability is further specialized with local and global methods (NonDLocalReliability and NonDGlobalReliability). NonDPOFDarts implements a computational geometry-based reliability method.
  - Stochastic Expansions: NonDExpansion includes specializations for generalized polynomial chaos (NonDPolynomialChaos) and stochastic collocation (NonDStochCollocation) and is supported by the NonDIntegration helper class, which supplies cubature, tensor-product quadrature and Smolyak sparse grid methods (NonDCubature, NonDQuadrature, and NonDSparseGrid).
  - Bayesian Calibration: NonDCalibration provides a base class for nondeterministic calibration methods with specialization to Bayesian calibration in NonDBayesCalibration, and specific implementations such as NonDQUESOBayesCalibration.
  - NonDInterval provides a base class for epistemic interval-based UQ methods. Three interval analysis approaches are provided: LHS (NonDLHSInterval), efficient global optimization (NonDGlobalInterval), and local optimization (NonDLocalInterval). Each of these three has specializations for single interval and Dempster-Shafer Theory of Evidence approaches.
  - Experimental: EfficientSubspaceMethod implements a prototype input-space dimension reduction method for UQ.

- Parameter studies and design of experiments: PStudyDACE provides a base class for ParamStudy, which provides capabilities for directed parameter space interrogation, PSUADEDesignCompExp, which provides access to the Morris One-At-a-Time (MOAT) method for parameter screening, and DDACEDesignCompExp and FSUDesignCompExp, which provide for parameter space exploration through design and analysis of computer experiments. NonDLHSSampling from the uncertainty quantification branch also supports design of experiments when in `active all` variables mode.

- Solution verification studies: Verification provides a base class for RichExtrapVerification (verification via Richardson extrapolation) and other solution verification methods in development.

1.2.3 Models

Class hierarchy: Model.

The model classes are responsible for mapping variables into responses when an iterator makes a function evaluation request. There are several types of models, some supporting sub-iterators and sub-models for enabling layered and nested relationships. When sub-models are used, they may be of arbitrary type so that a variety of recursions are supported.

- SingleModel: variables are mapped into responses using a single Interface object. No sub-iterators or sub-models are used.
- SurrogateModel: variables are mapped into responses using an approximation. The approximation is built and/or corrected using data from a sub-model (the truth model) and the data may be obtained using a sub-iterator (a design of experiments iterator). SurrogateModel has two derived classes: DataFitSurrModel for data fit surrogates and HierarchSurrModel for hierarchical models of varying fidelity. The relationship of the sub-iterators and sub-models is considered to be "layered" since they are not used as part of every response evaluation on the top level model, but rather used periodically in surrogate update and verification steps.
- NestedModel: variables are mapped into responses using a combination of an optional Interface and a sub-iterator/sub-model pair. The relationship of the sub-iterators and sub-models is considered to be "nested" since they are used to perform a complete iterative study as part of every response evaluation on the top level model.
- RecastModel: recasts the inputs and outputs of a sub-model for the purposes of variable transformations (e.g., variable scaling, transformations to standardized random variables) and problem reformulation (e.g., multi-objective optimization, response scaling, augmented Lagrangian merit functions, expected improvement).

1.2.4 Variables

Class hierarchy: Variables.

The Variables class hierarchy manages design, aleatory uncertain, epistemic uncertain, and state *variable types* for continuous, discrete integer, and discrete real *domain types*. This hierarchy is specialized according to how the domain types are managed:

- MixedVariables: domain type distinctions are retained, such that separate continuous, discrete integer, and discrete real domain types are managed. This is the default Variable perspective, and draws its name from "mixed continuous-discrete" optimization.
- RelaxedVariables: domain types are combined through relaxation of discrete constraints; i.e., continuous and discrete variables are merged into continuous arrays through relaxation of integrality (for discrete integer ranges) or set membership (for discrete integer or discrete real sets) requirements. The branch and bound minimizer is the only method using this approach at present.

Whereas domain types are defined based on the derived Variables class selection, the selection of active variable types is handled within each of these derived classes using variable views. These permit different algorithms to work on different subsets of variables. Data shared among Variables instances is stored in SharedVariablesData. For details on managing variables, see Working with Variable Containers and Views.

The Constraints hierarchy manages bound, linear, and nonlinear constraints and utilizes the same specializations for managing bounds on the variables (see MixedVarConstraints and RelaxedVarConstraints).

1.2.5 Interfaces

Class hierarchy: Interface.

Interfaces provide access to simulation codes or, conversely, approximations based on simulation code data. In the simulation case, an ApplicationInterface is used. ApplicationInterface is specialized according to the simulation invocation mechanism, for which the following nonintrusive approaches are supported:

- SysCallApplicInterface: the simulation is invoked using a system call (the C function `system()`). Asynchronous invocation utilizes a background system call. Utilizes the CommandShell utility.
- ForkApplicInterface: the simulation is invoked using a fork (the `fork/exec/wait` family of functions). Asynchronous invocation utilizes a nonblocking fork.
- SpawnApplicInterface: for Windows, fork is replaced by spawn. Asynchronous invocation utilizes a nonblocking spawn.

Fork and Spawn are inherited from ProcessHandleApplicInterface and System and ProcessHandle are inherited from ProcessApplicInterface. A semi-intrusive approach is also supported by:

- DirectApplicInterface: the simulation is linked into the Dakota executable and is invoked using a procedure call. Asynchronous invocations will utilize nonblocking threads (capability not yet available). Specializations of the direct interface are implemented in MatlabInterface, PythonInterface, ScilabInterface, and (for built-in testers) TestDriverInterface, while examples of plugin interfaces for library mode in serial and parallel, respectively, are included in SerialDirectApplicInterface and ParallelDirectApplicInterface

Scheduling of jobs for asynchronous local, message passing, and hybrid parallelism approaches is performed in the ApplicationInterface class, with job initiation and job capture specifics implemented in the derived classes.

In the approximation case, global, multipoint, or local data fit approximations to simulation code response data can be built and used as surrogates for the actual, expensive simulation. The interface class providing this capability is

- ApproximationInterface: builds an approximation using data from a truth model and then employs the approximation for mapping variables to responses. This class contains an array of Approximation objects, one per response function, which support a variety of approximation types using the different Approximation derived classes. These include SurfpackApproximation (provides kriging, MARS, moving least squares, neural network, polynomial regression, and radial basis functions), GaussProcApproximation (Gaussian process models), PecosApproximation (multivariate orthogonal and Lagrange interpolation polynomials from Pecos), TANA3Approximation (two-point adaptive nonlinearity approximation), and TaylorApproximation (local Taylor series).

which is an essential component within the DataFitSurrModel capability described above in Models.

1.2.6 Responses

Class: Response.

The Response class provides an abstract data representation of response functions and their first and second derivatives (gradient vectors and Hessian matrices). These response functions can be interpreted as objective functions and constraints (optimization data set), residual functions and constraints (least squares data set), or generic response functions (uncertainty quantification data set). This class is not currently part of a class hierarchy, since the abstraction has been sufficiently general and has not required specialization.

1.3 Services

A variety of services and utilities are used in Dakota for parallel computing, failure capturing, restart, graphics, etc. An overview of the classes and member functions involved in performing these services is included here.

- Multilevel parallel computing: Dakota supports multiple levels of nested parallelism. A meta-iterator can manage concurrent iterators, each of which manages concurrent function evaluations, each of which manages concurrent analyses executing on multiple processors. Partitioning of these levels with MPI communicators is managed in ParallelLibrary and scheduling routines for the levels are part of IteratorScheduler, ApplicationInterface, and ForkApplicInterface.

- Option management: Global options controlling behavior are managed in ProgramOptions, with the help of command-line option parsing in CommandLineHandler.

- Parsing: Dakota employs NIDR (New Input Deck Reader) via Dakota::ProblemDescDB::parse_inputs to parse user input files. NIDR uses the keyword handlers in the NIDRProblemDescDB derived class to populate data within the ProblemDescDB base class, which maintains a DataEnvironment specification and lists of DataMethod, DataModel, DataVariables, DataInterface, and DataResponses specifications. Procedures for modifying the parsing subsystem are described in Instructions for Modifying Dakota's Input Specification.

- Failure capturing: Simulation failures can be trapped and managed using exception handling in ApplicationInterface and its derived classes.

- Restart: Dakota maintains a record of all function evaluations both in memory (for capturing any duplication) and on the file system (for restarting runs). Restart options are managed through ProgramOptions (with the help of CommandLineHandler); file management in OutputManager; and restart file insertions occur in ApplicationInterface. The `dakota_restart_util` executable, built from restart_util.cpp, provides a variety of services for interrogating, converting, repairing, concatenating, and post-processing restart files.

- Memory management: Dakota employs the techniques of reference counting and representation sharing through the use of letter-envelope and handle-body idioms (Coplien, "Advanced C++"). The former idiom provides for memory efficiency and enhanced polymorphism in the following class hierarchies: Environment, Iterator, Model, Variables, Constraints, Interface, ProblemDescDB, and Approximation. The latter idiom provides for memory efficiency in data-intensive classes which do not involve a class hierarchy. The Response and parser data (DataEnvironment, DataMethod, DataModel, DataVariables, DataInterface, and DataResponses) classes use this idiom. When managing reference-counted data containers (e.g., Variables or Response objects), it is important to properly manage shallow and deep copies, to allow for both efficiency and data independence as needed in a particular context.

- Graphics and Output: Dakota provides 2D iteration history graphics using Motif widgets. Graphics data can also be cataloged in a tabular data file for post-processing with 3rd party tools such as Matlab, Tecplot, etc. These capabilities are encapsulated within the Graphics class. An experimental results database is implemented in ResultsManager and ResultsDBAny. Options for controlling output and facilities for managing it are in OutputManager.

1.4 Development Practices and Guidance

The following links provide guidance for core software components or specific development activities:

- Coding Style Guidelines and Conventions - coding practices used by the Dakota development team.

- Instructions for Modifying Dakota's Input Specification - how to interact with NIDR and the associated Dakota classes.
- Interfacing with Dakota as a Library - embed Dakota as a service within your application.
- Understanding Iterator Flow - explanation of the full granularity of steps in Iterator execution.
- Performing Function Evaluations - an overview of the classes and member functions involved in performing function evaluations synchronously or asynchronously.
- Working with Variable Containers and Views - discussion of data storage for variables and explanation of active and inactive views of this data.

1.5 Additional Resources

Additional development resources include:

- The Dakota Developer Portal linked from `http://dakota.sandia.gov/developer/` includes information on getting started as a developer and links to project management resources.
- Project web pages are maintained at `http://dakota.sandia.gov/` including links to frequently asked questions, documentation, publications, mailing lists, and other resources.

Chapter 2

Coding Style Guidelines and Conventions

2.1 Introduction

Common code development practices can be extremely useful in multiple developer environments. Particular styles for code components lead to improved readability of the code and can provide important visual cues to other developers. Much of this recommended practices document is borrowed from the CUBIT mesh generation project, which in turn borrows its recommended practices from other projects, yielding some consistency across Sandia projects. While not strict requirements, these guidelines suggest a best-practices starting point for coding in Dakota.

2.2 C++/c Style Guidelines

Style guidelines involve the ability to discern at a glance the type and scope of a variable or function.

2.2.1 Class and variable styles

Class names should be composed of two or more descriptive words, with the first character of each word capitalized, e.g.:

```
class ClassName;
```

Class member variables should be composed of two or more descriptive words, with the first character of the second and succeeding words capitalized, e.g.:

```
double classMemberVariable;
```

Temporary (i.e. local) variables are lower case, with underscores separating words in a multiple word temporary variable, e.g.:

```
int temporary_variable;
```

Constants (i.e. parameters) and enumeration values are upper case, with underscores separating words, e.g.:

```
const double CONSTANT_VALUE;
```

2.2.2 Function styles

Function names are lower case, with underscores separating words, e.g.:

```
int function_name();
```

There is no need to distinguish between member and non-member functions by style, as this distinction is usually clear by context. This style convention allows member function names which set and return the value of a similarly-named private member variable, e.g.:

```
int memberVariable;
void member_variable(int a) { // set
  memberVariable = a;
}
int member_variable() const { // get
  return memberVariable;
}
```

In cases where the data to be set or returned is more than a few bytes, it is highly desirable to employ const references to avoid unnecessary copying, e.g.:

```
void continuous_variables(const RealVector& c_vars) { // set
  continuousVariables = c_vars;
}
const RealVector& continuous_variables() const {      // get
  return continuousVariables;
}
```

Note that it is not necessary to always accept the returned data as a const reference. If it is desired to be able change this data, then accepting the result as a new variable will generate a copy, e.g.:

```
// reference to continuousVariables cannot be changed
const RealVector& c_vars = model.continuous_variables();
// local copy of continuousVariables can be changed
RealVector c_vars = model.continuous_variables();
```

2.2.3 Miscellaneous

Appearance of typedefs to redefine or alias basic types is isolated to a few header files (`data_types.h`, `template_defs.h`), so that issues like program precision can be changed by changing a few lines of typedefs rather than many lines of code, e.g.:

```
typedef double Real;
```

`xemacs` is the preferred source code editor, as it has C++ modes for enhancing readability through color (turn on "Syntax highlighting"). Other helpful features include "Paren highlighting" for matching parentheses and the "New Frame" utility to have more than one window operating on the same set of files (note that this is still the same edit session, so all windows are synchronized with each other). Window width should be set to 80 internal columns, which can be accomplished by manual resizing, or preferably, using the following alias in your shell resource file (e.g., .cshrc):

```
alias xemacs "xemacs -g 81x63"
```

where an external width of 81 gives 80 columns internal to the window and the desired height of the window will vary depending on monitor size. This window width imposes a coding standard since you should avoid line wrapping by continuing anything over 80 columns onto the next line.

Indenting increments are 2 spaces per indent and comments are aligned with the code they describe, e.g.:

```
void abort_handler(int code)
{
  int initialized = 0;
  MPI_Initialized(&initialized);
  if (initialized) {
    // comment aligned to block it describes
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    if (size>1)
      MPI_Abort(MPI_COMM_WORLD, code);
    else
      exit(code);
  }
  else
    exit(code);
}
```

Also, the continuation of a long command is indented 2 spaces, e.g.:

```
const String& iterator_scheduling
  = problem_db.get_string("strategy.iterator_scheduling");
```

and similar lines are aligned for readability, e.g.:

```
cout << "Numerical gradients using " << finiteDiffStepSize*100. << "%"
     << finiteDiffType << " differences\nto be calculated by the "
     << methodSource << " finite difference routine." << endl;
```

Lastly, #ifdef's are not indented (to make use of syntax highlighting in xemacs).

2.3 File Naming Conventions

In addition to the style outlined above, the following file naming conventions have been established for the Dakota project.

File names for C++ classes should, in general, use the same name as the class defined by the file. Exceptions include:

- with the introduction of the Dakota namespace, base classes which previously utilized prepended Dakota identifiers can now safely omit the identifiers. However, since file names do not have namespace protection from name collisions, they retain the prepended Dakota identifier. For example, a class previously named DakotaModel which resided in DakotaModel.cpp/hpp, is now Dakota::Model (class Model in namespace Dakota) residing in the same filenames. The retention of the previous filenames reduces the possibility of multiple instances of a Model.hpp causing problems. Derived classes (e.g., NestedModel) do not require a prepended Dakota identifier for either the class or file names.

- in a few cases, it is convenient to maintain several closely related classes in a single file, in which case the file name may reflect the top level class or some generalization of the set of classes (e.g., DakotaResponse.[CH] files contain Dakota::Response and Dakota::ResponseRep classes, and DakotaBinStream.[CH] files contain the Dakota::BiStream and Dakota::BoStream classes).

The type of file is determined by one of the four file name extensions listed below:

- **.hpp** A class header file ends in the suffix .hpp. The header file provides the class declaration. This file does not contain code for implementing the methods, except for the case of inline functions. Inline functions are to be placed at the bottom of the file with the keyword inline preceding the function name.

- **.cpp** A class implementation file ends in the suffix .cpp. An implementation file contains the definitions of the members of the class.

- **.h** A header file ends in the suffix .h. The header file contains information usually associated with procedures. Defined constants, data structures and function prototypes are typical elements of this file.
- **.c** A procedure file ends in the suffix .c. The procedure file contains the actual procedures.

2.4 Class Documentation Conventions

Class documentation uses the doxygen tool available from http://www.doxygen.org and employs the JAVA-doc comment style. Brief comments appear in header files next to the attribute or function declaration. Detailed descriptions for functions should appear alongside their implementations (i.e., in the .cpp files for non-inlined, or in the headers next to the function definition for inlined). Detailed comments for a class or a class attribute must go in the header file as this is the only option.

NOTE: Previous class documentation utilities (class2frame and class2html) used the "//-" comment style and comment blocks such as this:

```
//- Class:       Model
//- Description: The model to be iterated by the Iterator.
//-              Contains Variables, Interface, and Response objects.
//- Owner:       Mike Eldred
//- Version: $Id: Dev_Recomm_Pract.dox 4549 2007-09-20 18:25:03Z mseldre $
```

These tools are no longer used, so remaining comment blocks of this type are informational only and will not appear in the documentation generated by doxygen.

2.5 CMake Style Guidelines

Dakota conventions for CMake files, such as CMakeLists.txt, FooConfig.cmake, etc., follow. Our goal is ease of reading, maintenance, and support, similar to the C++ code itself. Current CMake versions and build hints are maintained at the Developer Portal http://dakota.sandia.gov/developer/.

2.5.1 CMake Code Formatting

- Indentation is 2 spaces, consistent with Dakota C++ style.
- Lines should be kept to less than 80 chars per line where possible.
- Wrapped lines may be indented two spaces or aligned with prior lines.
- For ease of viewing and correctness checking in Emacs, a customization file is available: http://www.cmake.org/CMakeDocs/cmake-mode.el

2.5.2 CMake Variable Naming Conventions

These variable naming conventions are especially important for those that ultimately become preprocessor defines and affect compilation of source files.

- Classic/core elements of the CMake language are set in lower_case, e.g., option, set, if, find_library.
- Static arguments to CMake functions and macros are set in UPPER_CASE, e.g. REQUIRED, NO_MODULE, QUIET.
- Minimize "global" variables, i.e., don't use 2 variables with the same meaning when one will do the job.

- Feature toggling: when possible, use the "HAVE_<pkg/feature>" convention already in use by many CMake-enabled TPLs, e.g.,

```
$ grep HAVE_SYSTEM Dakota/src/CMakeLists.txt

check_function_exists(system HAVE_SYSTEM)
if(HAVE_SYSTEM)
  add_definitions("-DHAVE_SYSTEM")
endif(HAVE_SYSTEM)

$ grep HAVE_CONMIN Dakota/src/CMakeLists.txt Dakota/packages/CMakeLists.txt
Dakota/src/CMakeLists.txt:if(HAVE_CONMIN)
Dakota/src/CMakeLists.txt:endif(HAVE_CONMIN)
Dakota/packages/CMakeLists.txt:option(HAVE_CONMIN "Build the CONMIN package." ON)
Dakota/packages/CMakeLists.txt:if(HAVE_CONMIN)
Dakota/packages/CMakeLists.txt:endif(HAVE_CONMIN)
```

- When a variable/preprocessor macro could result in name clashes beyond Dakota scope, e.g., for library_-mode users, consider prefixing the "HAVE_<pkg>" name with DAKOTA_, e.g. DAKOTA_HAVE_MPI. Currently, MPI is the only use case for such a variable in Dakota, but many examples can be found in the CMake Modules source, e.g.

```
grep _HAVE_ <cmake_prefix_dir>/share/cmake-2.8/Modules/*
```

Chapter 3

Instructions for Modifying Dakota's Input Specification

To modify Dakota's input specification (for maintenance or addition of new input syntax), specification maintenance mode must be enabled at Dakota configure time with the `-DENABLE_SPEC_MAINT` option, e.g.,

```
./cmake -DENABLE_SPEC_MAINT:BOOL=ON ..
```

This will enable regeneration of NIDR and Dakota components which must be updated following a spec change.

3.1 New XML specification

The authoritative source for valid Dakota input grammar is now `dakota/src/dakota.xml`. The schema defining valid content for this XML file is in `dakota/src/dakota.xsd`. In the transition period (until parser work is complete), Dakota input grammar will be edited in the .xml file and translated using Java to `dakota/src/dakota.input.nspec` for consumption by the Dakota build process.

Process overview for updating the XML input definition:

- Make sure ENABLE_SPEC_MAINT is enabled in your build and necessary Java development tools are installed (see below).
- Edit the XML spec in `dakota.xml`.
- Perform a make in `dakota.build/src` which will regenerate `dakota.source/src/dakota.input.nspec`.
- Review that any changes induced in the `dakota.input.nspec` file are as expected.
- Proceed with verifying code changes and making downstream parse handler changes as normal.
- Commit the modified `dakota.xml` and `dakota.input.nspec` with other code changes.

3.1.1 XML Build Requirements

Editing the XML and then compiling Dakota requires

- Java Development Kit (JDK) for Java 6, providing the Java compiler javac. Version 1.6 or newer should work. Can satisfy on RHEL6 with RPM packages `java-1.6.0-openjdk-devel` and `java-1.6.0-openjdk`. This is needed to build the Java-based XML to NIDR translator. If this becomes too burdensome, we can check in the generated `xml2nidr.jar` file.

3.1.2 XML Editing Tools

The following tools will make editing dakota.input.xml easier.

- **Recommended: Eclipse Web Tools Platform.** Includes both graphical and text editors.
  1. Download Eclipse Standard (Classic)
  2. Configure proxy if needed, setting to manual: Window > Preferences > General > Network Connection > Proxy
  3. Install Web Tools Platform
     - Help > Install New Software
     - Work With: Kepler - http://download.eclipse.org/releases/kepler
     - Search "Eclipse X" and install two packages under Web, XML, Java
       * Eclipse XML Editors and Tools
       * Eclipse XSL Developer Tools
     - Optionally install C/C++ Development Tools
  4. Optional: add Subclipse for subversion (Subversive is the other major competing tool and I don't think requires JavaHL) Help > Install New Software Work With: http://subclipse.tigris.org/update_1.6.x Install Subclipse On Linux: yum install subversion-javahl.x86_64
  5. Alternately install Eclipse for Java or Eclipse Java EE development which includes webtools, then optionally add subclipse and C/C++ dev
- **Alternate: Emacs or your usual editor.** For example, Emacs supports an Nxml mode. You can tell it where to find the schema, edit XML, and have it perform validation against the schema. See help at http://www.gnu.org/software/emacs/manual/html_mono/nxml-mode.html
- **Other Suggested Alternates:** XMLSpy, DreamWeaver, XML Copy Editor

3.1.3 XML Features (with map to NIDR)

Out of necessity, the initial Dakota XML `dakota.xml` closely mirrors `dakota.input.nspec`. Valid Dakota input grammar is constrained by `dakota.xml`, an XML document which must validate against `dakota.xsd`. The top-level element of interest is <input>, which is comprised of a sequence of content elements (keywords, alternates, etc.), which may themselves contain additional child content elements. The key content types are:

- **Keyword (<keyword>):**, specified with the <keyword> element whose definition is given by keywordType in `dakota.xsd`. The required attributes are:
  - **name:** the keyword name as it will be given in user input; must follow same uniqueness rules are historical NIDR.
  - **code:** the verbatim NIDR handler to be invoked when parsed. In NIDR this was specified with {N_macro(...)}.

  Optional/useful parser-related elements/attributes in order of importance are:
  - **param sub-element:**Parameters and data types: A keyword may have an associated parameter element with a specified data type: `<param type="PARAMTYPE" />`. NIDR data types remain the same (INTEGER, REAL, STRING and LISTs thereof, but new data types INPUT_FILE and OUTPUT_FILE add convenience for the GUI, mapping to STRING for NIDR purposes. Parameters can also include attributes `constraint`, `in_taglist`, or `taglist`, which are used to help validate the user-specified parameter value. For example `constraint >= 0 LEN normal_uncertain`

  - **alias sub-element:** historical aliases for this keyword (can appear multiple times). Alias has a single attribute **name**
  - **id:** unique ID for the keyword, usually name with an integer appended
  - **minOccurs:** minimum occurrences of the keyword in current context (set to 1 for required, 0 for optional)
  - **maxOccurs:** maximum occurrences of the keyword in current context (for example environment may appear at most once)

  And optional/useful GUI-related attributes are:

  - **help:** a pointer to the corresponding reference manual section (deprecated as not needed with new reference manual format which mirrors keyword hierarchy)
  - **label:** a friendly label for the keyword to use in the GUI
  - **group:** category or group for this keyword, e.g., optimization vs. parameter study

- **Alternation (<oneOf>):** Alternation of groups of content is done with the element <oneOf> which indicates that its immediate children are alternates. In NIDR this was done with the pipe symbol: OptionA | OptionB
- **Required Group (<required>):** A required group can be specified by enclosing the contents in the <required> element. In NIDR this was done by enclosing the content in parentheses: (required group...)
- **Optional Group (<optional>):** An optional group can be specified by enclosing the contents in the <optional> element. In NIDR this was done by enclosing the content in brackets: [optional group...]

3.2 Deprecated: Modify dakota.input.nspec

The master input specification dakota.input.nspec in Dakota/src is the primary file to update when making a specification change. It uses the following syntactic elements:

- () for required group specifications
- [] for optional specifications
- | for alternatives
- {} for functions to process keywords to express logical relationships. These syntactic elements can be used to express various dependency relationships in the input specification. It is recommended that you review the existing specification and have an understanding of the constructs in use before attempting to add new ones.

Warning

- Do *not* skip this step. Attempts to modify the NIDR_keywds.hpp file in Dakota/src without using the NIDR table generator are very error-prone. Moreover, the input specification provides a reference to the allowable inputs of a particular executable and should be kept in synch with the parser files; modifying the parser files independent of the input specification creates, at a minimum, undocumented features.
- All keywords in dakota.input.nspec are lower case by convention. All user inputs are converted to lower case by the parser prior to keyword match testing, resulting in case insensitive parsing.

- Since the NIDR parser allows abbreviation of keywords, you *must* avoid adding a keyword that could be misinterpreted as an abbreviation for a different keyword within the same top-level keyword, such as "environment" and "method". For example, adding the keyword "expansion" within the method specification would be a mistake if the keyword "expansion_factor" already was being used in this specification.
- The NIDR input is somewhat order-dependent, allowing the same keyword to be reused multiple times in the specification. This often happens with aliases, such as `lower_bounds`, `upper_bounds` and `initial_point`. Ambiguities are resolved by attaching a keyword to the most recently seen context in which it could appear, if such exists, or to the first relevant context that subsequently comes along in the input file. With the earlier IDR parser, non-exclusive specifications (those not in mutually exclusive blocks) were required to be unique. That is why there are such aliases for `initial_point` as `cdv_initial_point` and `ddv_initial_point`: so older input files can be used with no or fewer changes.

3.3 Rebuild generated files

When configured with `-DENABLE_SPEC_MAINT`, performing a make in Dakota/src will regenerate all files which derive from dakota.input.nspec, including NIDR_keywds.hpp, dakota.input.summary, NIDR_guikywds.-h, and dakota.input.desc. If you commit changes to a source repository, be sure to commit any automatically generated files in addition to any modified in the following steps. It is not strictly necessary to run make at this point in the sequence, and in fact may generate errors if necessary handlers aren't yet available.

3.4 Update NIDRProblemDescDB.cpp in Dakota/src

Many keywords have data associated with them: an integer, a floating-point number, a string, or arrays of such entities. Data requirements are specified in dakota.input.nspec by the tokens INTEGER, REAL, STRING, INTEGERLIST, REALLIST, STRINGLIST. (Some keywords have no associated data and hence no such token.) After each keyword and data token, the dakota.input.nspec file specifies functions that the NIDR parser should call to record the appearance of the keyword and deal with any associated data. The general form of this specification is

{ startfcn, startdata, stopfcn, stopdata }

i.e., a brace-enclosed list of one to four functions and data pointers, with trailing entities taken to be zero if not present; zero for a function means no function will be called. The startfcn must deal with any associated data. Otherwise, the distinction between startfcn and stopfcn is relevant only to keywords that begin a group of keywords (enclosed in parentheses or square brackets). The startfcn is called before other entities in the group are processed, and the stop function is called after they are processed. Top-level keywords often have both startfcn and stopfcn; stopfcn is uncommon but possible for lower-level keywords. The startdata and (if needed) stopdata values are usually pointers to little structures that provide keyword-specific details to generic functions for startfcn and stopfcn. Some keywords that begin groups (such as "approx_problem" within the top-level "environment" keyword) have no need of either a startfcn or a stopfcn; this is indicated by "{0}".

Most of the things within braces in dakota.input.nspec are invocations of macros defined in NIDRProblemDescDB.cpp. The macros simplify writing dakota.input.nspec and make it more readable. Most macro invocations refer to little structures defined in NIDRProblemDescDB.cpp, usually with the help of other macros, some of which have different definitions in different parts of NIDRProblemDescDB.cpp. When adding a keyword to dakota.input.nspec, you may need to add a structure definition or even introduce a new data type. NIDRProblemDescDB.cpp has sections corresponding to each top-level keyword. The top-level keywords are in alphabetical order, and most entities in the section for a top-level keyword are also in alphabetical order. While not required, it is probably good practice to maintain this structure, as it makes things easier to find.

Any integer, real, or string data associated with a keyword are provided to the keyword's startfcn, whose second argument is a pointer to a `Values` structure, defined in header file `nidr.h`.

Example 1: if you added the specification:

```
[method_setting REAL {method_setting_start, &method_setting_details} ]
```

you would provide a function

```
    void NIDRProblemDescDB::
method_setting_start(const char *keyname, Values *val, void **g, void *v)
{ ... }
```

in NIDRProblemDescDB.cpp. In this example, argument `&method_setting_details` would be passed as `v`, `val->n` (the number of values) would be 1 and `*val->r` would be the REAL value given for the `method_setting` keyword. The `method_setting_start` function would suitably store this value with the help of `method_setting_details`.

For some top-level keywords, `g` (the third argument to the startfcn and stopfcn) provides access to a relevant context. For example, `method_start` (the startfcn for the top-level `method` keyword) executes

```
DataMethod *dm = new DataMethod;
 g = (void*)dm;
```

(and supplies a couple of default values to `dm`). The start functions for lower-level keywords within the `method` keyword get access to `dm` through their `g` arguments. Here is an example:

```
 void NIDRProblemDescDB::
method_str(const char *keyname, Values *val, void **g, void *v)
{
        (*(DataMethod**)g)->**(String DataMethod::**)v = *val->s;
        }
```

In this example, `v` points to a pointer-to-member, and an assignment is made to one of the components of the DataMethod object pointed to by `*g`. The corresponding stopfcn for the top-level `method` keyword is

```
 void NIDRProblemDescDB::
method_stop(const char *keyname, Values *val, void **g, void *v)
{
        DataMethod *p = *(DataMethod**)g;
        pDDBInstance->dataMethodList.insert(*p);
        delete p;
        }
```

which copies the now populated DataMethod object to the right place and cleans up.

Example 2: if you added the specification

```
[method_setting REALLIST {{N_mdm(RealL,methodCoeffs)}
```

then method_RealL (defined in NIDRProblemDescDB.cpp) would be called as the startfcn, and methodCoeffs would be the name of a (currently nonexistent) component of DataMethod. The N_mdm macro is defined in NIDRProblemDescDB.cpp; among other things, it turns `RealL` into `NIDRProblemDescDB::method_RealL`. This function is used to process lists of REAL values for several keywords. By looking at the source, you can see that the list values are val->r[i] for 0 <= `i` < val->n.

3.5 Update ProblemDescDB.cpp in Dakota/src

3.5.1 Augment/update get_<data_type>() functions

The next update step involves extending the database retrieval functions in ProblemDescDB.cpp. These retrieval functions accept an identifier string and return a database attribute of a particular type, e.g., a RealVector:

```
const RealVector& get_rv(const String& entry_name);
```

The implementation of each of these functions contains tables of possible entry_name values and associated pointer-to-member values. There is one table for each relevant top-level keyword, with the top-level keyword omitted from the names in the table. Since binary search is used to look for names in these tables, each table must be kept in alphabetical order of its entry names. For example,

```
...
else if ((L = Begins(entry_name, "model."))) {
  if (dbRep->methodDBLocked)
      Locked_db();

  #define P &DataModelRep::
  static KW<RealVector, DataModelRep> RVdmo[] = {     // must be sorted
      {"nested.primary_response_mapping", P primaryRespCoeffs},
      {"nested.secondary_response_mapping", P secondaryRespCoeffs},
      {"surrogate.kriging_conmin_seed", P krigingConminSeed},
      {"surrogate.kriging_correlations", P krigingCorrelations},
      {"surrogate.kriging_max_correlations", P krigingMaxCorrelations},
      {"surrogate.kriging_min_correlations", P krigingMinCorrelations}};
  #undef P

  KW<RealVector, DataModelRep> *kw;
  if ((kw = (KW<RealVector, DataModelRep>*)Binsearch(RVdmo, L)))
      return dbRep->dataModelIter->dataModelRep->*kw->p;
}
```

is the "model" portion of ProblemDescDB::get_rv(). Based on entry_name, it returns the relevant attribute from a DataModel object. Since there may be multiple model specifications, the `dataModelIter` list iterator identifies which node in the list of DataModel objects is used. In particular, `dataModelList` contains a list of all of the `data_model` objects, one for each time a top-level `model` keyword was seen by the parser. The particular model object used for the data retrieval is managed by `dataModelIter`, which is set in a `set_db_list_nodes()` operation that will not be described here.

There may be multiple DataMethod, DataModel, DataVariables, DataInterface, and/or DataResponses objects. However, only one specification is currently allowed so a list of DataEnvironment objects is not needed. Rather, ProblemDescDB::environmentSpec is the lone DataEnvironment object.

To augment the get_<data_type>() functions, add table entries with new identifier strings and pointer-to-member values that address the appropriate data attributes from the Data class object. The style for the identifier strings is a top-down hierarchical description, with specification levels separated by periods and words separated with underscores, e.g., `"keyword.group_specification.individual_specification"`. Use the `dbRep->listIter->attribute` syntax for variables, interface, responses, and method specifications. For example, the `method_setting` example attribute would be added to `get_drv()` as:

```
{"method_name.method_setting", P methodSetting},
```

inserted at the beginning of the `RVdmo` array shown above (since the name in the existing first entry, i.e., "nested.-primary_response_mapping", comes alphabetically after "method_name.method_setting").

3.6 Update Corresponding Data Classes

In this step, we extend the Data class definitions (DataEnvironment, DataMethod, DataModel, DataVariables, DataInterface, and/or DataResponses) to include the new attributes referenced in Update NIDRProblemDescDB.-cpp in Dakota/src and Augment/update get_<data_type>() functions.

3.6.1 Update the Data class header file

Add a new attribute to the public data for each of the new specifications. Follow the style guide for class attribute naming conventions (or mimic the existing code).

3.6.2 Update the .cpp file

Define defaults for the new attributes in the constructor initialization list. Add the new attributes to the assign() function for use by the copy constructor and assignment operator. Add the new attributes to the write(MPIPackBuffer&), read(MPIUnpackBuffer&), and write(ostream&) functions, paying careful attention to the use of a consistent ordering.

3.7 Use get_<data_type>() Functions

At this point, the new specifications have been mapped through all of the database classes. The only remaining step is to retrieve the new data within the constructors of the classes that need it. This is done by invoking the get_<data_type>() function on the ProblemDescDB object using the identifier string you selected in Augment/update get_<data_type>() functions. For example:

```
const String& interface_type = problem_db.get_string("interface.type");
```

passes the `"interface.type"` identifier string to the ProblemDescDB::get_string() retrieval function, which returns the desired attribute from the active DataInterface object.

Warning

> Use of the get_<data_type>() functions is restricted to class constructors, since only in class constructors are the data list iterators (i.e., `dataMethodIter`, `dataModelIter`, `dataVariablesIter`, `dataInterfaceIter`, and `dataResponsesIter`) guaranteed to be set correctly. Outside of the constructors, the database list nodes will correspond to the last set operation, and may not return data from the desired list node.

3.8 Update the Documentation

Doxygen comments should be added to the Data class headers for the new attributes, and the reference manual sections describing the portions of dakota.input.nspec that have been modified should be updated. In particular, the reference manual tables summarizing keywords provide help data to the Jaguar user interface so need to be kept updated.

Chapter 4

Understanding Iterator Flow

This page explains the various phases comprising Iterator::run_iterator(). Prior to Iterator construction, when command-line options are parsed, Boolean run mode flags corresponding to PRERUN, RUN, and POSTRUN are set in ParallelLibrary. If the user didn't specify any specific run modes, the default is for all three to be true (all phases will execute).

Iterator is constructed.

When called, `run_iterator()` sequences:

- `initialize_run()`: unconditionally called, virtual. Performs common initialization such as allocating workspaces, setting communicators and evaluation counts. When re-implementing this virtual, a derived class must call its nearest parent's initialize_run(), typically *before* performing its own implementation steps.
- *Not implemented: pre-run input*
- IF PRERUN, invoke `pre_run()`: virtual function; default no-op. Purpose: derived classes should implement `pre_run()` if they are able to generate all parameter sets (variables) at once, separate from `run()`. Derived implementations should call their nearest parent's pre_run(), typically *before* performing their own steps.
- IF PRERUN, invoke `pre_output()`: non-virtual function; if user requested, output variables to file.
- *Not implemented: run input*
- IF RUN, invoke virtual function `run()`. Purpose: at a minimum, evaluate parameter sets through computing responses; for iterators without pre/post capability, their entire implementation is in `run()` and this is a reasonable default for new Iterators.
- *Not implemented: run output*
- IF POSTRUN, invoke `post_input()`: virtual function, default only print helpful message on mode. Purpose: derived iterators supporting post-run input from file must implement to read file and populate variables/responses (and possibly best points) appropriately. Implementations must check if the user requested file input.
- IF POSTRUN, invoke `post_run()`: virtual function. Purpose: generate statistics / final results. Any analysis that can be done solely on tabular data read by `post_input()` can be done here. Derived re-implementations should call their nearest parent's post-run(), typically *after* performing their specific post-run activities.
- *Not implemented: post-run output*

- `finalize_run()`: unconditionally called, virtual. Purpose: free workspaces. Default base class behavior is no-op, however, derived implementations should call their nearest parent's finalize_run *after* performing their specialized portions.

Iterator is destructed.

Chapter 5

Interfacing with Dakota as a Library

5.1 Introduction

Tightly integrating or linking Dakota into another application can improve user experience by delivering a more unified, inter-operable software tool for optimization and UQ analyses, improving performance by eliminating file system-based interfaces, and reducing challenges with parallel computing inter-operation. This benefit has been realized within several Sandia and external simulation applications. This section describes how to link Dakota into another C++ application.

Dakota has two primary application programming interfaces (APIs). The LibraryEnvironment class facilitates use of Dakota as an algorithm service library within another application. In this case, the simulation application is providing a ”front end” for Dakota. The second API, provided by the DirectApplicInterface class, provides an interface for Dakota to call the simulation code directly to perform function evaluations in core. This permits the simulation to be the ”back end” for Dakota. The most complete library integration of Dakota would use both in combination, with the overall simulation framework providing both the front end and back end for Dakota, creating a sandwich, as loosely depicted here:

```
[------------
[ Application
[
[   ( -----
[   ( Dakota (LibraryEnvironment)
[   (
[   (   { Function evaluation callback to Application (via DirectApplicInterface)
[   (   {   |
[ <------/
[   (   {
[   (
[   ( -----
[
[------------
```

Attention

> Dakota may be integrated as a library in other software applications subject to the terms of the GNU Lesser General Public License (LGPL). Refer to http://www.gnu.org/licenses/lgpl.html or the LICENSE file included with Dakota.

When Dakota is compiled and installed, the relevant library API headers are installed to CMAKE_INSTALL_PREFIX/include and the runtime libraries primarily to CMAKE_INSTALL_PREFIX/lib/ (on some platforms,

to CMAKE_INSTALL_PREFIX/bin/. The core C/C++ code is in the library dakota_src, while Fortran code lives in the dakota_src_fortran library. Information on using the API in Dakota headers is included throughout this section, while considerations for configuring and linking against Dakota and its various required and optional third–party libraries are emphasized in the section Linking against the Dakota library.

Steps involved in integrating Dakota into another application typically include:

1. Writing C++ code for your application to instantiate, configure, and execute Dakota's LibraryEnvironment ("front end"); see Basic Dakota library instantiation and Configuring Dakota operation.

2. Writing C++ code for Dakota to call a function in your application to perform function evaluations ("back end"); see Creating a simulator plugin interface.

3. Compiling Dakota and linking into your application (Linking against the Dakota library).

Several source code examples demonstrate Dakota library interfaces. The classes SIM::SerialDirectApplicInterface and SIM::ParallelDirectApplicInterface demonstrate serial and parallel simulation function evaluation plug-ins. The file library_mode.cpp includes a main program that exercises Dakota libraries in serial and parallel modes with these mock simulator programs, with various ways of configuring Dakota problem definition and operation. Finally, library_split.cpp demonstrates running Dakota as a library modular on an MPI sub-communicator.

5.2 Basic Dakota library instantiation

The function run_dakota_parse() in library_mode.cpp demonstrates the basic use of Dakota library objects as one would in another main application that embeds Dakota. In this example, Dakota is configured based on a typical user-provided text-based Dakota input file (the same that would be provided at the command line with dakota -i dakota_optimization.in) and a function evaluator derived from a DirectApplicInterface is plugged into the Dakota library environment.

First, an object of type ProgramOptions which manages top-level Dakota settings is instantiated and configured to specify the name of the Dakota user input file. Additional options for output and error redirection, restart operation, and more may be set via ProgramOptions. See its class documentation for details.

```
string dakota_input_file = "dakota_optimization.in";
Dakota::ProgramOptions opts;
opts.input_file(dakota_input_file);
```

Next, a LibraryEnvironment is created, passing the desired settings from opts:

```
Dakota::LibraryEnvironment env(opts);
```

This standard constructor will parse the specified input and create Dakota objects. It assumes many default settings, including that the parent application initialized MPI if running in parallel mode. (In this case, Dakota will detect whether MPI was initialized and not call MPI_Init or MPI_Finalize.) For more advanced use cases described below, alternate constructors allow constructing based on MPI communicators, with delayed finalization, and with Dakota database update function callbacks. Then the application's function evaluator implementing Dakota's DirectApplicInterface is plugged in with a convenience function serial_interface_plugin() or parallel_interface_plugin(). Finally, the Dakota analysis is run by calling

```
env.execute();
```

The next two sections offer additional details on (1) alternative and supplementary ways to configure Dakota's operation (Configuring Dakota operation) and (2) how to specialize Dakota's DirectApplicInterface to provide a function evaluator plugin to Dakota (Creating a simulator plugin interface).

Remarks

After LibraryEnvironment construction, all MPI communicator partitioning has been performed and the ParallelLibrary instance may be interrogated for parallel configuration data. For example, the lowest level communicators in Dakota's multilevel parallel partitioning are the analysis communicators, which can be retrieved using:

```
// retrieve the set of analysis communicators for simulation initialization:
// one analysis comm per ParallelConfiguration (PC), one PC per Model.
Array<MPI_Comm> analysis_comms = parallel_lib.analysis_intra_communicators();
```

These communicators can then be used for initializing parallel simulation instances when registering the plugin interface, where the number of MPI communicators in the array corresponds to one communicator per ParallelConfiguration instance. This is demonstrated below in Derivation.

5.3 Configuring Dakota operation

This section describes several alternate ways to initially set and later manipulate Dakota's configuration, including alternatives to using a text-based input file. The algorithm configuration for a particular Dakota analysis run is managed in its ProblemDescDB, which can be populated via an input file, string literal, or C++ API, and later modified through Dakota's C++ API. All Dakota objects then draw information from this database upon instantiation.

5.3.1 Input data parsing

The simplest way for an application to configure a Dakota analysis problem is to use Dakota's normal input parsing system to populate its problem database (ProblemDescDB). This is done by providing standard Dakota input file syntax through the library interface, via either a file name or string literal. An advantage is that native Dakota syntax can be used, but disadvantages include the requirement for an additional input file beyond those already required by the parent application and that application users also need to know Dakota syntax.

The two ways to configure Dakota via input parsing are shown near the beginning of run_dakota_mixed() in library_mode.cpp. Here the ProgramOptions are set to either parse from a named file:

```
Dakota::ProgramOptions opts;
opts.input_file(dakota_input_file);
```

or from a string literal provided by the wrapping application:

```
string serial_input = "% Dakota input file ...";
opts.input_string(serial_input);
```

This library approach is coarse-grained in that input is parsed, objects constructed, and the environment is immediately ready to run. The next approaches are more modular.

5.3.2 Problem database insertion

A second approach to configuring Dakota's operation is to bypass parsing phases and directly populate the ProblemDescDB with information on the methods, variables, interface, responses, etc., that define the Dakota analysis problem. This approach requires more interaction with Dakota classes and data structures. However, it can offer usability benefit when the integrating application does not want their users to interact with the full Dakota syntax, or efficiency benefit when for example there are a large number of variables to configure.

In the direct database population approach, Dakota DataMethod, DataModel, DataVariables, DataInterface, and DataResponses objects are instantiated and populated with the desired problem data. These objects are then

published to the problem database using insert_nodes() . An example of this approach is available in run_dakota-_data() in library_mode.cpp, where the OPT++ Quasi-Newton method is configured to work on a plugin version of `text_book` or `rosenbrock`. The data objects are populated with their default values upon instantiation and are often sufficient for basic Dakota studies. Only the non-default values need to be specified. Moreover the default Dakota Model is a SingleModel, so this object need not be configured unless tailoring its configuration or using a more advanced model type. Refer to the DataMethod, DataModel, DataVariables, DataInterface, and DataResponses class documentation and source code for lists of attributes and their defaults. Here is an excerpt of run_dakota_data() that specifies the OPT++ solver after default construction of DataMethod:

```
Dakota::DataMethod   dme;
Dakota::DataMethodRep* dmr = dme.data_rep();
dmr->methodName = Dakota::OPTPP_Q_NEWTON;
```

When using direct database population, it is critical to leave the database in an open, accessible state after initial construction. In this run_dakota_data() example, a flag `check_bcast_construct` is passed into the LibraryEnvironment constructor, indicating that it should not finalize the database and construct Dakota objects. Moreover, it is only necessary to populate the database on rank 0 of the MPI Comm on which Dakota is running. After database objects are inserted or adjusted, the LibraryEnvironment::done_modifying_db() function must be called before proceeding to execute. This synchronizes problem data across all ranks and constructs Dakota objects needed to run the specified analysis.

```
bool check_bcast_construct = false;
Dakota::LibraryEnvironment env(MPI_COMM_WORLD, opts, check_bcast_construct);
if (rank == 0)
  // insert/modify DB, then lock and proceed:
env.done_modifying_db();
env.execute();
```

5.3.3 Mixed mode, callbacks, and late updates

The LibraryEnvironment API also supports mixed approaches that combine the parsing of a Dakota input file (or input string literal) with direct database updates. This approach is motivated by large-scale applications where large vectors are cumbersome to specify in a Dakota input file or where later updates to an input template are needed. The example run_dakota_mixed() in library_mode.cpp demonstrates the combination of these more advanced approaches: (1) input text parsing, (2) database updates via a callback, (3) database updates via direct manipulation, and (4) further runtime updates to the Model before running.

First, a ProgramOptions class is instantiated and configured to parse either an input file or input string literal (as in earlier examples). The passed input data must contain all required inputs so the parser can validate them. Since vector data like variable values/bounds/tags, linear/nonlinear constraint coefficients/bounds, etc., are optional, these potentially large vector specifications can be omitted from the input file and updated later through the database API. Only the variable/response counts necessary for sizing, e.g.:

```
method
  linear_inequality_constraints = 500

variables
  continuous_design = 1000

responses
  objective_functions = 1
  nonlinear_inequality_constraints = 100000
```

and not the lists of values are required in this case. To update or add data after this initial parse, we use the ProblemDescDB::set() family of overloaded functions, e.g.

```
Dakota::RealVector drv(1000, 1.); // vector of length 1000, values initialized to 1.
problem_db.set("variables.continuous_design.initial_point", drv);
```

where the string identifiers are the same identifiers used when pulling information from the database using one of the get_<datatype>() functions (refer to ProblemDescDB for a full list). However, the supported ProblemDescDB::set() options are a restricted subset of the database attributes, focused on vector inputs that can be large scale.

Second, the example demonstrates a user-provided callback function which Dakota will invoke after input parsing to update ProblemDescDB. In `library_mode.cpp`, callback_function() is a user-provided post-parse callback that implements the type Dakota::DbCallbackFunction.

```
static void callback_function(Dakota::ProblemDescDB* db, void *ptr);
```

When Dakota calls this function it will pass back pointers to the ProblemDescDB instance and to user-provided data, so the application may convey its settings by calling methods on the ProblemDescDB, optionally using the provided data. An example of a user data structure is demonstrated in `callback_data`. In this case, when the LibraryEnvironment is constructed, it is constructed with the input data to initially parse, the callback function, and to leave it unlocked for further updates:

```
bool done_with_db = false;
Dakota::LibraryEnvironment env(opts, done_with_db,
      callback_function, &data);
```

Third, the example demonstrates changes to the database after parsing and callback-based updates. Again, these only need happen on Dakota's rank 0 before finalizing the DB with LibraryEnvironment::done_modifying_db(). The example demonstrates:

1. Getting access to the database through `env.problem_description_db()`
2. Setting the database nodes to the appropriate method through `problem_db.resolve_top_method()`
3. Getting data from the DB with a get string array function: `problem_db.get_sa`("interface.application.analysis_drivers")
4. Setting update data with `problem_db.set`("variables.continuous_design.initial_point", ip);

After any of these three types updates, calling LibraryEnvironment::done_modifying_db() will broadcast any updates (including potentially large vector data and post-process specification data to fill in any vector defaults that have not yet been provided through either file parsing or direct updates. (Note: scalar defaults are handled in the Data class constructors.)

Fourth and finally, run_dakota_mixed() demonstrates modifying a Model's data after database operations and interface plugin are complete. This involves finding the right Model (or other class) instance to modify, and directly adjusting its data through the public API. Since the database is finalized, any updates must be performed through direct set operations on the constructed objects. For example, to update other data such as variable values/bounds/tags or response bounds/targets/tags, refer to the set functions documented in Iterator and Model. As an example, the following code updates the active continuous variable values, which will be employed as the initial guess for certain classes of Iterators:

```
ModelList& all_models  = problem_db.model_list();
Model&     first_model = *all_models.begin();
Dakota::RealVector drv(1000, 1.); // vector of length 1000, values initialized to 1.
first_model.continuous_variables(drv);
```

Remarks

If performing such data updates within the constructor of a DirectApplicInterface extension/derivation (see Creating a simulator plugin interface), then this code is sufficient since the database is unlocked, the active list nodes of the ProblemDescDB have been set for you, and the correct method/model/variables/interface/responses specification instance will get updated. The difficulty in this case stems from the order of instantiation. Since the Variables and Response instances are constructed in the base Model class, prior to construction of Interface instances in derived Model classes, database information related to Variables and Response objects will have already been extracted by the time the Interface constructor is invoked and the database update will not propagate.

Therefore, it is preferred to perform these database set operations at a higher level (e.g., within your main program), prior to allowing Environment to broadcast, construct, and execute, such that instantiation order is not an issue. However, in this case, it is necessary to explicitly manage the list nodes of the ProblemDescDB using a specification instance identifier that corresponds to an identifier from the input file, e.g.:

```
problem_db.set_db_variables_node("MY_VARIABLES_ID");
Dakota::RealVector drv(1000, 1.); // vector of length 1000, values
initialized to 1.
problem_db.set("variables.continuous_design.initial_point", drv);
```

Alternatively, rather than setting just a single data node, all data nodes may be set using a method specification identifier:

```
problem_db.set_db_list_nodes("MY_METHOD_ID");
```

since the method specification is responsible for identifying a model specification, which in turn identifies variables, interface, and responses specifications. If hard-wiring specification identifiers is undesirable, then

```
problem_db.resolve_top_method();
```

can also be used to deduce the active method specification and set all list nodes based on it. This is most appropriate in the case where only single specifications exist for method/model/variables/interface/responses. This is the approach demonstrated in run_dakota_mixed(). In each of these cases, setting list nodes unlocks the corresponding portions of the database, allowing set/get operations.

5.4 Creating a simulator plugin interface

The DirectApplicInterface class provides an interface for Dakota to call the simulation code directly to perform function evaluations mapping variables to responses. This provides the ”back end” for Dakota to call back to the simulation framework. Two approaches to defining this direct interface are described here. The first is less common, while the second is recommended when possible.

5.4.1 Extension

The first approach involves extending one of the existing DirectApplicInterface subclasses (TestDriverInterface, MatlabInterface, etc.) to support additional direct simulation interfaces. For example, Dakota algebraic test problems are implemented in TestDriverInterface. One could add additional direct functions to Dakota in TestDriverInterface::derived_map_ac(). In addition, TestDriverInterface::derived_map_if() and TestDriverInterface::derived_map_of() can be extended to perform pre- and post-processing tasks if desired, but this is not required.

While this approach is the simplest, it has the disadvantage that the Dakota library will need to be recompiled when the simulation or its direct interface is modified. If it is desirable to maintain the independence of the Dakota library from the host application, then the derivation approach described in the next section should be employed.

Remarks

If the new direct evaluation function implementation will not be a member function of one of the Dakota classes, then the following prototype should be used in order to pass the required data:

```
int sim(const Dakota::Variables& vars, const Dakota::ActiveSet& set,
Dakota::Response& response);
```

If the new function will be a member function, e.g., in TestDriverInterface, then this can be simplified to

```
int sim();
```

since the data access can be performed through the DirectApplicInterface class attributes.}

5.4.2 Derivation

The second approach is to derive a new interface from DirectApplicInterface and redefine several virtual functions. As demonstrated in SIM::SerialDirectApplicInterface and SIM::ParallelDirectApplicInterface, a typical derived class declaration might be

```
namespace SIM {

class SerialDirectApplicInterface: public Dakota::DirectApplicInterface
{
public:

  // Constructor and destructor

  SerialDirectApplicInterface(const Dakota::ProblemDescDB& problem_db);
  ~SerialDirectApplicInterface();

protected:

  // Virtual function redefinitions

  int derived_map_if(const Dakota::String& if_name);
  int derived_map_ac(const Dakota::String& ac_name);
  int derived_map_of(const Dakota::String& of_name);

private:

  // Data
}

} // namespace SIM
```

where the new derived class resides in the simulation's namespace. Similar to the case of Extension, the DirectApplicInterface::derived_map_ac() function is the required redefinition, and DirectApplicInterface::derived_map_if() and DirectApplicInterface::derived_map_of() are optional.

Typically the new derived_map_ac() implementation delegates to the main simulation application for a function evaluation. Here Dakota variables would get mapped into the simulation's data structures, the simulation executed, and derived response data computed for return to Dakota.

Once a derived application class is created, it must be plugged in, or registered, with the appropriate Interface in the LibraryEnvironment. In MPI cases where Dakota is potentially managing concurrent evaluations of the simulation, the plugin must be configured to run on the right MPI sub-communicator, or Dakota analysis_comm. The simpler case is demonstrated in serial_interface_plugin() in library_mode.cpp, while a more advanced case using the analysis communicator is shown in parallel_interface_plugin().

The Dakota LibraryEnvironment provides a convenience function to plugin an Interface. This example will replace any interface found matching the given model, interface, and analysis driver with the passed plugin interface:

```
std::string model_type(""); // demo: empty string will match any model type
std::string interf_type("direct");
std::string an_driver("plugin_rosenbrock");

Dakota::ProblemDescDB& problem_db = env.problem_description_db();
Dakota::Interface* serial_iface =
  new SIM::SerialDirectApplicInterface(problem_db);

bool plugged_in =
  env.plugin_interface(model_type, interf_type, an_driver, serial_iface);
```

The LibraryEnvironment also provides convenience functions that allow the client to iterate the lists of available interfaces or models for more advanced cases. For instance if the client knows there is only a single interface active, it could get the list of available interfaces of length 1 and plugin to the first one. In the more advanced case where the simulation interface instance should manage parallel simulations within the context of an MPI communicator, one should pass in the relevant analysis communicator(s) to the derived constructor. For the latter case of looping over a set of models, the simplest approach of passing a single analysis communicator would use code similar to

```
Dakota::ModelList filt_models =
  env.filtered_model_list("single", "direct", "plugin_text_book");

Dakota::ProblemDescDB& problem_db = env.problem_description_db();
Dakota::ModelLIter ml_iter;
for (ml_iter = filt_models.begin(); ml_iter != filt_models.end(); ++ml_iter) {
  // set DB nodes to input specification for this Model
  problem_db.set_db_model_nodes(ml_iter->model_id());

  Dakota::Interface& model_interface = ml_iter->derived_interface();

  // Parallel case: plug in derived Interface object with an analysisComm.
  // Note: retrieval and passing of analysisComm is necessary only if
  // parallel operations will be performed in the derived constructor.

  // retrieve the currently active analysisComm from the Model.  In the most
  // general case, need an array of Comms to cover all Model configurations.
  const MPI_Comm& analysis_comm = ml_iter->analysis_comm();

  // don't increment ref count since no other envelope shares this letter
  model_interface.assign_rep(new
    SIM::ParallelDirectApplicInterface(problem_db, analysis_comm), false)
      ;
}
```

The file library_mode.cpp demonstrates each of these approaches. Since a Model may be used in multiple parallel contexts and may therefore have a set of parallel configurations, a more general approach would extract and pass an array of analysis communicators to allow initialization for each of the parallel configurations.

New derived direct interface instances inherit various attributes of use in configuring the simulation. In particular, the ApplicationInterface::parallelLib reference provides access to MPI communicator data (e.g., the analysis communicators discussed above), DirectApplicInterface::analysisDrivers provides the analysis driver names specified by the user in the input file, and DirectApplicInterface::analysisComponents provides additional analysis component identifiers (such as mesh file names) provided by the user which can be used to distinguish different instances of the same simulation interface. It is worth noting that inherited attributes that are set as part of the parallel configuration (instead of being extracted from the ProblemDescDB) will be set to their defaults following construction of the base class instance for the derived class plug-in. It is not until run-time (i.e., within derived_map_if/derived_map_ac/derived_map_of) that the parallel configuration settings are re-propagated to the plug-in instance. This is the reason that the analysis communicator should be passed in to the constructor of a parallel plug-in, if the constructor will be responsible for parallel application initialization.

5.5 Retrieving data after a run

After executing the Dakota Environment, final results can be obtained through the use of Environment::variables_results() and Environment::response_results(), e.g.:

```
// retrieve the final parameter values
const Variables& vars = env.variables_results();

// retrieve the final response values
const Response& resp  = env.response_results();
```

In the case of optimization, the final design is returned, and in the case of uncertainty quantification, the final statistics are returned. Dakota has a prototype results database, which will eventually provide better access to the results from a study.

5.6 Linking against the Dakota library

This section presumes Dakota has been configured with CMake, compiled, and installed to a `CMAKE_INSTALL_PREFIX` using `make install` or equivalent. The Dakota libraries against which you must link will typically install to `CMAKE_INSTALL_PREFIX/bin/` and `CMAKE_INSTALL_PREFIX/lib/`, while headers are provided in `CMAKE_INSTALL_PREFIX/lib/`. The core Dakota C and C++ code is in the library `dakota_src`, while Fortran code lives in the `dakota_src_fortran` library. Runtime libraries for any configure-enabled Dakota third-party software components (such as DOT, NPSOL, OPT++, LHS, etc.) are also installed to the lib/ directory. Applications link against these Dakota libraries by specifying appropriate include and link directives.

There two primary ways to determine the necessary Dakota-related libraries and link order for linking your application. First, when running CMake, a list of required Dakota and Dakota-included third-party libraries will be output to the console, e.g.,

```
-- Dakota_LIBRARIES: dakota_src;dakota_src_fortran;nidr;teuchos;pecos;pecos_src;lhs;mods;mod;dfftpack;sparseg
```

While external dependencies will be output as:

```
-- Dakota_TPL_LIBRARIES: /usr/lib64/libcurl.so;/usr/lib64/openmpi/lib/libmpi_cxx.so;debug;/usr/lib64/libz.so;
```

Note that depending on how you configured Dakota, some libraries may be omitted from these lists (for example commercial add-ons NPSOL, DOT, and NLPQL), or additional libraries may appear.

A second option is to check which libraries appear in `CMAKE_INSTALL_PREFIX/bin/` `CMAKE_INSTALL_PREFIX/lib/`, or more accurately, see the file Makefile.export.Dakota in the Dakota build/src/ or installation include/ directory. Here are some additional notes on specific libraries:

- Some Boost libraries (`boost_regex`, `boost_filesystem`, `boost_system`, `boost_serialization`) are required, and other Boost library components may be required depending on configuration, e.g., `boost_signals` when configuring with `HAVE_ACRO:BOOL=TRUE`
- System compiler and math libraries may need to be included, as may additional system libraries such as Expat and Curl, depending on how Dakota is configured.
- If configuring with graphics, you will need to add the Dakota `DGraphics` library and system X libraries (partial list here):

  ```
  -lXpm -lXm -lXt -lXmu -lXp -lXext -lX11 -lSM -lICE
  ```

- When configuring with AMPL (`HAVE_AMPL:BOOL=ON`), the AMPL solver library may require `dl`, `funcadd0.-o` and `fl` libraries. We have experienced problems with the creation of `libamplsolver.a` on some platforms; use the dakota-users mailing list to get help with any problems related to this.
- Optional library GSL (discouraged due to GPL license) and if linking with system-provided GSL, `gslcblas` may be needed if Dakota was configured with them.
- Newmat: as of Dakota 5.2, `-lnewmat` is no longer required

Finally, it is important to use the same C++ compiler (possibly an MPI wrapper) for compiling Dakota and your application and potentially include Dakota-related preprocessor defines as emitted by CMake during compilation of Dakota and included in Makefile.export.Dakota. This ensures that the platform configuration settings are properly synchronized across Dakota and your application.

Chapter 6

Performing Function Evaluations

Performing function evaluations is one of the most critical functions of the Dakota software. It can also be one of the most complicated, as a variety of scheduling approaches and parallelism levels are supported. This complexity manifests itself in the code through a series of cascaded member functions, from the top level model evaluation functions, through various scheduling routines, to the low level details of performing a system call, fork, or direct function invocation. This section provides an overview of the primary classes and member functions involved.

6.1 Synchronous function evaluations

For a synchronous (i.e., blocking) mapping of parameters to responses, an iterator invokes Model::compute_response() to perform a function evaluation. This function is all that is seen from the iterator level, as underlying complexities are isolated. The binding of this top level function with lower level functions is as follows:

- Model::compute_response() utilizes Model::derived_compute_response() for portions of the response computation specific to derived model classes.
- Model::derived_compute_response() directly or indirectly invokes Interface::map().
- Interface::map() utilizes ApplicationInterface::derived_map() for portions of the mapping specific to derived application interface classes.

6.2 Asynchronous function evaluations

For an asynchronous (i.e., nonblocking) mapping of parameters to responses, an iterator invokes Model::asynch_compute_response() multiple times to queue asynchronous jobs and then invokes either Model::synchronize() or Model::synchronize_nowait() to schedule the queued jobs in blocking or nonblocking fashion. Again, these functions are all that is seen from the iterator level, as underlying complexities are isolated. The binding of these top level functions with lower level functions is as follows:

- Model::asynch_compute_response() utilizes Model::derived_asynch_compute_response() for portions of the response computation specific to derived model classes.
- This derived model class function directly or indirectly invokes Interface::map() in asynchronous mode, which adds the job to a scheduling queue.
- Model::synchronize() or Model::synchronize_nowait() utilize Model::derived_synchronize() or Model::derived_synchronize_nowait() for portions of the scheduling process specific to derived model classes.

- These derived model class functions directly or indirectly invoke Interface::synch() or Interface::synch_-nowait().
- For application interfaces, these interface synchronization functions are responsible for performing evaluation scheduling in one of the following modes: master dynamic, peer dynamic or peer static.
- *NOTE: The Interface evaluation scheduling in Dakota was refactored for releases 5.4 and 6.0. Discussion of the new Interface-related functions is currently missing here.*

6.3 Analyses within each function evaluation

NOTE: The Interface evaluation scheduling in Dakota was refactored for releases 5.4 and 6.0. Discussion of the new Interface-related functions for analyses is currently missing here.

The discussion above covers the parallelism level of concurrent function evaluations serving an iterator. For the parallelism level of concurrent analyses serving a function evaluation, similar schedulers are involved

to support synchronous local, asynchronous local, message passing, and hybrid modes. Not all of the schedulers are elevated to the ApplicationInterface level since the system call and direct function interfaces do not yet support nonblocking local analyses (and therefore support synchronous local and message passing modes, but not asynchronous local or hybrid modes). Fork interfaces, however, support all modes of analysis parallelism.

Chapter 7

Working with Variable Containers and Views

Variable views control the subset of variable types that are active and inactive within a particular iterative study. For design optimization and uncertainty quantification (UQ), for example, the active variables view consists of design or uncertain types, respectively, and any other variable types are carried along invisible to the iterative algorithm being employed. For parameter studies and design of experiments, however, a variable subset view is not imposed and all variables are active. Selected UQ methods can also be toggled into an "All" view using the `active all` variables input specification. When not in an All view, finer gradations within the uncertain variable sets are also relevant: probabilistic methods (reliability, stochastic expansion) view aleatory uncertain variables as active, nonprobabilistic methods (interval, evidence) view epistemic uncertain variables as active, and a few UQ methods (sampling) view both as active. In a more advanced NestedModel use case such as optimization under uncertainty, design variables are active in the outer optimization context and the uncertain variables are active in the inner UQ context, with an additional requirement on the inner UQ level to return derivatives with respect to its "inactive" variables (i.e., the design variables) for use in the outer optimization loop.

For efficiency, contiguous arrays of data store variable information for each of the domain types (continuous, discrete integer, and discrete real), but active and inactive views into them permit selecting subsets in a given context. This management is encapsulated into the Variables and SharedVariablesData classes. This page clarifies concepts of relaxed (formerly merged) vs. mixed, fine-grained vs. aggregated types, domain types, and views into contiguous arrays.

We begin with an overview of the storage and management concept, for which the following two sections describe the storage of variable values and meta-data about their organization, used in part to manage views. They are intended to communicate rationale to maintainers of Variables and SharedVariablesData classes. The final section provides a discussion of active and inactive views.

7.1 Storage in Variables

As described in the Main Page Variables, a Variables object manages variable types (design, aleatory uncertain, epistemic uncertain, and state) and domain types (continuous, discrete integer, and discrete real) and supports different approaches to either distinguishing among these types or aggregating them. Two techniques are used in cooperation to accomplish this management: (1) class specialization (RelaxedVariables or MixedVariables) and (2) views into contiguous variable arrays. The latter technique is used whenever it can satisfy the requirement, with fallback to class specialization when it cannot. In particular, aggregation or separation of variable types can be accomplished with views, but for aggregation or separation of variable domains, we must resort to class

specialization in order to relax discrete domain types. In this class specialization, a RelaxedVariables object combines continuous and discrete types (relaxing integers to reals) whereas a MixedVariables object maintains the integer/real distinction throughout.

The core data for a Variables instance is stored in a set of three continguous arrays, corresponding to the domain types: allContinuousVars, allDiscreteIntVars, and allDiscreteRealVars, unique to each Variables instance.

Within the core variable data arrays, data corresponding to different aggregated variable types are stored in sequence for each domain type:

- continuous: [design, aleatory uncertain, epistemic uncertain, state]
- discrete integer: [design, aleatory uncertain, (epistemic uncertain), state]
- discrete real: [design, aleatory uncertain, (epistemic uncertain), state]

Note there are currently no epistemic discrete variables. This domain type ordering (continuous, discrete integer, discrete real) and aggregated variable type ordering (design, aleatory uncertain, epistemic uncertain, state) is preserved whenever distinct types are flattened into single contiguous arrays. Note that the aleatory and epistemic uncertain variables contain sub-types for different distributions (e.g., normal, uniform, histogram, poisson), and discrete integer types include both integer ranges and integer set sub-types. All sub-types are ordered according to their order of appearance in dakota.input.nspec.

When relaxing in MixedVariables, the allContinuousVars will also aggregate the discrete types, such that they contain ALL design, then ALL uncertain, then ALL state variables, each in aggregated type order; the allDiscrete-IntVars and allDiscreteRealVars arrays are empty.

7.2 Storage in SharedVariablesData

Each Variables instance contains a reference-counted SharedVariablesData object that stores information on the variables configuration. This configuration data includes counts, types, IDs, and labels, which are often the same across many Variables instances. Thus, SharedVariablesData is intended to reduce the memory footprint by allowing the sharing of a single copy of redundant information among different Variables instances.

One of the purposes of this shared information is to support mappings between variable types, IDs, and indices into the storage arrays. Variable "types" refer to the fine-grained variable types a user would specify in an input file, as enumerated in DataVariables.hpp, e.g, CONTINUOUS_DESIGN, WEIBULL_UNCERTAIN, DISCRETE_STATE_RANGE, etc. variablesComponents is a map from these variable types to counts of how many are present.

In contrast, the variablesCompsTotals array stores total counts of each "aggregated type" (design, aleatory uncertain, epistemic uncertain, state) which might be selected to be active in a given view. Thus this array has length 12 to track the combinations of three domain type storage arrays with four possible aggregated variable types: {continuous, discrete integer, discrete real} x {design, aleatory uncertain, epistemic uncertain, state}. For example, the first entry of this array stores the number of continuous design variables, the second the number of discrete integer design (including both discrete design range and discrete design set integer types), and the last the number of discrete real state variables.

The arrays allContinuousTypes, allDiscreteIntTypes, and allDiscreteRealTypes are sized to match the corresponding core domain type storage arrays. They track the fine-grained variable type stored in that entry of the data array (since when relaxed, the continuous array may be storing data corresponding to discrete data).

Finally allContinuousIds stores the 1-based IDs of the variables stored in the allContinuousVars array, i.e., the variable number of all the problem variables considered as a single contiguous set, in aggregate type order. For relaxed (formerly merged) views, relaxedDiscreteIds stores the 1-based IDs of the variables which have been relaxed into the continuous array.

These counts, types, and IDs are most commonly used within the Model classes for mappings between variables objects at different levels of a model recursion. See, for example, the variable mappings in the NestedModel constructor.

7.3 Active and inactive views

The pair SharedVariablesDataRep::variablesView tracks the active and inactive views of the data, with values taken from the enum in DataVariables.hpp. The valid values include EMPTY and the combinations {relaxed, mixed} x {all, design, aleatory uncertain, epistemic uncertain, uncertain, state}. The ALL cases indicate aggregation of the design, aleatory uncertain, epistemic uncertain, and state types, whereas the DISTINCT cases indicate either no aggregation (design, aleatory uncertain, epistemic uncertain, state) or reduced aggregation (aleatory+epistemic uncertain). The active view is determined by the algorithm in use, managed in Variables::get_view(). Any inactive view is set based on higher level iteration within a model recursion (e.g., a NestedModel), which enables lower level iteration to return derivatives with respect to variables that are active at the higher level. In the case where there is no higher level iteration, then the inactive view will remain EMPTY. It is important to stress that "inactive" at one level corresponds to active at another, and therefore the inactive set of variables should not be interpreted as the strict complement of the active set of variables; rather, active and inactive are both subsets whose union may still be a subset of the total container (more precise terminology might involve "primary" active and "secondary" active or similar). An active complement view could potentially be supported in the future, should the need arise, although this view would require management of non-contiguous portions of the aggregated arrays.

Given these groupings (views), the active and inactive subsets of the allContinuousVars, allDiscreteIntVars, and allDiscreteRealVars arrays are always contiguous, permitting vector views of the underlying data using either Teuchos::View (for numerical vectors) or Boost.MultiArray (for book-keeping arrays) views.

When a Variables envelope is constructed, its letter is initialized to either a RelaxedVariables or MixedVariables object depending on the active view. The derived classes size the contiguous storage arrays to accomodate all the problem variables, and then initialize active views into them, which could involve either subsets (DISTINCT active views) or views of the full arrays (ALL active views). Inactive views, on the other hand, are initialized during construction of a model recursion (e.g., a call to Model::inactive_view() in the NestedModel constructor). Thus, active variable subsets are always available but inactive variable subsets will be EMPTY prior to them being initialized within a Model recursion.

Accessors for continuous variables include:

- continuous_variables(): returns the active view which might return all (ALL views) or a subset (DISTINCT views) such as design, uncertain, only aleatory uncertain, etc.
- inactive_continuous_variables(): returns the inactive view which which is either a subset or empty
- all_continuous_variables(): returns the full vector allContinuousVars

and this pattern is followed for active/inactive/all access to discrete_int_variables() and discrete_real_variables() as well as for labels, IDs, and types in SharedVariablesData and variable bounds in Constraints.

Member SurfpackApproximation::build ()

Right now, we're completely deleting the old data and then recopying the current data into a SurfData object. This was just the easiest way to arrive at a solution that would build and run. This function is frequently called from addPoint rebuild, however, and it's not good to go through this whole process every time one more data point is added.

Member SurfpackApproximation::hessian (const Variables &vars)

Make this acceptably efficient

Chapter 8

Namespace Index

8.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Dakota
The primary namespace for DAKOTA 71
SIM
A sample namespace for derived classes that use assign_rep() to plug facilities into DAKOTA 306

Chapter 9

Hierarchical Index

9.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ActiveSet . . . 307
Approximation . . . 330
  GaussProcApproximation . . . 476
  PecosApproximation . . . 761
  SurfpackApproximation . . . 872
  TANA3Approximation . . . 895
  TaylorApproximation . . . 897
APPSEvalMgr . . . 342
BaseConstructor . . . 347
callback_data . . . 347
COLINApplication . . . 348
CommandShell . . . 358
ConsoleRedirector . . . 369
Constraints . . . 370
  MixedVarConstraints . . . 547
  RelaxedVarConstraints . . . 804
DataEnvironment . . . 379
DataEnvironmentRep . . . 380
DataInterface . . . 393
DataMethod . . . 394
DataMethodRep . . . 395
DataModel . . . 408
DataModelRep . . . 409
DataResponses . . . 413
DataResponsesRep . . . 414
DataVariables . . . 418
DataVariablesRep . . . 419
DiscrepancyCorrection . . . 440
JEGAOptimizer::Driver . . . 447
Environment . . . 457
  ExecutableEnvironment . . . 469

LibraryEnvironment . . . 529
NomadOptimizer::Evaluator . . . 461
JEGAOptimizer::Evaluator . . . 463
JEGAOptimizer::EvaluatorCreator . . . 468
ExperimentData . . . 470
GetLongOpt . . . 481
CommandLineHandler . . . 356
Graphics . . . 484
Interface . . . 491
ApplicationInterface . . . 316
DirectApplicInterface . . . 436
MatlabInterface . . . 532
PythonInterface . . . 793
ScilabInterface . . . 827
TestDriverInterface . . . 898
ParallelDirectApplicInterface . . . 738
SerialDirectApplicInterface . . . 834
ProcessApplicInterface . . . 775
ProcessHandleApplicInterface . . . 779
ForkApplicInterface . . . 471
SpawnApplicInterface . . . 871
SysCallApplicInterface . . . 892
GridApplicInterface . . . 486
ApproximationInterface . . . 337
Iterator . . . 500
Analyzer . . . 309
NonD . . . 616
EfficientSubspaceMethod . . . 451
NonDCalibration . . . 636
NonDBayesCalibration . . . 634
NonDDREAMBayesCalibration . . . 641
NonDGPMSABayesCalibration . . . 662
NonDQUESOBayesCalibration . . . 703
NonDExpansion . . . 644
NonDPolynomialChaos . . . 695
NonDStochCollocation . . . 717
NonDIntegration . . . 667
NonDCubature . . . 638
NonDQuadrature . . . 699
NonDSparseGrid . . . 714
NonDInterval . . . 670
NonDGlobalInterval . . . 652
NonDGlobalEvidence . . . 651
NonDGlobalSingleInterval . . . 658
NonDLHSInterval . . . 673
NonDLHSEvidence . . . 672
NonDLHSSingleInterval . . . 676

NonDLocalInterval . . . 679
NonDLocalEvidence . . . 678
NonDLocalSingleInterval . . . 691
NonDPOFDarts . . . 692
NonDReliability . . . 705
NonDGlobalReliability . . . 655
NonDLocalReliability . . . 681
NonDSampling . . . 707
NonDAdaptImpSampling . . . 626
NonDAdaptiveSampling . . . 629
NonDGPImpSampling . . . 659
NonDIncremLHSSampling . . . 665
NonDLHSSampling . . . 674
PStudyDACE . . . 788
DDACEDesignCompExp . . . 432
FSUDesignCompExp . . . 473
ParamStudy . . . 755
PSUADEDesignCompExp . . . 790
Verification . . . 916
RichExtrapVerification . . . 825
MetaIterator . . . 533
CollabHybridMetaIterator . . . 355
ConcurrentMetaIterator . . . 359
EmbedHybridMetaIterator . . . 455
SeqHybridMetaIterator . . . 831
Minimizer . . . 535
LeastSq . . . 525
NL2SOLLeastSq . . . 604
NLSSOLLeastSq . . . 611
SNLLLeastSq . . . 859
Optimizer . . . 727
APPSOptimizer . . . 344
COLINOptimizer . . . 351
CONMINOptimizer . . . 362
DOTOptimizer . . . 443
JEGAOptimizer . . . 519
NCSUOptimizer . . . 585
NLPQLPOptimizer . . . 607
NomadOptimizer . . . 613
NonlinearCGOptimizer . . . 719
NPSOLOptimizer . . . 722
OptDartsOptimizer . . . 724
SNLLOptimizer . . . 863
SurrBasedMinimizer . . . 883
EffGlobalMinimizer . . . 449
SurrBasedGlobalMinimizer . . . 875
SurrBasedLocalMinimizer . . . 876
IteratorScheduler . . . 514

LightWtBaseConstructor . . . 531
MatchesWC . . . 531
Model . . . 549
  NestedModel . . . 588
  RecastModel . . . 795
  SingleModel . . . 855
  SurrogateModel . . . 888
    DataFitSurrModel . . . 382
    HierarchSurrModel . . . 487
MPIManager . . . 580
MPIPackBuffer . . . 581
MPIUnpackBuffer . . . 583
NL2Res . . . 604
NoDBBaseConstructor . . . 613
OutputManager . . . 732
OutputWriter . . . 735
ParallelConfiguration . . . 736
ParallelLevel . . . 739
ParallelLibrary . . . 742
ParamResponsePair . . . 751
partial_prp_equality . . . 760
partial_prp_hash . . . 760
ProblemDescDB . . . 765
  NIDRProblemDescDB . . . 599
ProgramOptions . . . 784
Response . . . 806
ResponseRep . . . 810
RestartWriter . . . 817
ResultsDBAny . . . 818
ResultsEntry< StoredType > . . . 820
ResultsID . . . 821
ResultsManager . . . 822
ResultsNames . . . 824
SensAnalysisGlobal . . . 828
SharedApproxData . . . 835
  SharedPecosApproxData . . . 840
  SharedSurfpackApproxData . . . 843
SharedVariablesData . . . 845
SharedVariablesDataRep . . . 850
SNLLBase . . . 857
  SNLLLeastSq . . . 859
  SNLLOptimizer . . . 863
SOLBase . . . 869
  NLSSOLLeastSq . . . 611
  NPSOLOptimizer . . . 722
TrackerHTTP . . . 903
UsageTracker . . . 905
Var_icheck . . . 906

Var_rcheck 906
Variables 907
MixedVariables 548
RelaxedVariables 805
VLint 918
VLreal 918
VLstr 919
WorkdirHelper 919

Chapter 10

Class Index

10.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ActiveSet
Container class for active set tracking information. Contains the active set request vector and the derivative variables vector . . . 307
Analyzer
Base class for NonD, DACE, and ParamStudy branches of the iterator hierarchy . . . 309
ApplicationInterface
Derived class within the interface class hierarchy for supporting interfaces to simulation codes 316
Approximation
Base class for the approximation class hierarchy . . . 330
ApproximationInterface
Derived class within the interface class hierarchy for supporting approximations to simulation-based results . . . 337
APPSEvalMgr
Evaluation manager class for APPSPACK . . . 342
APPSOptimizer
Wrapper class for HOPSPACK . . . 344
BaseConstructor
Dummy struct for overloading letter-envelope constructors . . . 347
callback_data . . . 347
COLINApplication . . . 348
COLINOptimizer
Wrapper class for optimizers defined using COLIN . . . 351
CollabHybridMetaIterator
Meta-iterator for hybrid iteration using multiple collaborating optimization and nonlinear least squares methods . . . 355
CommandLineHandler
Utility class for managing command line inputs to DAKOTA . . . 356
CommandShell
Utility class which defines convenience operators for spawning processes with system calls . 358
ConcurrentMetaIterator
Meta-iterator for multi-start iteration or pareto set optimization . . . 359

CONMINOptimizer
Wrapper class for the CONMIN optimization library . . . 362
ConsoleRedirector . . . 369
Constraints
Base class for the variable constraints class hierarchy . . . 370
DataEnvironment
Handle class for environment specification data . . . 379
DataEnvironmentRep
Body class for environment specification data . . . 380
DataFitSurrModel
Derived model class within the surrogate model branch for managing data fit surrogates (global and local) . . . 382
DataInterface
Handle class for interface specification data . . . 393
DataMethod
Handle class for method specification data . . . 394
DataMethodRep
Body class for method specification data . . . 395
DataModel
Handle class for model specification data . . . 408
DataModelRep
Body class for model specification data . . . 409
DataResponses
Handle class for responses specification data . . . 413
DataResponsesRep
Body class for responses specification data . . . 414
DataVariables
Handle class for variables specification data . . . 418
DataVariablesRep
Body class for variables specification data . . . 419
DDACEDesignCompExp
Wrapper class for the DDACE design of experiments library . . . 432
DirectApplicInterface
Derived application interface class which spawns simulation codes and testers using direct procedure calls . . . 436
DiscrepancyCorrection
Base class for discrepancy corrections . . . 440
DOTOptimizer
Wrapper class for the DOT optimization library . . . 443
JEGAOptimizer::Driver
A subclass of the JEGA front end driver that exposes the individual protected methods to execute the algorithm . . . 447
EffGlobalMinimizer
Implementation of Efficient Global Optimization/Least Squares algorithms . . . 449
EfficientSubspaceMethod
Efficient Subspace Method (ESM), as proposed by Hany S. Abdel-Khalik . . . 451
EmbedHybridMetaIterator
Meta-iterator for closely-coupled hybrid iteration, typically involving the embedding of local search methods within global search methods . . . 455

Environment
Base class for the environment class hierarchy 457
NomadOptimizer::Evaluator
NOMAD-based Evaluator class 461
JEGAOptimizer::Evaluator
An evaluator specialization that knows how to interact with Dakota 463
JEGAOptimizer::EvaluatorCreator
A specialization of the JEGA::FrontEnd::EvaluatorCreator that creates a new instance of a Evaluator 468
ExecutableEnvironment
Environment corresponding to execution as a stand-alone application 469
ExperimentData 470
ForkApplicInterface
Derived application interface class which spawns simulation codes using fork/execvp/waitpid 471
FSUDesignCompExp
Wrapper class for the FSUDace QMC/CVT library 473
GaussProcApproximation
Derived approximation class for Gaussian Process implementation 476
GetLongOpt
GetLongOpt is a general command line utility from S. Manoharan (Advanced Computer Research Institute, Lyon, France) 481
Graphics
Single interface to 2D (motif) and 3D (PLPLOT) graphics; there is only one instance of this OutputManager::dakotaGraphics 484
GridApplicInterface
Derived application interface class which spawns simulation codes using grid services such as Condor or Globus 486
HierarchSurrModel
Derived model class within the surrogate model branch for managing hierarchical surrogates (models of varying fidelity) 487
Interface
Base class for the interface class hierarchy 491
Iterator
Base class for the iterator class hierarchy 500
IteratorScheduler
Environment corresponding to execution as a stand-alone application 514
JEGAOptimizer
A version of Dakota::Optimizer for instantiation of John Eddy's Genetic Algorithms (JEGA) . 519
LeastSq
Base class for the nonlinear least squares branch of the iterator hierarchy 525
LibraryEnvironment
Environment corresponding to execution as an embedded library 529
LightWtBaseConstructor
Dummy struct for overloading constructors used in on-the-fly Model instantiations 531
MatchesWC
Predicate that returns true when the passed path matches the wild_card with which it was configured. Currently supports $*$ and ? 531
MatlabInterface 532

MetaIterator
Base class for meta-iterators 533
Minimizer
Base class for the optimizer and least squares branches of the iterator hierarchy 535
MixedVarConstraints
Derived class within the Constraints hierarchy which separates continuous and discrete variables (no domain type array merging) 547
MixedVariables
Derived class within the Variables hierarchy which separates continuous and discrete variables (no domain type array merging) 548
Model
Base class for the model class hierarchy 549
MPIManager
Class MPIManager to manage Dakota's MPI world, which may be a subset of MPI_COMM_WORLD 580
MPIPackBuffer
Class for packing MPI message buffers 581
MPIUnpackBuffer
Class for unpacking MPI message buffers 583
NCSUOptimizer
Wrapper class for the NCSU DIRECT optimization library 585
NestedModel
Derived model class which performs a complete sub-iterator execution within every evaluation of the model 588
NIDRProblemDescDB
The derived input file database utilizing the new IDR parser 599
NL2Res
Auxiliary information passed to calcr and calcj via ur 604
NL2SOLLeastSq
Wrapper class for the NL2SOL nonlinear least squares library 604
NLPQLPOptimizer
Wrapper class for the NLPQLP optimization library, Version 2.0 607
NLSSOLLeastSq
Wrapper class for the NLSSOL nonlinear least squares library 611
NoDBBaseConstructor
Dummy struct for overloading constructors used in on-the-fly instantiations without ProblemDescDB support 613
NomadOptimizer
Wrapper class for NOMAD Optimizer 613
NonD
Base class for all nondetermistic iterators (the DAKOTA/UQ branch) 616
NonDAdaptImpSampling
Class for the Adaptive Importance Sampling methods within DAKOTA 626
NonDAdaptiveSampling
Class for testing various Adaptively sampling methods using geometric, statisctical, and topological information of the surrogate 629
NonDBayesCalibration
Base class for Bayesian inference: generates posterior distribution on model parameters given experimental data 634

NonDCalibration 636
NonDCubature
Derived nondeterministic class that generates N-dimensional numerical cubature points for evaluation of expectation integrals 638
NonDDREAMBayesCalibration
Bayesian inference using the DREAM approach 641
NonDExpansion
Base class for polynomial chaos expansions (PCE) and stochastic collocation (SC) 644
NonDGlobalEvidence
Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ 651
NonDGlobalInterval
Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification 652
NonDGlobalReliability
Class for global reliability methods within DAKOTA/UQ 655
NonDGlobalSingleInterval
Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification 658
NonDGPImpSampling
Class for the Gaussian Process-based Importance Sampling method 659
NonDGPMSABayesCalibration
Generates posterior distribution on model parameters given experiment data 662
NonDIncremLHSSampling
Performs icremental LHS sampling for uncertainty quantification 665
NonDIntegration
Derived nondeterministic class that generates N-dimensional numerical integration points for evaluation of expectation integrals 667
NonDInterval
Base class for interval-based methods within DAKOTA/UQ 670
NonDLHSEvidence
Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ 672
NonDLHSInterval
Class for the LHS-based interval methods within DAKOTA/UQ 673
NonDLHSSampling
Performs LHS and Monte Carlo sampling for uncertainty quantification 674
NonDLHSSingleInterval
Class for pure interval propagation using LHS 676
NonDLocalEvidence
Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ 678
NonDLocalInterval
Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification 679
NonDLocalReliability
Class for the reliability methods within DAKOTA/UQ 681
NonDLocalSingleInterval
Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification 691
NonDPOFDarts
Base class for POF Dart methods within DAKOTA/UQ 692

NonDPolynomialChaos
Nonintrusive polynomial chaos expansion approaches to uncertainty quantification 695
NonDQuadrature
Derived nondeterministic class that generates N-dimensional numerical quadrature points for evaluation of expectation integrals over uncorrelated standard normals/uniforms/exponentials/betas/gammas 699
NonDQUESOBayesCalibration
Bayesian inference using the QUESO library from UT Austin 703
NonDReliability
Base class for the reliability methods within DAKOTA/UQ 705
NonDSampling
Base class for common code between NonDLHSSampling, NonDIncremLHSSampling, and NonDAdaptImpSampling 707
NonDSparseGrid
Derived nondeterministic class that generates N-dimensional Smolyak sparse grids for numerical evaluation of expectation integrals over independent standard random variables 714
NonDStochCollocation
Nonintrusive stochastic collocation approaches to uncertainty quantification 717
NonlinearCGOptimizer 719
NPSOLOptimizer
Wrapper class for the NPSOL optimization library 722
OptDartsOptimizer
Wrapper class for OptDarts Optimizer 724
Optimizer
Base class for the optimizer branch of the iterator hierarchy 727
OutputManager
Class to manage redirection of stdout/stderr, keep track of current redir state, and manage rank 0 output. Also manage tabular data output for post-processing with Matlab, Tecplot, etc. and delegate to Graphics for X Windows Graphics 732
OutputWriter 735
ParallelConfiguration
Container class for a set of ParallelLevel list iterators that collectively identify a particular multilevel parallel configuration 736
ParallelDirectApplicInterface
Sample derived interface class for testing parallel simulator plug-ins using assign_rep() 738
ParallelLevel
Container class for the data associated with a single level of communicator partitioning 739
ParallelLibrary
Class for partitioning multiple levels of parallelism and managing message passing within these levels 742
ParamResponsePair
Container class for a variables object, a response object, and an evaluation id 751
ParamStudy
Class for vector, list, centered, and multidimensional parameter studies 755
partial_prp_equality
Predicate for comparing ONLY the interfaceId and Vars attributes of PRPair 760
partial_prp_hash
Wrapper to delegate to the ParamResponsePair hash_value function 760

PecosApproximation
Derived approximation class for global basis polynomials 761
ProblemDescDB
The database containing information parsed from the DAKOTA input file 765
ProcessApplicInterface
Derived application interface class that spawns a simulation code using a separate process and communicates with it through files 775
ProcessHandleApplicInterface
Derived application interface class that spawns a simulation code using a separate process, receives a process identifier, and communicates with the spawned process through files 779
ProgramOptions
ProgramOptions stores options whether from the CLH or from library user; initially valid only on worldRank = 0, but then broadcast in ParallelLibrary::push_output_tag() 784
PStudyDACE
Base class for managing common aspects of parameter studies and design of experiments methods 788
PSUADEDesignCompExp
Wrapper class for the PSUADE library 790
PythonInterface 793
RecastModel
Derived model class which provides a thin wrapper around a sub-model in order to recast the form of its inputs and/or outputs 795
RelaxedVarConstraints
Derived class within the Constraints hierarchy which employs relaxation of discrete variables 804
RelaxedVariables
Derived class within the Variables hierarchy which employs the relaxation of discrete variables 805
Response
Container class for response functions and their derivatives. Response provides the handle class 806
ResponseRep
Container class for response functions and their derivatives. ResponseRep provides the body class 810
RestartWriter 817
ResultsDBAny 818
ResultsEntry< StoredType >
Class to manage in-core vs. file database lookups 820
ResultsID
Get a globally unique 1-based execution number for a given iterator name (combination of methodName and methodID) for use in results DB. Each Iterator::run() call creates or increments this count for its string identifier 821
ResultsManager
Results manager for iterator final data 822
ResultsNames
List of valid names for iterator results 824
RichExtrapVerification
Class for Richardson extrapolation for code and solution verification 825
ScilabInterface 827
SensAnalysisGlobal
Class for a utility class containing correlation calculations and variance-based decomposition 828

SeqHybridMetaIterator
Method for sequential hybrid iteration using multiple optimization and nonlinear least squares methods on multiple models of varying fidelity . . . 831
SerialDirectApplicInterface
Sample derived interface class for testing serial simulator plug-ins using assign_rep() . . . 834
SharedApproxData
Base class for the shared approximation data class hierarchy . . . 835
SharedPecosApproxData
Derived approximation class for global basis polynomials . . . 840
SharedSurfpackApproxData
Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota . . . 843
SharedVariablesData
Container class encapsulating variables data that can be shared among a set of Variables instances . . . 845
SharedVariablesDataRep
The representation of a SharedVariablesData instance. This representation, or body, may be shared by multiple SharedVariablesData handle instances . . . 850
SingleModel
Derived model class which utilizes a single interface to map variables into responses . . . 855
SNLLBase
Base class for OPT++ optimization and least squares methods . . . 857
SNLLLeastSq
Wrapper class for the OPT++ optimization library . . . 859
SNLLOptimizer
Wrapper class for the OPT++ optimization library . . . 863
SOLBase
Base class for Stanford SOL software . . . 869
SpawnApplicInterface
Derived application interface class which spawns simulation codes using spawnvp . . . 871
SurfpackApproximation
Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota . . . 872
SurrBasedGlobalMinimizer
The global surrogate-based minimizer which sequentially minimizes and updates a global surrogate model without trust region controls . . . 875
SurrBasedLocalMinimizer
Class for provably-convergent local surrogate-based optimization and nonlinear least squares . 876
SurrBasedMinimizer
Base class for local/global surrogate-based optimization/least squares . . . 883
SurrogateModel
Base class for surrogate models (DataFitSurrModel and HierarchSurrModel) . . . 888
SysCallApplicInterface
Derived application interface class which spawns simulation codes using system calls . . . 892
TANA3Approximation
Derived approximation class for TANA-3 two-point exponential approximation (a multipoint approximation) . . . 895
TaylorApproximation
Derived approximation class for first- or second-order Taylor series (a local approximation) . 897

TestDriverInterface 898
TrackerHTTP
TrackerHTTP: a usage tracking module that uses HTTP/HTTPS via the curl library 903
UsageTracker
Lightweight class to manage conditionally active Curl-based HTTP tracker via PIMPL 905
Var_icheck
Structure for verifying bounds and initial point for string-valued vars 906
Var_rcheck
Structure for verifying bounds and initial point for real-valued vars 906
Variables
Base class for the variables class hierarchy 907
Verification
Base class for managing common aspects of verification studies 916
VLint
Structure for validating integer uncertain variable labels, bounds, values 918
VLreal
Structure for validating real uncertain variable labels, bounds, values 918
VLstr
Structure for validating string uncertain variable labels, bounds, values 919
WorkdirHelper 919

Chapter 11

File Index

11.1 File List

Here is a list of all documented files with brief descriptions:

dakota_dll_api.cpp
This file contains a DakotaRunner class, which launches DAKOTA 925

dakota_dll_api.h
API for DLL interactions 926

dakota_tabular_io.hpp
Utility functions for reading and writing tabular data files Emerging utilities for tabular file I/O. For now, just extraction of capability from separate contexts to faciliate rework. These augment (and leverage) those in data_util.h 927

dll_tester.cpp
Test the DLL with a DAKOTA input file 929

JEGAOptimizer.cpp
Contains the implementation of the JEGAOptimizer class 929

JEGAOptimizer.hpp
Contains the definition of the JEGAOptimizer class 930

library_mode.cpp
File containing a mock simulator main for testing Dakota in library mode 930

library_split.cpp
File containing a mock simulator main for testing DAKOTA in library mode on a split communicator 934

main.cpp
File containing the main program for DAKOTA 934

restart_util.cpp
File containing the DAKOTA restart utility main program 935

Chapter 12

Namespace Documentation

12.1 Dakota Namespace Reference

The primary namespace for DAKOTA.

Classes

- class ApplicationInterface

  Derived class within the interface class hierarchy for supporting interfaces to simulation codes.
- class ApproximationInterface

  Derived class within the interface class hierarchy for supporting approximations to simulation-based results.
- class APPSEvalMgr

  Evaluation manager class for APPSPACK.
- class APPSOptimizer

  Wrapper class for HOPSPACK.
- class COLINApplication
- class COLINOptimizer

  Wrapper class for optimizers defined using COLIN.
- class CollabHybridMetaIterator

  Meta-iterator for hybrid iteration using multiple collaborating optimization and nonlinear least squares methods.
- class GetLongOpt

  GetLongOpt is a general command line utility from S. Manoharan (Advanced Computer Research Institute, Lyon, France).
- class CommandLineHandler

  Utility class for managing command line inputs to DAKOTA.
- class CommandShell

  Utility class which defines convenience operators for spawning processes with system calls.
- class ConcurrentMetaIterator

  Meta-iterator for multi-start iteration or pareto set optimization.
- class CONMINOptimizer

  Wrapper class for the CONMIN optimization library.
- struct BaseConstructor

Dummy struct for overloading letter-envelope constructors.

- struct NoDBBaseConstructor

  Dummy struct for overloading constructors used in on-the-fly instantiations without ProblemDescDB support.

- struct LightWtBaseConstructor

  Dummy struct for overloading constructors used in on-the-fly Model instantiations.

- class ActiveSet

  Container class for active set tracking information. Contains the active set request vector and the derivative variables vector.

- class Analyzer

  Base class for NonD, DACE, and ParamStudy branches of the iterator hierarchy.

- class Approximation

  Base class for the approximation class hierarchy.

- class Constraints

  Base class for the variable constraints class hierarchy.

- class Environment

  Base class for the environment class hierarchy.

- class Graphics

  The Graphics class provides a single interface to 2D (motif) and 3D (PLPLOT) graphics; there is only one instance of this OutputManager::dakotaGraphics.

- class Interface

  Base class for the interface class hierarchy.

- class Iterator

  Base class for the iterator class hierarchy.

- class LeastSq

  Base class for the nonlinear least squares branch of the iterator hierarchy.

- class Minimizer

  Base class for the optimizer and least squares branches of the iterator hierarchy.

- class Model

  Base class for the model class hierarchy.

- class NonD

  Base class for all nondetermistic iterators (the DAKOTA/UQ branch).

- class Optimizer

  Base class for the optimizer branch of the iterator hierarchy.

- class PStudyDACE

  Base class for managing common aspects of parameter studies and design of experiments methods.

- class ResponseRep

  Container class for response functions and their derivatives. ResponseRep provides the body class.

- class Response

  Container class for response functions and their derivatives. Response provides the handle class.

- class Variables

  Base class for the variables class hierarchy.

- class Verification

  Base class for managing common aspects of verification studies.

- class DataEnvironmentRep

  Body class for environment specification data.
- class DataEnvironment

  Handle class for environment specification data.
- class DataFitSurrModel

  Derived model class within the surrogate model branch for managing data fit surrogates (global and local)
- class DataInterface

  Handle class for interface specification data.
- class DataMethodRep

  Body class for method specification data.
- class DataMethod

  Handle class for method specification data.
- class DataModelRep

  Body class for model specification data.
- class DataModel

  Handle class for model specification data.
- class DataResponsesRep

  Body class for responses specification data.
- class DataResponses

  Handle class for responses specification data.
- class DataVariablesRep

  Body class for variables specification data.
- class DataVariables

  Handle class for variables specification data.
- class DDACEDesignCompExp

  Wrapper class for the DDACE design of experiments library.
- class DirectApplicInterface

  Derived application interface class which spawns simulation codes and testers using direct procedure calls.
- class DiscrepancyCorrection

  Base class for discrepancy corrections.
- class DOTOptimizer

  Wrapper class for the DOT optimization library.
- class EffGlobalMinimizer

  Implementation of Efficient Global Optimization/Least Squares algorithms.
- class EfficientSubspaceMethod

  Efficient Subspace Method (ESM), as proposed by Hany S. Abdel-Khalik.
- class EmbedHybridMetaIterator

  Meta-iterator for closely-coupled hybrid iteration, typically involving the embedding of local search methods within global search methods.
- class ExecutableEnvironment

  Environment corresponding to execution as a stand-alone application.
- class ExperimentData
- class ForkApplicInterface

Derived application interface class which spawns simulation codes using fork/execvp/waitpid.

- class FSUDesignCompExp

  Wrapper class for the FSUDace QMC/CVT library.
- class GaussProcApproximation

  Derived approximation class for Gaussian Process implementation.
- class GridApplicInterface

  Derived application interface class which spawns simulation codes using grid services such as Condor or Globus.
- class HierarchSurrModel

  Derived model class within the surrogate model branch for managing hierarchical surrogates (models of varying fidelity).
- class IteratorScheduler

  Environment corresponding to execution as a stand-alone application.
- class JEGAOptimizer

  A version of Dakota::Optimizer for instantiation of John Eddy's Genetic Algorithms (JEGA).
- class LibraryEnvironment

  Environment corresponding to execution as an embedded library.
- class MatlabInterface
- class MetaIterator

  Base class for meta-iterators.
- class MixedVarConstraints

  Derived class within the Constraints hierarchy which separates continuous and discrete variables (no domain type array merging).
- class MixedVariables

  Derived class within the Variables hierarchy which separates continuous and discrete variables (no domain type array merging).
- class MPIManager

  Class MPIManager to manage Dakota's MPI world, which may be a subset of MPI_COMM_WORLD.
- class MPIPackBuffer

  Class for packing MPI message buffers.
- class MPIUnpackBuffer

  Class for unpacking MPI message buffers.
- class NCSUOptimizer

  Wrapper class for the NCSU DIRECT optimization library.
- class NestedModel

  Derived model class which performs a complete sub-iterator execution within every evaluation of the model.
- struct Var_rcheck

  structure for verifying bounds and initial point for real-valued vars
- struct Var_icheck

  structure for verifying bounds and initial point for string-valued vars
- struct VLreal

  structure for validating real uncertain variable labels, bounds, values
- struct VLint

  structure for validating integer uncertain variable labels, bounds, values
- struct VLstr

structure for validating string uncertain variable labels, bounds, values

- class NIDRProblemDescDB

  The derived input file database utilizing the new IDR parser.
- struct NL2Res

  Auxiliary information passed to calcr and calcj via ur.
- class NL2SOLLeastSq

  Wrapper class for the NL2SOL nonlinear least squares library.
- class NLPQLPOptimizer

  Wrapper class for the NLPQLP optimization library, Version 2.0.
- class NLSSOLLeastSq

  Wrapper class for the NLSSOL nonlinear least squares library.
- class NomadOptimizer

  Wrapper class for NOMAD Optimizer.
- class NonDAdaptImpSampling

  Class for the Adaptive Importance Sampling methods within DAKOTA.
- class NonDAdaptiveSampling

  Class for testing various Adaptively sampling methods using geometric, statisctical, and topological information of the surrogate.
- class NonDBayesCalibration

  Base class for Bayesian inference: generates posterior distribution on model parameters given experimental data.
- class NonDCalibration
- class NonDCubature

  Derived nondeterministic class that generates N-dimensional numerical cubature points for evaluation of expectation integrals.
- class NonDDREAMBayesCalibration

  Bayesian inference using the DREAM approach.
- class NonDExpansion

  Base class for polynomial chaos expansions (PCE) and stochastic collocation (SC)
- class NonDGlobalEvidence

  Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.
- class NonDGlobalInterval

  Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.
- class NonDGlobalReliability

  Class for global reliability methods within DAKOTA/UQ.
- class NonDGlobalSingleInterval

  Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.
- class NonDGPImpSampling

  Class for the Gaussian Process-based Importance Sampling method.
- class NonDGPMSABayesCalibration

  Generates posterior distribution on model parameters given experiment data.
- class NonDIncremLHSSampling

  Performs icremental LHS sampling for uncertainty quantification.

- class NonDIntegration

  Derived nondeterministic class that generates N-dimensional numerical integration points for evaluation of expectation integrals.
- class NonDInterval

  Base class for interval-based methods within DAKOTA/UQ.
- class NonDLHSEvidence

  Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.
- class NonDLHSInterval

  Class for the LHS-based interval methods within DAKOTA/UQ.
- class NonDLHSSampling

  Performs LHS and Monte Carlo sampling for uncertainty quantification.
- class NonDLHSSingleInterval

  Class for pure interval propagation using LHS.
- class NonDLocalEvidence

  Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.
- class NonDLocalInterval

  Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.
- class NonDLocalReliability

  Class for the reliability methods within DAKOTA/UQ.
- class NonDLocalSingleInterval

  Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.
- class NonDPOFDarts

  Base class for POF Dart methods within DAKOTA/UQ.
- class NonDPolynomialChaos

  Nonintrusive polynomial chaos expansion approaches to uncertainty quantification.
- class NonDQuadrature

  Derived nondeterministic class that generates N-dimensional numerical quadrature points for evaluation of expectation integrals over uncorrelated standard normals/uniforms/exponentials/betas/gammas.
- class NonDQUESOBayesCalibration

  Bayesian inference using the QUESO library from UT Austin.
- class NonDReliability

  Base class for the reliability methods within DAKOTA/UQ.
- class NonDSampling

  Base class for common code between NonDLHSSampling, NonDIncremLHSSampling, and NonDAdaptImpSampling.
- class NonDSparseGrid

  Derived nondeterministic class that generates N-dimensional Smolyak sparse grids for numerical evaluation of expectation integrals over independent standard random variables.
- class NonDStochCollocation

  Nonintrusive stochastic collocation approaches to uncertainty quantification.
- class NonlinearCGOptimizer
- class NPSOLOptimizer

  Wrapper class for the NPSOL optimization library.

- class OptDartsOptimizer

  Wrapper class for OptDarts Optimizer.
- class OutputWriter
- class ConsoleRedirector
- class RestartWriter
- class OutputManager

  Class to manage redirection of stdout/stderr, keep track of current redir state, and manage rank 0 output. Also manage tabular data output for post-processing with Matlab, Tecplot, etc. and delegate to Graphics for X Windows Graphics.
- class ParallelLevel

  Container class for the data associated with a single level of communicator partitioning.
- class ParallelConfiguration

  Container class for a set of ParallelLevel list iterators that collectively identify a particular multilevel parallel configuration.
- class ParallelLibrary

  Class for partitioning multiple levels of parallelism and managing message passing within these levels.
- class ParamResponsePair

  Container class for a variables object, a response object, and an evaluation id.
- class ParamStudy

  Class for vector, list, centered, and multidimensional parameter studies.
- class PecosApproximation

  Derived approximation class for global basis polynomials.
- class ProblemDescDB

  The database containing information parsed from the DAKOTA input file.
- class ProcessApplicInterface

  Derived application interface class that spawns a simulation code using a separate process and communicates with it through files.
- class ProcessHandleApplicInterface

  Derived application interface class that spawns a simulation code using a separate process, receives a process identifier, and communicates with the spawned process through files.
- class ProgramOptions

  ProgramOptions stores options whether from the CLH or from library user; initially valid only on worldRank = 0, but then broadcast in ParallelLibrary::push_output_tag()
- struct partial_prp_hash

  wrapper to delegate to the ParamResponsePair hash_value function
- struct partial_prp_equality

  predicate for comparing ONLY the interfaceId and Vars attributes of PRPair
- class PSUADEDesignCompExp

  Wrapper class for the PSUADE library.
- class PythonInterface
- class RecastModel

  Derived model class which provides a thin wrapper around a sub-model in order to recast the form of its inputs and/or outputs.
- class RelaxedVarConstraints

  Derived class within the Constraints hierarchy which employs relaxation of discrete variables.

- class RelaxedVariables

  Derived class within the Variables hierarchy which employs the relaxation of discrete variables.
- class ResultsDBAny
- class ResultsID

  Get a globally unique 1-based execution number for a given iterator name (combination of methodName and methodID) for use in results DB. Each Iterator::run() call creates or increments this count for its string identifier.
- class ResultsNames

  List of valid names for iterator results.
- class ResultsManager

  Results manager for iterator final data.
- class ResultsEntry

  Class to manage in-core vs. file database lookups.
- class RichExtrapVerification

  Class for Richardson extrapolation for code and solution verification.
- class ScilabInterface
- class SensAnalysisGlobal

  Class for a utility class containing correlation calculations and variance-based decomposition.
- class SeqHybridMetaIterator

  Method for sequential hybrid iteration using multiple optimization and nonlinear least squares methods on multiple models of varying fidelity.
- class SharedApproxData

  Base class for the shared approximation data class hierarchy.
- class SharedPecosApproxData

  Derived approximation class for global basis polynomials.
- class SharedSurfpackApproxData

  Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota.
- class SharedVariablesDataRep

  The representation of a SharedVariablesData instance. This representation, or body, may be shared by multiple SharedVariablesData handle instances.
- class SharedVariablesData

  Container class encapsulating variables data that can be shared among a set of Variables instances.
- class SingleModel

  Derived model class which utilizes a single interface to map variables into responses.
- class SNLLBase

  Base class for OPT++ optimization and least squares methods.
- class SNLLLeastSq

  Wrapper class for the OPT++ optimization library.
- class SNLLOptimizer

  Wrapper class for the OPT++ optimization library.
- class SOLBase

  Base class for Stanford SOL software.
- class SpawnApplicInterface

  Derived application interface class which spawns simulation codes using spawnvp.
- class SurfpackApproximation

Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota.

- class SurrBasedGlobalMinimizer

  The global surrogate-based minimizer which sequentially minimizes and updates a global surrogate model without trust region controls.

- class SurrBasedLocalMinimizer

  Class for provably-convergent local surrogate-based optimization and nonlinear least squares.

- class SurrBasedMinimizer

  Base class for local/global surrogate-based optimization/least squares.

- class SurrogateModel

  Base class for surrogate models (DataFitSurrModel and HierarchSurrModel).

- class SysCallApplicInterface

  Derived application interface class which spawns simulation codes using system calls.

- class TANA3Approximation

  Derived approximation class for TANA-3 two-point exponential approximation (a multipoint approximation).

- class TaylorApproximation

  Derived approximation class for first- or second-order Taylor series (a local approximation).

- class TestDriverInterface
- class TrackerHTTP

  TrackerHTTP: a usage tracking module that uses HTTP/HTTPS via the curl library.

- class UsageTracker

  Lightweight class to manage conditionally active Curl-based HTTP tracker via PIMPL.

- struct MatchesWC

  Predicate that returns true when the passed path matches the wild_card with which it was configured. Currently supports $*$ and ?.

- class WorkdirHelper

Typedefs

- typedef double **Real**
- typedef std::string **String**
- typedef Teuchos::SerialDenseVector $<$ int, Real $>$ **RealVector**
- typedef Teuchos::SerialDenseMatrix $<$ int, Real $>$ **RealMatrix**
- typedef Teuchos::SerialSymDenseMatrix $<$ int, Real $>$ **RealSymMatrix**
- typedef Teuchos::SerialDenseVector $<$ int, int $>$ **IntVector**
- typedef Teuchos::SerialDenseMatrix $<$ int, int $>$ **IntMatrix**
- typedef std::deque$<$ bool $>$ **BoolDeque**

- typedef boost::dynamic_bitset
  < unsigned long > **BitArray**
- typedef std::vector< BoolDeque > **BoolDequeArray**
- typedef std::vector< Real > **RealArray**
- typedef std::vector< RealArray > **Real2DArray**
- typedef std::vector< int > **IntArray**
- typedef std::vector< IntArray > **Int2DArray**
- typedef std::vector< short > **ShortArray**
- typedef std::vector< unsigned
  short > **UShortArray**
- typedef std::vector< UShortArray > **UShort2DArray**
- typedef std::vector
  < UShort2DArray > **UShort3DArray**
- typedef std::vector< size_t > **SizetArray**
- typedef std::vector< SizetArray > **Sizet2DArray**
- typedef std::vector< String > **StringArray**
- typedef std::vector< StringArray > **String2DArray**
- typedef
  boost::multi_array_types::index_range **idx_range**
- typedef boost::multi_array
  < String, 1 > **StringMultiArray**
- typedef boost::multi_array
  < String, 2 > **StringMulti2DArray**
- typedef
  StringMultiArray::array_view
  < 1 >::type **StringMultiArrayView**
- typedef
  StringMultiArray::const_array_view
  < 1 >::type **StringMultiArrayConstView**
- typedef boost::multi_array
  < unsigned short, 1 > **UShortMultiArray**
- typedef
  UShortMultiArray::array_view
  < 1 >::type **UShortMultiArrayView**
- typedef
  UShortMultiArray::const_array_view
  < 1 >::type **UShortMultiArrayConstView**
- typedef boost::multi_array
  < size_t, 1 > **SizetMultiArray**
- typedef
  SizetMultiArray::array_view
  < 1 >::type **SizetMultiArrayView**
- typedef
  SizetMultiArray::const_array_view
  < 1 >::type **SizetMultiArrayConstView**
- typedef boost::multi_array
  < Real, 1 > **RealMultiArray**

- typedef boost::multi_array
  < Real, 2 > **RealMulti2DArray**
- typedef boost::multi_array
  < Real, 3 > **RealMulti3DArray**
- typedef std::vector< RealVector > **RealVectorArray**
- typedef std::vector
  < RealVectorArray > **RealVector2DArray**
- typedef std::vector< RealMatrix > **RealMatrixArray**
- typedef std::vector
  < RealSymMatrix > **RealSymMatrixArray**
- typedef std::vector< IntVector > **IntVectorArray**
- typedef std::vector< Variables > **VariablesArray**
- typedef std::vector< Response > **ResponseArray**
- typedef std::vector
  < ParamResponsePair > **PRPArray**
- typedef std::vector< PRPArray > **PRP2DArray**
- typedef std::vector< Model > **ModelArray**
- typedef std::vector< Iterator > **IteratorArray**
- typedef std::vector
  < RealMultiArray > **BoostMAArray**
- typedef std::vector
  < RealMulti2DArray > **BoostMA2DArray**
- typedef std::vector
  < RealMulti3DArray > **BoostMA3DArray**
- typedef std::list< bool > **BoolList**
- typedef std::list< int > **IntList**
- typedef std::list< size_t > **SizetList**
- typedef std::list< Real > **RealList**
- typedef std::list< RealVector > **RealVectorList**
- typedef std::list< String > **StringList**
- typedef std::list< Variables > **VariablesList**
- typedef std::list< Interface > **InterfaceList**
- typedef std::list< Response > **ResponseList**
- typedef std::list< Model > **ModelList**
- typedef std::list< Iterator > **IteratorList**
- typedef std::pair< int, int > **IntIntPair**
- typedef std::pair< size_t, int > **SizetIntPair**
- typedef std::pair< int, String > **IntStringPair**
- typedef std::pair< Real, Real > **RealRealPair**
- typedef std::pair< int, Response > **IntResponsePair**
- typedef std::set< Real > **RealSet**
- typedef std::set< int > **IntSet**
- typedef std::set< String > **StringSet**
- typedef std::set< unsigned short > **UShortSet**
- typedef std::set< size_t > **SizetSet**
- typedef std::vector< RealSet > **RealSetArray**
- typedef std::vector< IntSet > **IntSetArray**

- typedef std::vector< StringSet > **StringSetArray**
- typedef std::vector< UShortSet > **UShortSetArray**
- typedef std::map< int, int > **IntIntMap**
- typedef std::map< int, short > **IntShortMap**
- typedef std::map< int, short > **IntSizetMap**
- typedef std::map< int, Real > **IntRealMap**
- typedef std::map< Real, Real > **RealRealMap**
- typedef std::map< String, Real > **StringRealMap**
- typedef std::vector< IntRealMap > **IntRealMapArray**
- typedef std::vector< RealRealMap > **RealRealMapArray**
- typedef std::vector
  < StringRealMap > **StringRealMapArray**
- typedef std::map< int, RealVector > **IntRealVectorMap**
- typedef std::map< int, ActiveSet > **IntActiveSetMap**
- typedef std::map< int, Variables > **IntVariablesMap**
- typedef std::map< int, Response > **IntResponseMap**
- typedef std::map< IntArray,
  size_t > **IntArraySizetMap**
- typedef std::map< IntIntPair,
  Real > **IntIntPairRealMap**
- typedef std::map< RealRealPair,
  Real > **RealRealPairRealMap**
- typedef std::vector
  < IntIntPairRealMap > **IntIntPairRealMapArray**
- typedef std::vector
  < RealRealPairRealMap > **RealRealPairRealMapArray**
- typedef std::multimap
  < RealRealPair,
  ParamResponsePair > **RealPairPRPMultiMap**
- typedef IntList::iterator **ILIter**
- typedef IntList::const_iterator **ILCIter**
- typedef SizetList::iterator **StLIter**
- typedef SizetList::const_iterator **StLCIter**
- typedef RealList::iterator **RLIter**
- typedef RealList::const_iterator **RLCIter**
- typedef RealVectorList::iterator **RVLIter**
- typedef
  RealVectorList::const_iterator **RVLCIter**
- typedef StringList::iterator **StringLIter**
- typedef StringList::const_iterator **StringLCIter**
- typedef VariablesList::iterator **VarsLIter**
- typedef InterfaceList::iterator **InterfLIter**
- typedef ResponseList::iterator **RespLIter**
- typedef ModelList::iterator **ModelLIter**
- typedef ModelList::reverse_iterator **ModelLRevIter**
- typedef IteratorList::iterator **IterLIter**

- typedef std::list
  < ParallelLevel >::iterator **ParLevLIter**
- typedef std::list
  < ParallelConfiguration >
  ::iterator **ParConfigLIter**
- typedef IntSet::iterator **ISIter**
- typedef IntSet::const_iterator **ISCIter**
- typedef StringSet::iterator **SSIter**
- typedef StringSet::const_iterator **SSCIter**
- typedef RealSet::iterator **RSIter**
- typedef RealSet::const_iterator **RSCIter**
- typedef IntIntMap::iterator **IntIntMIter**
- typedef IntIntMap::const_iterator **IntIntMCIter**
- typedef IntShortMap::iterator **IntShMIter**
- typedef IntShortMap::const_iterator **IntShMCIter**
- typedef IntRealMap::iterator **IRMIter**
- typedef IntRealMap::const_iterator **IRMCIter**
- typedef StringRealMap::iterator **SRMIter**
- typedef
  StringRealMap::const_iterator **SRMCIter**
- typedef RealRealMap::iterator **RRMIter**
- typedef RealRealMap::const_iterator **RRMCIter**
- typedef IntIntPairRealMap::iterator **IIPRMIter**
- typedef
  IntIntPairRealMap::const_iterator **IIPRMCIter**
- typedef
  RealRealPairRealMap::iterator **RRPRMIter**
- typedef
  RealRealPairRealMap::const_iterator **RRPRMCIter**
- typedef IntRealVectorMap::iterator **IntRDVMIter**
- typedef
  IntRealVectorMap::const_iterator **IntRDVMCIter**
- typedef IntActiveSetMap::iterator **IntASMIter**
- typedef IntVariablesMap::iterator **IntVarsMIter**
- typedef
  IntVariablesMap::const_iterator **IntVarsMCIter**
- typedef IntResponseMap::iterator **IntRespMIter**
- typedef
  IntResponseMap::const_iterator **IntRespMCIter**
- typedef boost::tuple
  < std::string, std::string,
  size_t, std::string > ResultsKeyType

  Data type for results key (instance name / id, unique run, label), where data_key is a valid colon-delimited string from ResultsNames tuple<method_name, method_id, execution_number, data_key>
- typedef std::string MetaDataKeyType

  Data type for metadata key.
- typedef std::vector< std::string > MetaDataValueType

Data type for metadata value.

- typedef std::map < MetaDataKeyType, MetaDataValueType > MetaDataType

  A single MetaData entry is map<string, vector<string> > Example: pair("Column labels", ["Mean", "Std Dev", "Skewness", "Kurtosis"])

- typedef boost::tuple < std::string, std::string, size_t > StrStrSizet

  Iterator unique ID: <method_name, method_id, exec_num>

- typedef void(∗ **dl_find_optimum_t**)(void ∗, Optimizer1 ∗, char ∗)
- typedef void(∗ **dl_destructor_t**)(void ∗∗)
- typedef Teuchos::SerialDenseSolver < int, Real > **RealSolver**
- typedef Teuchos::SerialSpdDenseSolver < int, Real > **RealSpdSolver**
- typedef int(∗ start_grid_computing_t)(char ∗analysis_driver_script, char ∗params_file, char ∗results_file)

  definition of start grid computing type (function pointer)

- typedef int(∗ perform_analysis_t)(char ∗iteration_num)

  definition of perform analysis type (function pointer)

- typedef int ∗(∗ get_jobs_completed_t)()

  definition of get completed jobs type (function pointer)

- typedef int(∗ stop_grid_computing_t)()

  definition of stop grid computing type (function pointer)

- typedef int **MPI_Comm**
- typedef void ∗ **MPI_Request**
- typedef unsigned char **u_char**
- typedef unsigned short **u_short**
- typedef unsigned int **u_int**
- typedef unsigned long **u_long**
- typedef long long **long_long**
- typedef unsigned long **UL**
- typedef void(∗ **Calcrj**)(int ∗n, int ∗p, Real ∗x, int ∗nf, Real ∗r, int ∗ui, void ∗ur, Vf vf)
- typedef void(∗ **Vf**)()
- typedef void(∗ **DbCallbackFunctionPtr**)(Dakota::ProblemDescDB ∗db, void ∗data_ptr)
- typedef boost::tuple < bfs::path, bfs::path, bfs::path > PathTriple

  Triplet of filesystem paths: e.g., params, results, workdir.

- typedef
  bmi::multi_index_container
  < Dakota::ParamResponsePair,
  bmi::indexed_by
  < bmi::ordered_non_unique
  < bmi::tag< ordered >
  , bmi::const_mem_fun
  < Dakota::ParamResponsePair,
  const IntStringPair
  &,&Dakota::ParamResponsePair::eval_interface_ids >
  >, bmi::hashed_non_unique
  < bmi::tag< hashed >
  , bmi::identity
  < Dakota::ParamResponsePair >
  , partial_prp_hash,
  partial_prp_equality > > > PRPMultiIndexCache

  Boost Multi-Index Container for globally caching ParamResponsePairs.
- typedef PRPMultiIndexCache **PRPCache**
- typedef
  PRPCache::index_iterator
  < ordered >::type **PRPCacheOIter**
- typedef
  PRPCache::index_const_iterator
  < ordered >::type **PRPCacheOCIter**
- typedef
  PRPCache::index_iterator
  < hashed >::type **PRPCacheHIter**
- typedef
  PRPCache::index_const_iterator
  < hashed >::type **PRPCacheHCIter**
- typedef PRPCacheOIter **PRPCacheIter**
- typedef PRPCacheOCIter **PRPCacheCIter**
- typedef
  bmi::multi_index_container
  < Dakota::ParamResponsePair,
  bmi::indexed_by
  < bmi::ordered_unique
  < bmi::tag< ordered >
  , bmi::const_mem_fun
  < Dakota::ParamResponsePair,
  int,&Dakota::ParamResponsePair::eval_id >
  >, bmi::hashed_non_unique
  < bmi::tag< hashed >
  , bmi::identity
  < Dakota::ParamResponsePair >
  , partial_prp_hash,
  partial_prp_equality > > > PRPMultiIndexQueue

  Boost Multi-Index Container for locally queueing ParamResponsePairs.

- typedef PRPMultiIndexQueue **PRPQueue**
- typedef PRPQueue::index_iterator < ordered >::type **PRPQueueOIter**
- typedef PRPQueue::index_const_iterator < ordered >::type **PRPQueueOCIter**
- typedef PRPQueue::index_iterator < hashed >::type **PRPQueueHIter**
- typedef PRPQueue::index_const_iterator < hashed >::type **PRPQueueHCIter**
- typedef PRPQueueOIter **PRPQueueIter**
- typedef PRPQueueOCIter **PRPQueueCIter**
- typedef std::pair< boost::any, MetaDataType > ResultsValueType

  Core data storage type: boost::any, with optional metadata (see other types in results_types.hpp)
- typedef boost::function< bool(const bfs::path &src_path, const bfs::path &dest_path, bool overwrite)> file_op_function

  define a function type that operates from src to dest, with option to overwrite
- typedef boost::filter_iterator < MatchesWC, bfs::directory_iterator > glob_iterator

  a glob_iterator filters a directory_iterator based on a wildcard predicate

Enumerations

- enum { **COBYLA**, **DIRECT**, **EA**, **MS**, **PS**, **SW**, **BETA** }
- enum { **MODEL_ERROR** = -6, **IO_ERROR** = -5, **INTERFACE_ERROR** = -4, **CONSTRUCT_ERROR** = -3, **PARSE_ERROR** = -2, **OTHER_ERROR** = -1 }

  enum for Dakota abort reasons; using negative numbers to avoid clash with signal codes 1–64 in signum.h
- enum { **ABORT_EXITS**, **ABORT_THROWS** }

  enum for dakota abort behaviors
- enum { **DEFAULT_INTERFACE** =0, **APPROX_INTERFACE**, **FORK_INTERFACE** =PROCESS_INTERFACE_BIT, **SYSTEM_INTERFACE**, **GRID_INTERFACE**, **TEST_INTERFACE** =DIRECT_INTERFACE_BIT, **MATLAB_INTERFACE**, **PYTHON_INTERFACE**, **SCILAB_INTERFACE** }

  special values for interface type
- enum { **SYNCHRONOUS_INTERFACE**, **ASYNCHRONOUS_INTERFACE** }

interface synchronization types

- enum { **OBJECTIVE**, **INEQUALITY_CONSTRAINT**, **EQUALITY_CONSTRAINT** }

  define algebraic function types

- enum {
  DEFAULT_METHOD =0, **HYBRID** =(META_BIT | PARALLEL_BIT), **PARETO_SET**, **MULTI_START**,
  BRANCH_AND_BOUND, **RICHARDSON_EXTRAP** =(ANALYZER_BIT | VERIF_BIT), **CENTERED_PARAMETER_STUDY** =(ANALYZER_BIT | PSTUDYDACE_BIT), **LIST_PARAMETER_STUDY**,
  MULTIDIM_PARAMETER_STUDY, **VECTOR_PARAMETER_STUDY**, **DACE**, **FSU_CVT**,
  FSU_HALTON, **FSU_HAMMERSLEY**, **PSUADE_MOAT**, **LOCAL_RELIABILITY** =(ANALYZER_BIT | NOND_BIT),
  GLOBAL_RELIABILITY, **POLYNOMIAL_CHAOS**, **STOCH_COLLOCATION**, **CUBATURE_INTEGRATION**,
  SPARSE_GRID_INTEGRATION, **QUADRATURE_INTEGRATION**, **BAYES_CALIBRATION**, **GPAIS**,
  POF_DARTS, **EFFICIENT_SUBSPACE**, **IMPORTANCE_SAMPLING**, **ADAPTIVE_SAMPLING**,
  RANDOM_SAMPLING, **LOCAL_INTERVAL_EST**, **LOCAL_EVIDENCE**, **GLOBAL_INTERVAL_EST**,
  GLOBAL_EVIDENCE, **SURROGATE_BASED_LOCAL** =(MINIMIZER_BIT | SURRBASED_BIT),
  SURROGATE_BASED_GLOBAL, **EFFICIENT_GLOBAL**,
  NL2SOL =(MINIMIZER_BIT | LEASTSQ_BIT), **NLSSOL_SQP**, **OPTPP_G_NEWTON**, **ASYNCH_PATTERN_SEARCH** =(MINIMIZER_BIT | OPTIMIZER_BIT),
  OPTPP_PDS, **COLINY_BETA**, **COLINY_COBYLA**, **COLINY_DIRECT**,
  COLINY_MULTI_START, **COLINY_EA**, **COLINY_PATTERN_SEARCH**, **COLINY_SOLIS_WETS**,
  MOGA, **SOGA**, **NCSU_DIRECT**, **MESH_ADAPTIVE_SEARCH**,
  GENIE_OPT_DARTS, **GENIE_DIRECT**, **NONLINEAR_CG**, **OPTPP_CG**,
  OPTPP_Q_NEWTON, **OPTPP_FD_NEWTON**, **OPTPP_NEWTON**, **NPSOL_SQP**,
  NLPQL_SQP, **DOT_BFGS**, **DOT_FRCG**, **DOT_MMFD**,
  DOT_SLP, **DOT_SQP**, **CONMIN_FRCG**, **CONMIN_MFD**,
  DL_SOLVER }
- enum {
  SUBMETHOD_DEFAULT =0, SUBMETHOD_COLLABORATIVE, **SUBMETHOD_EMBEDDED**, **SUBMETHOD_SEQUENTIAL**,
  SUBMETHOD_LHS, **SUBMETHOD_RANDOM**, **SUBMETHOD_INCREMENTAL_LHS**, **SUBMETHOD_INCREMENTAL_RANDOM**,
  SUBMETHOD_BOX_BEHNKEN, **SUBMETHOD_CENTRAL_COMPOSITE**, **SUBMETHOD_GRID**, **SUBMETHOD_OA_LHS**,
  SUBMETHOD_OAS, **SUBMETHOD_DREAM**, **SUBMETHOD_GPMSA**, **SUBMETHOD_QUESO**,
  SUBMETHOD_NIP, **SUBMETHOD_SQP**, **SUBMETHOD_EA**, **SUBMETHOD_EGO**,
  SUBMETHOD_SBO, **SUBMETHOD_CONVERGE_ORDER**, **SUBMETHOD_CONVERGE_QOI**, **SUBMETHOD_ESTIMATE_ORDER** }

  Sub-methods, including sampling, inference algorithm, opt algorithm types.

- enum {
  SILENT_OUTPUT, **QUIET_OUTPUT**, **NORMAL_OUTPUT**, **VERBOSE_OUTPUT**,
  DEBUG_OUTPUT }
- enum {
  DEFAULT_SCHEDULING, **MASTER_SCHEDULING**, **PEER_SCHEDULING**, **PEER_DYNAMIC-**

_SCHEDULING,
PEER_STATIC_SCHEDULING, **DYNAMIC_SCHEDULING**, **STATIC_SCHEDULING** }

- enum { **DEFAULT_CONFIG**, **PUSH_DOWN**, **PUSH_UP** }
- enum { **STD_NORMAL_U**, **STD_UNIFORM_U**, **ASKEY_U**, **EXTENDED_U** }
- enum { **DEFAULT_COVARIANCE**, **NO_COVARIANCE**, **DIAGONAL_COVARIANCE**, **FULL_COVARIANCE** }
- enum { **NO_INT_REFINE** =0, **IS**, **AIS**, **MMAIS** }
- enum { **PROBABILITIES**, **RELIABILITIES**, **GEN_RELIABILITIES** }
- enum { **COMPONENT** =0, **SYSTEM_SERIES**, **SYSTEM_PARALLEL** }
- enum { **CUMULATIVE**, **COMPLEMENTARY** }
- enum { **DEFAULT_LS** =0, **SVD_LS**, **EQ_CON_LS** }
- enum {
  NO_EMULATOR, **PCE_EMULATOR**, **SC_EMULATOR**, **GP_EMULATOR**,
  KRIGING_EMULATOR, **VPS_EMULATOR** }
- enum { **IGNORE_RANKS**, **SET_RANKS**, **GET_RANKS**, **SET_GET_RANKS** }
- enum {
  UNCERTAIN, **UNCERTAIN_UNIFORM**, **ALEATORY_UNCERTAIN**, **ALEATORY_UNCERTAIN_UNIFORM**,
  EPISTEMIC_UNCERTAIN, **EPISTEMIC_UNCERTAIN_UNIFORM**, **ACTIVE**, **ACTIVE_UNIFORM**,
  ALL, **ALL_UNIFORM** }
- enum {
  MV =0, **AMV_X**, **AMV_U**, **AMV_PLUS_X**,
  AMV_PLUS_U, **TANA_X**, **TANA_U**, **NO_APPROX**,
  EGRA_X, **EGRA_U** }
- enum { **BREITUNG**, **HOHENRACK**, **HONG** }
- enum { **ORIGINAL_PRIMARY**, **SINGLE_OBJECTIVE**, **LAGRANGIAN_OBJECTIVE**, **AUGMENTED_LAGRANGIAN_OBJECTIVE** }
- enum { **NO_CONSTRAINTS**, **LINEARIZED_CONSTRAINTS**, **ORIGINAL_CONSTRAINTS** }
- enum { **NO_RELAX**, **HOMOTOPY**, **COMPOSITE_STEP** }
- enum { **PENALTY_MERIT**, **ADAPTIVE_PENALTY_MERIT**, **LAGRANGIAN_MERIT**, **AUGMENTED_LAGRANGIAN_MERIT** }
- enum { **FILTER**, **TR_RATIO** }
- enum { **SCALE_NONE**, **SCALE_VALUE**, **SCALE_LOG** }
- enum { **CDV**, **LINEAR**, **NONLIN**, **FN_LSQ** }
- enum { **DISALLOW**, **TARGET**, **BOUNDS** }
- enum { **DEFAULT_POINTS**, **MINIMUM_POINTS**, **RECOMMENDED_POINTS**, **TOTAL_POINTS** }

  define special values for pointsManagement
- enum {
  NO_SURROGATE =0, **UNCORRECTED_SURROGATE**, **AUTO_CORRECTED_SURROGATE**, **BYPASS_SURROGATE**,
  MODEL_DISCREPANCY }

  define special values for SurrogateModel::responseMode
- enum { **NO_CORRECTION** =0, **ADDITIVE_CORRECTION**, **MULTIPLICATIVE_CORRECTION**, **COMBINED_CORRECTION** }

  define special values for approxCorrectionType

- enum { **DEFAULT_DOMAIN** =0, **RELAXED_DOMAIN**, **MIXED_DOMAIN** }
- enum {
  DEFAULT_VIEW =0, **ALL_VIEW**, **DESIGN_VIEW**, **UNCERTAIN_VIEW**,
  ALEATORY_UNCERTAIN_VIEW, **EPISTEMIC_UNCERTAIN_VIEW**, **STATE_VIEW** }
- enum {
  EMPTY =0, **RELAXED_ALL**, **MIXED_ALL**, **RELAXED_DESIGN**,
  RELAXED_UNCERTAIN, **RELAXED_ALEATORY_UNCERTAIN**, **RELAXED_EPISTEMIC_UNCERTAIN**, **RELAXED_STATE**,
  MIXED_DESIGN, **MIXED_UNCERTAIN**, **MIXED_ALEATORY_UNCERTAIN**, **MIXED_EPISTEMIC_UNCERTAIN**,
  MIXED_STATE }
- enum {
  CONTINUOUS_DESIGN =1, **DISCRETE_DESIGN_RANGE**, **DISCRETE_DESIGN_SET_INT**, **DISCRETE_DESIGN_SET_STRING**,
  DISCRETE_DESIGN_SET_REAL, **NORMAL_UNCERTAIN**, **LOGNORMAL_UNCERTAIN**, **UNIFORM_UNCERTAIN**,
  LOGUNIFORM_UNCERTAIN, **TRIANGULAR_UNCERTAIN**, **EXPONENTIAL_UNCERTAIN**, **BETA_UNCERTAIN**,
  GAMMA_UNCERTAIN, **GUMBEL_UNCERTAIN**, **FRECHET_UNCERTAIN**, **WEIBULL_UNCERTAIN**,
  HISTOGRAM_BIN_UNCERTAIN, **POISSON_UNCERTAIN**, **BINOMIAL_UNCERTAIN**, **NEGATIVE_BINOMIAL_UNCERTAIN**,
  GEOMETRIC_UNCERTAIN, **HYPERGEOMETRIC_UNCERTAIN**, **HISTOGRAM_POINT_UNCERTAIN_INT**, **HISTOGRAM_POINT_UNCERTAIN_STRING**,
  HISTOGRAM_POINT_UNCERTAIN_REAL, **CONTINUOUS_INTERVAL_UNCERTAIN**, **DISCRETE_INTERVAL_UNCERTAIN**, **DISCRETE_UNCERTAIN_SET_INT**,
  DISCRETE_UNCERTAIN_SET_STRING, **DISCRETE_UNCERTAIN_SET_REAL**, **CONTINUOUS_STATE**, **DISCRETE_STATE_RANGE**,
  DISCRETE_STATE_SET_INT, **DISCRETE_STATE_SET_STRING**, **DISCRETE_STATE_SET_REAL** }
- enum {
  TOTAL_CDV =0, **TOTAL_DDIV**, **TOTAL_DDSV**, **TOTAL_DDRV**,
  TOTAL_CAUV, **TOTAL_DAUIV**, **TOTAL_DAUSV**, **TOTAL_DAURV**,
  TOTAL_CEUV, **TOTAL_DEUIV**, **TOTAL_DEUSV**, **TOTAL_DEURV**,
  TOTAL_CSV, **TOTAL_DSIV**, **TOTAL_DSSV**, **TOTAL_DSRV**,
  NUM_VC_TOTALS }
- enum var_t {
  VAR_x1, **VAR_x2**, **VAR_x3**, **VAR_b**,
  VAR_h, **VAR_P**, **VAR_M**, **VAR_Y**,
  VAR_w, **VAR_t**, **VAR_R**, **VAR_E**,
  VAR_X, **VAR_Fs**, **VAR_P1**, **VAR_P2**,
  VAR_P3, **VAR_B**, **VAR_D**, **VAR_H**,
  VAR_F0, **VAR_d**, **VAR_MForm** }

  enumeration of possible variable types (to index to names)
- enum driver_t {
  NO_DRIVER =0, **CANTILEVER_BEAM**, **MOD_CANTILEVER_BEAM**, **CYLINDER_HEAD**,
  EXTENDED_ROSENBROCK, **GENERALIZED_ROSENBROCK**, **LF_ROSENBROCK**, **MF_ROS-**

ENBROCK,
ROSENBROCK, **GERSTNER**, **SCALABLE_GERSTNER**, **LOGNORMAL_RATIO**, **MULTIMODAL**, **PLUGIN_ROSENBROCK**, **PLUGIN_TEXT_BOOK**, **SHORT_COLUMN**, **LF_SHORT_COLUMN**, **MF_SHORT_COLUMN**, **SIDE_IMPACT_COST**, **SIDE_IMPACT_PERFORMANCE**,
SOBOL_RATIONAL, **SOBOL_G_FUNCTION**, **SOBOL_ISHIGAMI**, **STEEL_COLUMN_COST**, **STEEL_COLUMN_PERFORMANCE**, **TEXT_BOOK**, **TEXT_BOOK1**, **TEXT_BOOK2**, **TEXT_BOOK3**, **TEXT_BOOK_OUU**, **SCALABLE_TEXT_BOOK**, **SCALABLE_MONOMIALS**, **HERBIE**, **SMOOTH_HERBIE**, **SHUBERT**, **SALINAS**,
MODELCENTER, **GENZ** }

enumeration of possible direct driver types (to index to names)

- enum local_data_t { **VARIABLES_MAP** =1, **VARIABLES_VECTOR** =2 }

  enumeration for how local variables are stored (values must employ a bit representation)
- enum sigtype { **NO_SIGMA**, **SCALAR_SIGMA**, **COVARIANCE_MATRIX** }

  special values for sigma_type
- enum edtype { **SCALAR_DATA**, **FUNCTIONAL_DATA** }

  special values for experimental data type
- enum { **SETUP_MODEL**, **SETUP_USERFUNC** }
- enum {
  CAUVar_normal = 0, **CAUVar_lognormal** = 1, **CAUVar_uniform** = 2, **CAUVar_loguniform** = 3, **CAUVar_triangular** = 4, **CAUVar_exponential** = 5, **CAUVar_beta** = 6, **CAUVar_gamma** = 7, **CAUVar_gumbel** = 8, **CAUVar_frechet** = 9, **CAUVar_weibull** = 10, **CAUVar_histogram_bin** = 11, **CAUVar_Nkinds** = 12 }
- enum {
  DAUIVar_poisson = 0, **DAUIVar_binomial** = 1, **DAUIVar_negative_binomial** = 2, **DAUIVar_geometric** = 3,
  DAUIVar_hypergeometric = 4, **DAUIVar_histogram_point_int** = 5, **DAUIVar_Nkinds** = 6 }
- enum { **DAUSVar_histogram_point_str** = 0, **DAUSVar_Nkinds** = 1 }
- enum { **DAURVar_histogram_point_real** = 0, **DAURVar_Nkinds** = 1 }
- enum { **CEUVar_interval** = 0, **CEUVar_Nkinds** = 1 }
- enum { **DEUIVar_interval** = 0, **DEUIVar_set_int** = 1, **DEUIVar_Nkinds** = 2 }
- enum { **DEUSVar_set_str** = 0, **DEUSVar_Nkinds** = 1 }
- enum { **DEURVar_set_real** = 0, **DEURVar_Nkinds** = 1 }
- enum {
  DiscSetVar_design_set_int = 0, **DiscSetVar_design_set_str** = 1, **DiscSetVar_design_set_real** = 2, **DiscSetVar_state_set_int** = 3,
  DiscSetVar_state_set_str = 4, **DiscSetVar_state_set_real** = 5, **DiscSetVar_Nkinds** = 6 }
- enum { **NUM_UNC_REAL_CONT** = 4 }

  number of real-valued uncertain contiguous containers
- enum { **NUM_UNC_INT_CONT** = 2 }

  number of int-valued uncertain contiguous containers
- enum { **NUM_UNC_STR_CONT** = 2 }

  number of string-valued uncertain contiguous containers
- enum { **FULL_TENSOR**, **FILTERED_TENSOR**, **RANDOM_TENSOR** }
- enum CG_UPDATETYPE {
  CG_STEEPEST, **CG_FLETCHER_REEVES**, **CG_POLAK_RIBIERE**, **CG_POLAK_RIBIERE_PLUS**,
  CG_HESTENES_STIEFEL }

NonlinearCG update options.

- enum CG_LINESEARCHTYPE { **CG_FIXED_STEP**, **CG_LS_SIMPLE**, **CG_LS_BRENT**, **CG_LS_WOLFE** }

  NonlinearCG linesearch options.
- enum EvalType { **NLFEvaluator**, **CONEvaluator** }

  enumeration for the type of evaluator function
- enum {
  TH_SILENT_OUTPUT, **TH_QUIET_OUTPUT**, **TH_NORMAL_OUTPUT**, **TH_VERBOSE_OUTPUT**,
  TH_DEBUG_OUTPUT }
- enum { **DIR_CLEAN**, **DIR_PERSIST**, **DIR_ERROR** }

  define directory creation options
- enum { **FILEOP_SILENT**, **FILEOP_WARN**, **FILEOP_ERROR** }

  enum indicating action on failed file operation

Functions

- CommandShell & flush (CommandShell &shell)

  convenient shell manipulator function to "flush" the shell
- bool nearby (const RealVector &rv1, const RealVector &rv2, Real rel_tol)

  tolerance-based equality operator for RealVector
- bool operator== (const ShortArray &dsa1, const ShortArray &dsa2)

  equality operator for ShortArray
- bool operator== (const StringArray &dsa1, const StringArray &dsa2)

  equality operator for StringArray
- bool operator== (const SizetArray &sa, SizetMultiArrayConstView smav)

  equality operator for SizetArray and SizetMultiArrayConstView
- bool operator!= (const StringMultiArrayView &sma1, const StringMultiArray &sma2)

  inequality operator for StringMultiArray view vs. container
- Real rel_change_L2 (const RealVector &curr_rv, const RealVector &prev_rv)

  Computes relative change between RealVectors using Euclidean L2 norm.
- Real rel_change_L2 (const RealVector &curr_rv1, const RealVector &prev_rv1, const IntVector &curr_iv, const IntVector &prev_iv, const RealVector &curr_rv2, const RealVector &prev_rv2)

  Computes relative change between Real/int/Real vector triples using Euclidean L2 norm.
- bool operator== (const IntArray &dia1, const IntArray &dia2)

  equality operator for IntArray
- bool operator!= (const IntArray &dia1, const IntArray &dia2)

  inequality operator for IntArray
- bool operator!= (const ShortArray &dsa1, const ShortArray &dsa2)

  inequality operator for ShortArray
- bool operator!= (const StringArray &dsa1, const StringArray &dsa2)

  inequality operator for StringArray
- bool operator!= (const SizetArray &sa, SizetMultiArrayConstView smav)

  inequality operator for StringArray

- std::string strtolower (const std::string &s)

  Return lowercase copy of string s.
- bool strbegins (const std::string &input, const std::string &test)

  Return true if input string begins with string test.
- bool strends (const std::string &input, const std::string &test)

  Return true if input string ends with string test.
- bool strcontains (const std::string &input, const std::string &test)

  Return true if input string contains string test.
- void build_label (String &label, const String &root_label, size_t tag)

  create a label by appending a numerical tag to the root_label
- void build_labels (StringArray &label_array, const String &root_label)

  create an array of labels by tagging root_label for each entry in label_array. Uses build_label().
- void build_labels (StringMultiArray &label_array, const String &root_label)

  create an array of labels by tagging root_label for each entry in label_array. Uses build_label().
- void build_labels_partial (StringArray &label_array, const String &root_label, size_t start_index, size_t num_items)

  create a partial array of labels by tagging root_label for a subset of entries in label_array. Uses build_label().
- void copy_row_vector (const RealMatrix &m, RealMatrix::ordinalType i, std::vector< Real > &row)

  Copies a row of a Teuchos_SerialDenseMatrix<int,Real> to std::vector<Real>
- template<typename T >
  void copy_data (const std::vector< T > &vec, T *ptr, const size_t ptr_len)

  *copy Array<T> to T**
- template<typename T >
  void copy_data (const T *ptr, const size_t ptr_len, std::vector< T > &vec)

  copy T to Array<T>*
- template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType >
  void copy_data (const std::vector< Teuchos::SerialDenseVector< OrdinalType1, ScalarType > > &va, ScalarType *ptr, const OrdinalType2 ptr_len, const String &ptr_type)

  *copy Array<Teuchos::SerialDenseVector<OT,ST> > to ST**
- template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType >
  void copy_data (const Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv, Teuchos::SerialDenseMatrix< OrdinalType1, ScalarType > &sdm, OrdinalType2 nr, OrdinalType2 nc)

  copy Teuchos::SerialDenseVector<OT,ST> to Teuchos::SerialDenseMatrix<OT,ST>
- template<typename T >
  void copy_data (const std::list< T > &dl, std::vector< T > &da)

  copy std::list<T> to std::vector<T>
- template<typename T >
  void copy_data (const std::list< T > &dl, std::vector< std::vector< T > > &d2a, size_t num_a, size_t a_len)

  copy std::list<T> to std::vector<std::vector<T> >
- template<typename T >
  void copy_data (const std::vector< std::vector< T > > &d2a, std::vector< T > &da)

  copy std::vector<vector<T> > to std::vector<T>(unroll vecOfvecs into vector)
- template<typename T >
  void copy_data (const std::map< int, T > &im, std::vector< T > &da)

copy map<int, T> to std::vector<T> (discard integer keys)

- template<typename OrdinalType , typename ScalarType >
  void copy_data (const Teuchos::SerialDenseVector< OrdinalType, ScalarType > &sdv1, Teuchos::SerialDenseVector< OrdinalType, ScalarType > &sdv2)

  copy Teuchos::SerialDenseVector<OrdinalType, ScalarType> to same (used in place of operator= when a deep copy of a vector view is needed)

- template<typename OrdinalType , typename ScalarType >
  void copy_data (const Teuchos::SerialDenseVector< OrdinalType, ScalarType > &sdv, std::vector< ScalarType > &da)

  copy Teuchos::SerialDenseVector<OrdinalType, ScalarType> to std::vector<ScalarType>

- template<typename OrdinalType , typename ScalarType >
  void copy_data (const std::vector< ScalarType > &da, Teuchos::SerialDenseVector< OrdinalType, ScalarType > &sdv)

  copy Array<ScalarType> to Teuchos::SerialDenseVector<OrdinalType, ScalarType>

- template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType >
  void copy_data (const ScalarType *ptr, const OrdinalType2 ptr_len, Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv)

  copy ScalarType to Teuchos::SerialDenseVector<OrdinalType, ScalarType>*

- template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType >
  void copy_data (const Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv, ScalarType *ptr, const OrdinalType2 ptr_len)

  copy ScalarType to Teuchos::SerialDenseVector<OrdinalType, ScalarType>*

- template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType >
  void copy_data (const Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv, std::vector< Teuchos::SerialDenseVector< OrdinalType1, ScalarType > > &sdva, OrdinalType2 num_vec, OrdinalType2 vec_len)

  copy SerialDenseVector<> to Array<SerialDenseVector<> >

- template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType >
  void copy_data_partial (const Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv1, OrdinalType2 start_index1, OrdinalType2 num_items, Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv2)

  copy portion of first SerialDenseVector to all of second SerialDenseVector

- template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType >
  void copy_data_partial (const Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv1, Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv2, OrdinalType2 start_index2)

  copy all of first SerialDenseVector to portion of second SerialDenseVector

- template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType >
  void copy_data_partial (const Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv1, OrdinalType2 start_index1, OrdinalType2 num_items, Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv2, OrdinalType2 start_index2)

  copy portion of first SerialDenseVector to portion of second SerialDenseVector

- template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType >
  void copy_data_partial (const Teuchos::SerialDenseVector< OrdinalType1, ScalarType > &sdv1, std::vector< ScalarType > &da2, OrdinalType2 start_index2)

  copy all of first SerialDenseVector to portion of second SerialDenseVector

- template<typename T >
  void copy_data_partial (const std::vector< T > &da1, size_t start_index1, size_t num_items, std::vector< T > &da2)

  copy portion of first Array<T> to all of second Array<T>
- template<typename T >
  void copy_data_partial (const std::vector< T > &da1, std::vector< T > &da2, size_t start_index2)

  copy all of first Array<T> to portion of second Array<T>
- template<typename T >
  void copy_data_partial (const std::vector< T > &da, boost::multi_array< T, 1 > &bma, size_t start_index_bma)

  copy all of first Array<T> to portion of boost::multi_array<T, 1>
- template<typename T >
  void copy_data_partial (const std::vector< T > &da1, size_t start_index1, size_t num_items, std::vector< T > &da2, size_t start_index2)

  copy portion of first Array<T> to portion of second Array<T>
- void merge_data_partial (const IntVector &d_vec, RealVector &m_vec, size_t start_index_ma)

  merge a discrete integer vector into a single continuous vector
- void merge_data_partial (const IntVector &d_vec, RealArray &m_array, size_t start_index_ma)

  merge a discrete integer vector into a single continuous array
- template<typename OrdinalType , typename ScalarType >
  const ScalarType & set_index_to_value (OrdinalType index, const std::set< ScalarType > &values)

  retrieve the set value corresponding to the passed index
- template<typename ScalarType >
  size_t set_value_to_index (const ScalarType &value, const std::set< ScalarType > &values)

  calculate the set index corresponding to the passed value
- template<typename OrdinalType , typename KeyType , typename ValueType >
  const KeyType & map_index_to_key (OrdinalType index, const std::map< KeyType, ValueType > &pairs)

  retrieve the set value corresponding to the passed index
- template<typename OrdinalType , typename KeyType , typename ValueType >
  const ValueType & map_index_to_value (OrdinalType index, const std::map< KeyType, ValueType > &pairs)

  retrieve the set value corresponding to the passed index
- template<typename KeyType , typename ValueType >
  void map_keys_to_set (const std::map< KeyType, ValueType > &source_map, std::set< KeyType > &target_set)

  calculate the map index corresponding to the passed key
- template<typename KeyType , typename ValueType >
  size_t map_key_to_index (const KeyType &key, const std::map< KeyType, ValueType > &pairs)

  calculate the map index corresponding to the passed key
- template<typename OrdinalType , typename ScalarType >
  void x_y_pairs_to_x_set (const Teuchos::SerialDenseVector< OrdinalType, ScalarType > &xy_pairs, std::set< ScalarType > &x_set)

  convert a SerialDenseVector of head-to-tail (x,y) pairs into a std::set of (x), discarding the y values
- template<typename ContainerType >
  size_t find_index (const ContainerType &c, const typename ContainerType::value_type &search_data)

  generic find_index (inactive)

- template<typename MultiArrayType , typename DakArrayType >
  void copy_data (const MultiArrayType &ma, DakArrayType &da)

  generic copy (inactive)
- template<typename T >
  size_t find_index (const boost::multi_array< T, 1 > &bma, const T &search_data)

  compute the index of an entry within a boost::multi_array
- size_t find_index (SizetMultiArrayConstView bmacv, size_t search_data)

  compute the index of an entry within a boost::multi_array view
- size_t find_index (StringMultiArrayConstView bmacv, const String &search_data)

  compute the index of an entry within a boost::multi_array view
- template<typename ListT >
  size_t find_index (const ListT &l, const typename ListT::value_type &val)

  compute the index of an entry within a std::list
- void copy_data (SizetMultiArrayConstView ma, SizetArray &da)

  copy boost::multi_array view to Array
- void copy_data (StringMultiArrayConstView ma, StringArray &da)

  copy boost::multi_array view to Array
- template<typename ListT >
  ListT::const_iterator find_if (const ListT &c, bool(∗test_fn)(const typename ListT::value_type &, const std::string &), const std::string &test_fn_data)

  return an iterator to the first list element satisfying the predicate test_fn w.r.t. the passed test_fn_data; end if not found
- template<typename DakContainerType >
  bool contains (const DakContainerType &v, const typename DakContainerType::value_type &val)

  return true if the item val appears in container v
- void abort_handler (int code)

  global function which handles serial or parallel aborts
- void abort_throw_or_exit (int code)

  throw or exit depending on abort_mode
- void register_signal_handlers ()

  Tie various signal handlers to Dakota's abort_handler function.
- void mpi_debug_hold ()

  Global function to hold Dakota processes to help with MPI debugging.
- template<typename T >
  T abort_handler_t (int code)
- ResultsKeyType make_key (const StrStrSizet &iterator_id, const std::string &data_name)

  Make a full ResultsKeyType from the passed iterator_id and data_name.
- MetaDataValueType make_metadatavalue (StringMultiArrayConstView labels)

  create MetaDataValueType from the passed strings
- MetaDataValueType make_metadatavalue (StringMultiArrayConstView cv_labels, StringMultiArrayConstView div_labels, StringMultiArrayConstView drv_labels, const StringArray &resp_labels)

  create MetaDataValueType from the passed strings
- MetaDataValueType make_metadatavalue (const StringArray &resp_labels)

  create MetaDataValueType from the passed strings

- MetaDataValueType make_metadatavalue (const std::string &)

  create MetaDataValueType from the passed strings
- MetaDataValueType make_metadatavalue (const std::string &, const std::string &)

  create MetaDataValueType from the passed strings
- MetaDataValueType make_metadatavalue (const std::string &, const std::string &, const std::string &)

  create MetaDataValueType from the passed strings
- MetaDataValueType make_metadatavalue (const std::string &, const std::string &, const std::string &, const std::string &)

  create MetaDataValueType from the passed strings
- MetaDataValueType **make_metadatavalue** (StringMultiArrayConstView cv_labels, StringMultiArrayConstView div_labels, StringMultiArrayConstView dsv_labels, StringMultiArrayConstView drv_labels, const StringArray &resp_labels)
- std::istream & operator>> (std::istream &s, ActiveSet &set)

  std::istream extraction operator for ActiveSet. Calls read(std::istream&).
- std::ostream & operator<< (std::ostream &s, const ActiveSet &set)

  std::ostream insertion operator for ActiveSet. Calls write(std::istream&).
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, ActiveSet &set)

  MPIUnpackBuffer extraction operator for ActiveSet. Calls read(MPIUnpackBuffer&).
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const ActiveSet &set)

  MPIPackBuffer insertion operator for ActiveSet. Calls write(MPIPackBuffer&).
- bool operator!= (const ActiveSet &set1, const ActiveSet &set2)

  inequality operator for ActiveSet
- std::istream & operator>> (std::istream &s, Constraints &con)

  std::istream extraction operator for Constraints
- std::ostream & operator<< (std::ostream &s, const Constraints &con)

  std::ostream insertion operator for Constraints
- bool interface_id_compare (const Interface &interface_in, const void *id)

  global comparison function for Interface
- bool method_id_compare (const Iterator &iterator, const void *id)

  global comparison function for Iterator
- bool model_id_compare (const Model &model, const void *id)

  global comparison function for Model
- bool operator== (const Model &m1, const Model &m2)

  equality operator for Envelope is true if same letter instance
- bool operator!= (const Model &m1, const Model &m2)

  inequality operator for Envelope is true if different letter instance
- bool responses_id_compare (const Response &resp, const void *id)

  global comparison function for Response
- std::istream & operator>> (std::istream &s, Response &response)

  std::istream extraction operator for Response. Calls read(std::istream&).
- std::ostream & operator<< (std::ostream &s, const Response &response)

  std::ostream insertion operator for Response. Calls write(std::ostream&).
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, Response &response)

MPIUnpackBuffer extraction operator for Response. Calls read(MPIUnpackBuffer&).

- MPIPackBuffer & operator<< (MPIPackBuffer &s, const Response &response)

  MPIPackBuffer insertion operator for Response. Calls write(MPIPackBuffer&).
- bool operator== (const Response &resp1, const Response &resp2)

  equality operator for Response
- bool operator!= (const Response &resp1, const Response &resp2)

  inequality operator for Response
- std::string re_match (const std::string &token, const boost::regex &re)

  Global utility function to ease migration from CtelRegExp to Boost.Regex.
- bool variables_id_compare (const Variables &vars, const void *id)

  global comparison function for Variables
- std::istream & operator>> (std::istream &s, Variables &vars)

  std::istream extraction operator for Variables.
- std::ostream & operator<< (std::ostream &s, const Variables &vars)

  std::ostream insertion operator for Variables.
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, Variables &vars)

  MPIUnpackBuffer extraction operator for Variables.
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const Variables &vars)

  MPIPackBuffer insertion operator for Variables.
- bool operator!= (const Variables &vars1, const Variables &vars2)

  inequality operator for Variables
- template<typename OrdinalType , typename ScalarType1 , typename ScalarType2 , typename ScalarType3 , typename ScalarType4 >
  void write_ordered (std::ostream &s, const SizetArray &comp_totals, const Teuchos::SerialDenseVector< OrdinalType, ScalarType1 > &c_vector, const Teuchos::SerialDenseVector< OrdinalType, ScalarType2 > &di_vector, const Teuchos::SerialDenseVector< OrdinalType, ScalarType3 > &ds_vector, const Teuchos::SerialDenseVector< OrdinalType, ScalarType4 > &dr_vector)

  free function to write Variables data vectors in input spec ordering
- template<typename OrdinalType , typename ScalarType1 , typename ScalarType2 , typename ScalarType3 , typename ScalarType4 >
  void write_ordered (std::ostream &s, const SizetArray &comp_totals, const Teuchos::SerialDenseVector< OrdinalType, ScalarType1 > &c_vector, const Teuchos::SerialDenseVector< OrdinalType, ScalarType2 > &di_vector, const boost::multi_array< ScalarType3, 1 > &ds_array, const Teuchos::SerialDenseVector< OrdinalType, ScalarType4 > &dr_vector)

  free function to write Variables data vectors in input spec ordering
- template<typename ScalarType >
  void write_ordered (std::ostream &s, const SizetArray &comp_totals, const std::vector< ScalarType > &c_array, const std::vector< ScalarType > &di_array, const std::vector< ScalarType > &ds_array, const std::vector< ScalarType > &dr_array)

  free function to write Variables data vectors in input spec ordering
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const DataEnvironment &data)

  MPIPackBuffer insertion operator for DataEnvironment.
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, DataEnvironment &data)

  MPIUnpackBuffer extraction operator for DataEnvironment.
- std::ostream & operator<< (std::ostream &s, const DataEnvironment &data)

std::ostream insertion operator for DataEnvironment

- static String **interface_enum_to_string** (unsigned short interface_type)
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const DataInterface &data)

  MPIPackBuffer insertion operator for DataInterface.
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, DataInterface &data)

  MPIUnpackBuffer extraction operator for DataInterface.
- std::ostream & operator<< (std::ostream &s, const DataInterface &data)

  std::ostream insertion operator for DataInterface
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const DataMethod &data)

  MPIPackBuffer insertion operator for DataMethod.
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, DataMethod &data)

  MPIUnpackBuffer extraction operator for DataMethod.
- std::ostream & operator<< (std::ostream &s, const DataMethod &data)

  std::ostream insertion operator for DataMethod
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const DataModel &data)

  MPIPackBuffer insertion operator for DataModel.
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, DataModel &data)

  MPIUnpackBuffer extraction operator for DataModel.
- std::ostream & operator<< (std::ostream &s, const DataModel &data)

  std::ostream insertion operator for DataModel
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const DataResponses &data)

  MPIPackBuffer insertion operator for DataResponses.
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, DataResponses &data)

  MPIUnpackBuffer extraction operator for DataResponses.
- std::ostream & operator<< (std::ostream &s, const DataResponses &data)

  std::ostream insertion operator for DataResponses
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const DataVariables &data)

  MPIPackBuffer insertion operator for DataVariables.
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, DataVariables &data)

  MPIUnpackBuffer extraction operator for DataVariables.
- std::ostream & operator<< (std::ostream &s, const DataVariables &data)

  std::ostream insertion operator for DataVariables
- int **dlsolver_option** (Opt_Info *)
- RealVector const * **continuous_lower_bounds** (Optimizer1 *o)
- RealVector const * **continuous_upper_bounds** (Optimizer1 *o)
- RealVector const * **nonlinear_ineq_constraint_lower_bounds** (Optimizer1 *o)
- RealVector const * **nonlinear_ineq_constraint_upper_bounds** (Optimizer1 *o)
- RealVector const * **nonlinear_eq_constraint_targets** (Optimizer1 *o)
- RealVector const * **linear_ineq_constraint_lower_bounds** (Optimizer1 *o)
- RealVector const * **linear_ineq_constraint_upper_bounds** (Optimizer1 *o)
- RealVector const * **linear_eq_constraint_targets** (Optimizer1 *o)
- RealMatrix const * **linear_ineq_constraint_coeffs** (Optimizer1 *o)
- RealMatrix const * **linear_eq_constraint_coeffs** (Optimizer1 *o)
- void **ComputeResponses** (Optimizer1 *o, int mode, int n, double *x)

- void **GetFuncs** (Optimizer1 ∗o, int m0, int m1, double ∗f)
- void **GetGrads** (Optimizer1 ∗o, int m0, int m1, int n, int is, int js, double ∗g)
- void **GetContVars** (Optimizer1 ∗o, int n, double ∗x)
- void **SetBestContVars** (Optimizer1 ∗o, int n, double ∗x)
- void **SetBestRespFns** (Optimizer1 ∗o, int n, double ∗x)
- void ∗ **dl_constructor** (Optimizer1 ∗, Dakota_funcs ∗, dl_find_optimum_t ∗, dl_destructor_t ∗)
- static RealVector const ∗ **continuous_lower_bounds1** (Optimizer1 ∗o)
- static RealVector const ∗ **continuous_upper_bounds1** (Optimizer1 ∗o)
- static RealVector const ∗ **nonlinear_ineq_constraint_lower_bounds1** (Optimizer1 ∗o)
- static RealVector const ∗ **nonlinear_ineq_constraint_upper_bounds1** (Optimizer1 ∗o)
- static RealVector const ∗ **nonlinear_eq_constraint_targets1** (Optimizer1 ∗o)
- static RealVector const ∗ **linear_ineq_constraint_lower_bounds1** (Optimizer1 ∗o)
- static RealVector const ∗ **linear_ineq_constraint_upper_bounds1** (Optimizer1 ∗o)
- static RealVector const ∗ **linear_eq_constraint_targets1** (Optimizer1 ∗o)
- static RealMatrix const ∗ **linear_eq_constraint_coeffs1** (Optimizer1 ∗o)
- static RealMatrix const ∗ **linear_ineq_constraint_coeffs1** (Optimizer1 ∗o)
- static void **ComputeResponses1** (Optimizer1 ∗o, int mode, int n, double ∗x)
- static void **GetFuncs1** (Optimizer1 ∗o, int m0, int m1, double ∗f)
- static void **GetGrads1** (Optimizer1 ∗o, int m0, int m1, int n, int is, int js, double ∗g)
- static void **GetContVars1** (Optimizer1 ∗o, int n, double ∗x)
- static void **SetBestContVars1** (Optimizer1 ∗o, int n, double ∗x)
- static void **SetBestDiscVars1** (Optimizer1 ∗o, int n, int ∗x)
- static void **SetBestRespFns1** (Optimizer1 ∗o, int n, double ∗x)
- static double **Get_Real1** (Optimizer1 ∗o, const char ∗name)
- static int **Get_Int1** (Optimizer1 ∗o, const char ∗name)
- static bool **Get_Bool1** (Optimizer1 ∗o, const char ∗name)
- DOTOptimizer ∗ **new_DOTOptimizer** (ProblemDescDB &problem_db)
- DOTOptimizer ∗ **new_DOTOptimizer** (Model &model)
- DOTOptimizer ∗ **new_DOTOptimizer** (ProblemDescDB &problem_db, Model &model)
- void read_historical_data (const std::string &expDataFileName, const std::string &context_message, size_t numExperiments, size_t numExpConfigVars, size_t numFunctions, size_t numExpStdDeviationsRead, bool expDataFileAnnotated, bool calc_sigma_from_data, RealMatrix &xObsData, RealMatrix &yObsData, RealMatrix &yStdData)

  Read data in historical format into x, y, sigma matrices.
- Real getdist (const RealVector &x1, const RealVector &x2)
- Real mindist (const RealVector &x, const RealMatrix &xset, int except)
- Real mindistindx (const RealVector &x, const RealMatrix &xset, const IntArray &indx)
- Real getRmax (const RealMatrix &xset)
- int start_grid_computing (char ∗analysis_driver_script, char ∗params_file, char ∗results_file)
- int stop_grid_computing ()
- int perform_analysis (char ∗iteration_num)
- template<typename T >
  string asstring (const T &val)

  Creates a string from the argument val using an ostringstream.
- **PACKBUF** (int, MPI_INT) PACKBUF(u_int
- MPI_UNSIGNED **PACKBUF** (long, MPI_LONG) PACKBUF(u_long

- MPI_UNSIGNED MPI_UNSIGNED_LONG **PACKBUF** (short, MPI_SHORT) PACKBUF(u_short
- MPI_UNSIGNED MPI_UNSIGNED_LONG
  MPI_UNSIGNED_SHORT **PACKBUF** (char, MPI_CHAR) PACKBUF(u_char
- MPI_UNSIGNED MPI_UNSIGNED_LONG
  MPI_UNSIGNED_SHORT
  MPI_UNSIGNED_CHAR **PACKBUF** (double, MPI_DOUBLE) PACKBUF(float
- **UNPACKBUF** (int, MPI_INT) UNPACKBUF(u_int
- MPI_UNSIGNED **UNPACKBUF** (long, MPI_LONG) UNPACKBUF(u_long
- MPI_UNSIGNED MPI_UNSIGNED_LONG **UNPACKBUF** (short, MPI_SHORT) UNPACKBUF(u_short
- MPI_UNSIGNED MPI_UNSIGNED_LONG
  MPI_UNSIGNED_SHORT **UNPACKBUF** (char, MPI_CHAR) UNPACKBUF(u_char
- MPI_UNSIGNED MPI_UNSIGNED_LONG
  MPI_UNSIGNED_SHORT
  MPI_UNSIGNED_CHAR **UNPACKBUF** (double, MPI_DOUBLE) UNPACKBUF(float
- **PACKSIZE** (int, MPI_INT) PACKSIZE(u_int
- MPI_UNSIGNED **PACKSIZE** (long, MPI_LONG) PACKSIZE(u_long
- MPI_UNSIGNED MPI_UNSIGNED_LONG **PACKSIZE** (short, MPI_SHORT) PACKSIZE(u_short
- MPI_UNSIGNED MPI_UNSIGNED_LONG
  MPI_UNSIGNED_SHORT **PACKSIZE** (char, MPI_CHAR) PACKSIZE(u_char
- MPI_UNSIGNED MPI_UNSIGNED_LONG
  MPI_UNSIGNED_SHORT
  MPI_UNSIGNED_CHAR **PACKSIZE** (double, MPI_DOUBLE) PACKSIZE(float
- MPI_UNSIGNED MPI_UNSIGNED_LONG
  MPI_UNSIGNED_SHORT
  MPI_UNSIGNED_CHAR MPI_FLOAT
  int MPIPackSize (const bool &data, const int num=1)

  return packed size of a bool
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const int &data)

  insert an int
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const u_int &data)

  insert a u_int
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const long &data)

  insert a long
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const u_long &data)

  insert a u_long
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const short &data)

  insert a short
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const u_short &data)

  insert a u_short
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const char &data)

  insert a char
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const u_char &data)

  insert a u_char
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const double &data)

  insert a double

- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const float &data)

  insert a float
- MPIPackBuffer & operator<< (MPIPackBuffer &buff, const bool &data)

  insert a bool
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, int &data)

  extract an int
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, u_int &data)

  extract a u_int
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, long &data)

  extract a long
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, u_long &data)

  extract a u_long
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, short &data)

  extract a short
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, u_short &data)

  extract a u_short
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, char &data)

  extract a char
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, u_char &data)

  extract a u_char
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, double &data)

  extract a double
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, float &data)

  extract a float
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &buff, bool &data)

  extract a bool
- template<class ContainerT >
  void container_read (ContainerT &c, MPIUnpackBuffer &s)

  Read a generic container (vector<T>, list<T>) from MPIUnpackBuffer, s.
- template<class ContainerT >
  void container_write (const ContainerT &c, MPIPackBuffer &s)

  Write a generic container to MPIPackBuffer, s.
- template<typename Block , typename Allocator >
  MPIPackBuffer & operator<< (MPIPackBuffer &s, const boost::dynamic_bitset< Block, Allocator > &bs)

  stream insertion for BitArray
- template<typename Block , typename Allocator >
  MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, boost::dynamic_bitset< Block, Allocator > &bs)

  stream extraction for BitArray
- template<class ContainerT >
  MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, ContainerT &data)

  global MPIUnpackBuffer extraction operator for generic container
- template<class ContainerT >
  MPIPackBuffer & operator<< (MPIPackBuffer &s, const ContainerT &data)

global MPIPackBuffer insertion operator for generic container

- int MPIPackSize (const int &data, const int num=1)

  return packed size of an int
- int MPIPackSize (const u_int &data, const int num=1)

  return packed size of a u_int
- int MPIPackSize (const long &data, const int num=1)

  return packed size of a long
- int MPIPackSize (const u_long &data, const int num=1)

  return packed size of a u_long
- int MPIPackSize (const short &data, const int num=1)

  return packed size of a short
- int MPIPackSize (const u_short &data, const int num=1)

  return packed size of a u_short
- int MPIPackSize (const char &data, const int num=1)

  return packed size of a char
- int MPIPackSize (const u_char &data, const int num=1)

  return packed size of a u_char
- int MPIPackSize (const double &data, const int num=1)

  return packed size of a double
- int MPIPackSize (const float &data, const int num=1)

  return packed size of a float
- int **nidr_parse** (const char ∗, FILE ∗)
- const char ∗∗ **arg_list_adjust** (const char ∗∗, void ∗∗)
- int **not_executable** (const char ∗driver_name, const char ∗tdir)
- static void **scale_chk** (StringArray &ST, RealVector &S, const char ∗what, const char ∗∗univ)
- static void **BuildLabels** (StringArray ∗sa, size_t nsa, size_t n1, size_t n2, const char ∗stub)
- static int **mixed_check** (IntSet ∗S, int n, IntArray ∗iv, const char ∗what)
- static void **mixed_check2** (size_t n, IntArray ∗iv, const char ∗what)
- static int **wronglen** (size_t n, RealVector ∗V, const char ∗what)
- static int **wronglen** (size_t n, IntVector ∗V, const char ∗what)
- static void **Vcopyup** (RealVector ∗V, RealVector ∗M, size_t i, size_t n)
- static void **Set_rv** (RealVector ∗V, double d, size_t n)
- static void **Set_iv** (IntVector ∗V, int d, size_t n)
- static void **wrong_number** (const char ∗what, const char ∗kind, size_t nsv, size_t m)
- static void **too_small** (const char ∗kind)
- static void **not_div** (const char ∗kind, size_t nsv, size_t m)
- static void **suppressed** (const char ∗kind, int ndup, int ∗ip, String ∗sp, Real ∗rp)
- static void **bad_initial_ivalue** (const char ∗kind, int val)
- static void **bad_initial_svalue** (const char ∗kind, String val)
- static void **bad_initial_rvalue** (const char ∗kind, Real val)
- static void **Vgen_ContinuousDes** (DataVariablesRep ∗dv, size_t offset)
- static void **Vgen_DiscreteDesRange** (DataVariablesRep ∗dv, size_t offset)
- static void **Vgen_ContinuousState** (DataVariablesRep ∗dv, size_t offset)
- static void **Vgen_DiscreteStateRange** (DataVariablesRep ∗dv, size_t offset)
- static void **Vchk_NormalUnc** (DataVariablesRep ∗dv, size_t offset, Var_Info ∗vi)

- static void **Vgen_NormalUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_LognormalUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_LognormalUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_UniformUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_UniformUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_LoguniformUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_LoguniformUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_TriangularUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_TriangularUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_ExponentialUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_ExponentialUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_BetaUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_BetaUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_GammaUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_GammaUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_GumbelUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_GumbelUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_FrechetUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_FrechetUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_WeibullUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_WeibullUnc** (DataVariablesRep *dv, size_t offset)
- static void Vchk_HistogramBinUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)

  Check the histogram bin input data, normalize the counts and populate the histogramUncBinPairs map data structure; map keys are guaranteed unique since the abscissas must increase.
- static void Vgen_HistogramBinUnc (DataVariablesRep *dv, size_t offset)

  Infer lower/upper bounds for histogram and set initial variable values based on initial_point or moments, snapping to bounds as needed. (Histogram bin doesn't have lower/upper bounds specifcation)
- static void **Vchk_PoissonUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_PoissonUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_BinomialUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_BinomialUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_NegBinomialUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_NegBinomialUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_GeometricUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_GeometricUnc** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_HyperGeomUnc** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_HyperGeomUnc** (DataVariablesRep *dv, size_t offset)
- static void Vchk_HistogramPtIntUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)

  Check the histogram point integer input data, normalize the counts, and populate DataVariables::histogramUncPointIntPairs; map keys are guaranteed unique since the abscissas must increase.
- static void Vgen_HistogramPtIntUnc (DataVariablesRep *dv, size_t offset)

  Use the integer-valued point histogram data to initialize the lower, upper, and initial values of the variables, using value closest to mean if no initial point.
- static void Vchk_HistogramPtStrUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)

  Check the histogram point string input data, normalize the counts, and populate DataVariables::histogramUncPointStrPairs; map keys are guaranteed unique since the abscissas must increase (lexicographically)

- static void Vgen_HistogramPtStrUnc (DataVariablesRep *dv, size_t offset)

  Use the string-valued point histogram data to initialize the lower, upper, and initial values of the variables, using index closest to mean index if no initial point.
- static void Vchk_HistogramPtRealUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)

  Check the histogram point integer real data, normalize the counts, and populate DataVariables::histogramUncPointRealPairs; map keys are guaranteed unique since the abscissas must increase.
- static void Vgen_HistogramPtRealUnc (DataVariablesRep *dv, size_t offset)

  Use the real-valued point histogram data to initialize the lower, upper, and initial values of the variables, using value closest to mean if no initial point.
- static void Vchk_ContinuousIntervalUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)

  Check the continuous interval uncertain input data and populate DataVariables::continuousIntervalUncBasicProbs; map keys (real intervals) are checked for uniqueness because we don't have a theoretically sound way to combine duplicate intervals.
- static void **Vgen_ContinuousIntervalUnc** (DataVariablesRep *dv, size_t offset)
- static void Vchk_DiscreteIntervalUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)

  Check the discrete interval uncertain input data and populate DataVariables::discreteIntervalUncBasicProbs; map keys (integer intervals) are checked for uniqueness because we don't have a theoretically sound way to combine duplicate intervals.
- static void **Vgen_DiscreteIntervalUnc** (DataVariablesRep *dv, size_t offset)
- static bool check_set_keys (size_t num_v, size_t ds_len, const char *kind, IntArray *input_nds, int &avg_num_ds)

  validate the number of set elements (values) given the number of variables and an optional apportionment with elements_per_variable; return the average number per variable if equally distributed
- static void Vchk_DIset (size_t num_v, const char *kind, IntArray *input_ndsi, IntVector *input_dsi, IntSetArray &dsi_all, IntVector &dsi_init_pt)

  check discrete sets of integers (design and state variables); error if a duplicate value is specified
- static void Vchk_DIset (size_t num_v, const char *kind, IntArray *input_ndsi, IntVector *input_dsi, RealVector *input_dsip, IntRealMapArray &dsi_vals_probs, IntVector &dsi_init_pt)

  check discrete sets of integers (uncertain variables); error if a duplicate value is specified
- static void **Vchk_DSset** (size_t num_v, const char *kind, IntArray *input_ndss, StringArray *input_dss, StringSetArray &dss_all, StringArray &dss_init_pt)
- static void **Vchk_DSset** (size_t num_v, const char *kind, IntArray *input_ndss, StringArray *input_dss, RealVector *input_dssp, StringRealMapArray &dss_vals_probs, StringArray &dss_init_pt)
- static void **Vchk_DRset** (size_t num_v, const char *kind, IntArray *input_ndsr, RealVector *input_dsr, RealSetArray &dsr_all, RealVector &dsr_init_pt)
- static void **Vchk_DRset** (size_t num_v, const char *kind, IntArray *input_ndsr, RealVector *input_dsr, RealVector *input_dsrp, RealRealMapArray &dsr_vals_probs, RealVector &dsr_init_pt)
- static bool **check_LUV_size** (size_t num_v, IntVector &L, IntVector &U, IntVector &V, bool aggregate_LUV, size_t offset)
- static bool **check_LUV_size** (size_t num_v, StringArray &L, StringArray &U, StringArray &V, bool aggregate_LUV, size_t offset)
- static bool **check_LUV_size** (size_t num_v, RealVector &L, RealVector &U, RealVector &V, bool aggregate_LUV, size_t offset)
- static void **Vgen_DIset** (size_t num_v, IntSetArray &sets, IntVector &L, IntVector &U, IntVector &V, bool aggregate_LUV=false, size_t offset=0)
- static void Vgen_DSset (size_t num_v, StringSetArray &sets, StringArray &L, StringArray &U, StringArray &V, bool aggregate_LUV=false, size_t offset=0)

generate lower, upper, and initial point for string-valued sets

- static void **Vgen_DIset** (size_t num_v, IntRealMapArray &vals_probs, IntVector &IP, IntVector &L, IntVector &U, IntVector &V, bool aggregate_LUV=false, size_t offset=0)
- static void **Vgen_DRset** (size_t num_v, RealSetArray &sets, RealVector &L, RealVector &U, RealVector &V, bool aggregate_LUV=false, size_t offset=0)
- static void **Vgen_DRset** (size_t num_v, RealRealMapArray &vals_probs, RealVector &IP, RealVector &L, RealVector &U, RealVector &V, bool aggregate_LUV=false, size_t offset=0)
- static void **Vgen_DSset** (size_t num_v, StringRealMapArray &vals_probs, StringArray &IP, StringArray &L, StringArray &U, StringArray &V, bool aggregate_LUV=false, size_t offset=0)
- static void **Vchk_DiscreteDesSetInt** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_DiscreteDesSetInt** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_DiscreteDesSetStr** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_DiscreteDesSetStr** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_DiscreteDesSetReal** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_DiscreteDesSetReal** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_DiscreteUncSetInt** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_DiscreteUncSetInt** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_DiscreteUncSetStr** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_DiscreteUncSetStr** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_DiscreteUncSetReal** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_DiscreteUncSetReal** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_DiscreteStateSetInt** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_DiscreteStateSetInt** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_DiscreteStateSetStr** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_DiscreteStateSetStr** (DataVariablesRep *dv, size_t offset)
- static void **Vchk_DiscreteStateSetReal** (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void **Vgen_DiscreteStateSetReal** (DataVariablesRep *dv, size_t offset)
- static const char * **Var_Name** (StringArray *sa, char *buf, size_t i)
- static void Var_RealBoundIPCheck (DataVariablesRep *dv, Var_rcheck *b)

  For real-valued variables: verify lengths of bounds and initial point, validate bounds and adjust initial point to bounds.
- static void Var_IntBoundIPCheck (DataVariablesRep *dv, Var_icheck *ib)

  For integer-valued variables: verify lengths of bounds and initial point, validate bounds and initial point against bounds.
- static void **flatten_rva** (RealVectorArray *rva, RealVector **prv)
- static void **flatten_iva** (IntVectorArray *iva, IntVector **piv)
- static void **flatten_rsm** (RealSymMatrix *rsm, RealVector **prv)
- static void **flatten_rsa** (RealSetArray *rsa, RealVector **prv)
- static void **flatten_ssa** (StringSetArray *ssa, StringArray **psa)
- static void **flatten_isa** (IntSetArray *isa, IntVector **piv)
- static void **flatten_rrma_keys** (RealRealMapArray *rrma, RealVector **prv)
- static void **flatten_rrma_values** (RealRealMapArray *rrma, RealVector **prv)
- static void **flatten_irma_keys** (IntRealMapArray *irma, IntVector **piv)
- static void **flatten_irma_values** (IntRealMapArray *irma, RealVector **prv)
- static void **flatten_srma_keys** (StringRealMapArray *srma, StringArray **psa)
- static void **flatten_srma_values** (StringRealMapArray *srma, RealVector **prv)

- static void flatten_real_intervals (const RealRealPairRealMapArray &rrprma, RealVector **probs, RealVector **lb, RealVector **ub)

  Flatten real-valued interval uncertain variable intervals and probabilities back into separate arrays.
- static void flatten_int_intervals (const IntIntPairRealMapArray &iiprma, RealVector **probs, IntVector **lb, IntVector **ub)

  Flatten integer-valued interval uncertain variable intervals and probabilities back into separate arrays.
- static void **var_iulbl** (const char *keyname, Values *val, VarLabel *vl)
- static Iface_mp_Rlit **MP3** (failAction, recoveryFnVals, recover)
- static Iface_mp_ilit **MP3** (failAction, retryLimit, retry)
- static Iface_mp_lit **MP2** (failAction, abort)
- static Iface_mp_lit **MP2** (failAction, continuation)
- static Iface_mp_type **MP2s** (analysisScheduling, MASTER_SCHEDULING)
- static Iface_mp_type **MP2s** (analysisScheduling, PEER_SCHEDULING)
- static Iface_mp_type **MP2s** (evalScheduling, MASTER_SCHEDULING)
- static Iface_mp_type **MP2s** (evalScheduling, PEER_DYNAMIC_SCHEDULING)
- static Iface_mp_type **MP2s** (evalScheduling, PEER_STATIC_SCHEDULING)
- static Iface_mp_type **MP2s** (asynchLocalEvalScheduling, DYNAMIC_SCHEDULING)
- static Iface_mp_type **MP2s** (asynchLocalEvalScheduling, STATIC_SCHEDULING)
- static Iface_mp_type **MP2s** (interfaceSynchronization, ASYNCHRONOUS_INTERFACE)
- static Iface_mp_type **MP2s** (interfaceSynchronization, SYNCHRONOUS_INTERFACE)
- static Iface_mp_utype **MP2s** (interfaceType, TEST_INTERFACE)
- static Iface_mp_utype **MP2s** (interfaceType, FORK_INTERFACE)
- static Iface_mp_utype **MP2s** (interfaceType, GRID_INTERFACE)
- static Iface_mp_utype **MP2s** (interfaceType, MATLAB_INTERFACE)
- static Iface_mp_utype **MP2s** (interfaceType, PYTHON_INTERFACE)
- static Iface_mp_utype **MP2s** (interfaceType, SCILAB_INTERFACE)
- static Iface_mp_utype **MP2s** (interfaceType, SYSTEM_INTERFACE)
- static String **MP_** (algebraicMappings)
- static String **MP_** (idInterface)
- static String **MP_** (inputFilter)
- static String **MP_** (outputFilter)
- static String **MP_** (parametersFile)
- static String **MP_** (resultsFile)
- static String **MP_** (workDir)
- static String2DArray **MP_** (analysisComponents)
- static StringArray **MP_** (analysisDrivers)
- static StringArray **MP_** (copyFiles)
- static StringArray **MP_** (linkFiles)
- static bool **MP_** (activeSetVectorFlag)
- static bool **MP_** (allowExistingResultsFlag)
- static bool **MP_** (apreproFlag)
- static bool **MP_** (dirSave)
- static bool **MP_** (dirTag)
- static bool **MP_** (evalCacheFlag)
- static bool **MP_** (fileSaveFlag)
- static bool **MP_** (fileTagFlag)

- static bool **MP_** (nearbyEvalCacheFlag)
- static bool **MP_** (numpyFlag)
- static bool **MP_** (restartFileFlag)
- static bool **MP_** (templateReplace)
- static bool **MP_** (useWorkdir)
- static bool **MP_** (verbatimFlag)
- static int **MP_** (analysisServers)
- static int **MP_** (asynchLocalAnalysisConcurrency)
- static int **MP_** (asynchLocalEvalConcurrency)
- static int **MP_** (evalServers)
- static int **MP_** (procsPerAnalysis)
- static int **MP_** (procsPerEval)
- static Real **MP_** (nearbyEvalCacheTol)
- static IntVector **MP_** (primeBase)
- static IntVector **MP_** (sequenceLeap)
- static IntVector **MP_** (sequenceStart)
- static IntVector **MP_** (stepsPerVariable)
- static Method_mp_ilit2 **MP3** (replacementType, numberRetained, chc)
- static Method_mp_ilit2 **MP3** (replacementType, numberRetained, elitist)
- static Method_mp_ilit2 **MP3** (replacementType, numberRetained, random)
- static Method_mp_ilit2z **MP3** (crossoverType, numCrossPoints, multi_point_binary)
- static Method_mp_ilit2z **MP3** (crossoverType, numCrossPoints, multi_point_parameterized_binary)
- static Method_mp_ilit2z **MP3** (crossoverType, numCrossPoints, multi_point_real)
- static Method_mp_lit **MP2** (batchSelectionType, naive)
- static Method_mp_lit **MP2** (batchSelectionType, distance_penalty)
- static Method_mp_lit **MP2** (batchSelectionType, topology)
- static Method_mp_lit **MP2** (batchSelectionType, constant_liar)
- static Method_mp_lit **MP2** (boxDivision, all_dimensions)
- static Method_mp_lit **MP2** (boxDivision, major_dimension)
- static Method_mp_lit **MP2** (convergenceType, average_fitness_tracker)
- static Method_mp_lit **MP2** (convergenceType, best_fitness_tracker)
- static Method_mp_lit **MP2** (convergenceType, metric_tracker)
- static Method_mp_lit **MP2** (crossoverType, blend)
- static Method_mp_lit **MP2** (crossoverType, two_point)
- static Method_mp_lit **MP2** (crossoverType, uniform)
- static Method_mp_lit **MP2** (evalSynchronize, blocking)
- static Method_mp_lit **MP2** (evalSynchronize, nonblocking)
- static Method_mp_lit **MP2** (expansionSampleType, incremental_lhs)
- static Method_mp_lit **MP2** (exploratoryMoves, adaptive)
- static Method_mp_lit **MP2** (exploratoryMoves, multi_step)
- static Method_mp_lit **MP2** (exploratoryMoves, simple)
- static Method_mp_lit **MP2** (fitnessType, domination_count)
- static Method_mp_lit **MP2** (fitnessType, layer_rank)
- static Method_mp_lit **MP2** (fitnessType, linear_rank)
- static Method_mp_lit **MP2** (fitnessType, merit_function)
- static Method_mp_lit **MP2** (fitnessType, proportional)

- static Method_mp_lit **MP2** (fitnessMetricType, predicted_variance)
- static Method_mp_lit **MP2** (fitnessMetricType, distance)
- static Method_mp_lit **MP2** (fitnessMetricType, gradient)
- static Method_mp_lit **MP2** (initializationType, random)
- static Method_mp_lit **MP2** (initializationType, unique_random)
- static Method_mp_lit **MP2** (lipschitzType, global)
- static Method_mp_lit **MP2** (lipschitzType, local)
- static Method_mp_lit **MP2** (meritFunction, merit_max)
- static Method_mp_lit **MP2** (meritFunction, merit_max_smooth)
- static Method_mp_lit **MP2** (meritFunction, merit1)
- static Method_mp_lit **MP2** (meritFunction, merit1_smooth)
- static Method_mp_lit **MP2** (meritFunction, merit2)
- static Method_mp_lit **MP2** (meritFunction, merit2_smooth)
- static Method_mp_lit **MP2** (meritFunction, merit2_squared)
- static Method_mp_lit **MP2** (mcmcType, dram)
- static Method_mp_lit **MP2** (mcmcType, multilevel)
- static Method_mp_lit **MP2** (metropolisType, adaptive)
- static Method_mp_lit **MP2** (metropolisType, hastings)
- static Method_mp_lit **MP2** (mutationType, bit_random)
- static Method_mp_lit **MP2** (mutationType, offset_cauchy)
- static Method_mp_lit **MP2** (mutationType, offset_normal)
- static Method_mp_lit **MP2** (mutationType, offset_uniform)
- static Method_mp_lit **MP2** (mutationType, replace_uniform)
- static Method_mp_lit **MP2** (patternBasis, coordinate)
- static Method_mp_lit **MP2** (patternBasis, simplex)
- static Method_mp_lit **MP2** (pointReuse, all)
- static Method_mp_lit **MP2** (rejectionType, standard)
- static Method_mp_lit **MP2** (rejectionType, delayed)
- static Method_mp_lit **MP2** (reliabilityIntegration, first_order)
- static Method_mp_lit **MP2** (reliabilityIntegration, second_order)
- static Method_mp_lit **MP2** (replacementType, elitist)
- static Method_mp_lit **MP2** (replacementType, favor_feasible)
- static Method_mp_lit **MP2** (replacementType, roulette_wheel)
- static Method_mp_lit **MP2** (replacementType, unique_roulette_wheel)
- static Method_mp_lit **MP2** (rngName, mt19937)
- static Method_mp_lit **MP2** (rngName, rnum2)
- static Method_mp_lit **MP2** (searchMethod, gradient_based_line_search)
- static Method_mp_lit **MP2** (searchMethod, tr_pds)
- static Method_mp_lit **MP2** (searchMethod, trust_region)
- static Method_mp_lit **MP2** (searchMethod, value_based_line_search)
- static Method_mp_lit **MP2** (trialType, grid)
- static Method_mp_lit **MP2** (trialType, halton)
- static Method_mp_lit **MP2** (trialType, random)
- static Method_mp_litc **MP3** (crossoverType, crossoverRate, shuffle_random)
- static Method_mp_litc **MP3** (crossoverType, crossoverRate, null_crossover)
- static Method_mp_litc **MP3** (mutationType, mutationRate, null_mutation)

- static Method_mp_litc **MP3** (mutationType, mutationRate, offset_cauchy)
- static Method_mp_litc **MP3** (mutationType, mutationRate, offset_normal)
- static Method_mp_litc **MP3** (mutationType, mutationRate, offset_uniform)
- static Method_mp_litc **MP3** (replacementType, fitnessLimit, below_limit)
- static Method_mp_litrv **MP3** (nichingType, nicheVector, distance)
- static Method_mp_litrv **MP3** (nichingType, nicheVector, max_designs)
- static Method_mp_litrv **MP3** (nichingType, nicheVector, radial)
- static Method_mp_litrv **MP3** (postProcessorType, distanceVector, distance_postprocessor)
- static Method_mp_slit2 **MP3** (initializationType, flatFile, flat_file)
- static Method_mp_utype_lit **MP3s** (methodName, dlDetails, DL_SOLVER)
- static Real **MP_** (absConvTol)
- static Real **MP_** (centeringParam)
- static Real **MP_** (collocationRatio)
- static Real **MP_** (collocRatioTermsOrder)
- static Real **MP_** (constraintPenalty)
- static Real **MP_** (constrPenalty)
- static Real **MP_** (constraintTolerance)
- static Real **MP_** (contractFactor)
- static Real **MP_** (contractStepLength)
- static Real **MP_** (convergenceTolerance)
- static Real **MP_** (crossoverRate)
- static Real **MP_** (falseConvTol)
- static Real **MP_** (functionPrecision)
- static Real **MP_** (globalBalanceParam)
- static Real **MP_** (gradientTolerance)
- static Real **MP_** (hybridLSProb)
- static Real **MP_** (grThreshold)
- static Real **MP_** (initDelta)
- static Real **MP_** (initStepLength)
- static Real **MP_** (initTRRadius)
- static Real **MP_** (likelihoodScale)
- static Real **MP_** (lineSearchTolerance)
- static Real **MP_** (localBalanceParam)
- static Real **MP_** (maxBoxSize)
- static Real **MP_** (maxStep)
- static Real **MP_** (minBoxSize)
- static Real **MP_** (mutationRate)
- static Real **MP_** (mutationScale)
- static Real **MP_** (refinementRate)
- static Real **MP_** (regressionL2Penalty)
- static Real **MP_** (shrinkagePercent)
- static Real **MP_** (singConvTol)
- static Real **MP_** (singRadius)
- static Real **MP_** (smoothFactor)
- static Real **MP_** (solnTarget)
- static Real **MP_** (stepLenToBoundary)

- static Real **MP_** (surrBasedLocalTRContract)
- static Real **MP_** (surrBasedLocalTRContractTrigger)
- static Real **MP_** (surrBasedLocalTRExpand)
- static Real **MP_** (surrBasedLocalTRExpandTrigger)
- static Real **MP_** (surrBasedLocalTRInitSize)
- static Real **MP_** (surrBasedLocalTRMinSize)
- static Real **MP_** (threshDelta)
- static Real **MP_** (threshStepLength)
- static Real **MP_** (vbdDropTolerance)
- static Real **MP_** (volBoxSize)
- static Real **MP_** (vns)
- static Real **MP_** (xConvTol)
- static RealVector **MP_** (anisoDimPref)
- static RealVector **MP_** (concurrentParameterSets)
- static RealVector **MP_** (finalPoint)
- static RealVector **MP_** (linearEqConstraintCoeffs)
- static RealVector **MP_** (linearEqScales)
- static RealVector **MP_** (linearEqTargets)
- static RealVector **MP_** (linearIneqConstraintCoeffs)
- static RealVector **MP_** (linearIneqLowerBnds)
- static RealVector **MP_** (linearIneqUpperBnds)
- static RealVector **MP_** (linearIneqScales)
- static RealVector **MP_** (listOfPoints)
- static RealVector **MP_** (proposalCovScale)
- static RealVector **MP_** (regressionNoiseTol)
- static RealVector **MP_** (stepVector)
- static RealVectorArray **MP_** (genReliabilityLevels)
- static RealVectorArray **MP_** (probabilityLevels)
- static RealVectorArray **MP_** (reliabilityLevels)
- static RealVectorArray **MP_** (responseLevels)
- static unsigned short **MP_** (adaptedBasisAdvancements)
- static unsigned short **MP_** (cubIntOrder)
- static unsigned short **MP_** (softConvLimit)
- static unsigned short **MP_** (vbdOrder)
- static SizetArray **MP_** (collocationPoints)
- static SizetArray **MP_** (expansionSamples)
- static UShortArray **MP_** (expansionOrder)
- static UShortArray **MP_** (quadratureOrder)
- static UShortArray **MP_** (sparseGridLevel)
- static UShortArray **MP_** (tensorGridOrder)
- static UShortArray **MP_** (varPartitions)
- static String **MP_** (approxExportFile)
- static String **MP_** (approxImportFile)
- static String **MP_** (betaSolverName)
- static String **MP_** (displayFormat)
- static String **MP_** (expansionExportFile)

- static String **MP**_ (expansionImportFile)
- static String **MP**_ (historyFile)
- static String **MP**_ (hybridGlobalMethodName)
- static String **MP**_ (hybridGlobalMethodPointer)
- static String **MP**_ (hybridGlobalModelPointer)
- static String **MP**_ (hybridLocalMethodName)
- static String **MP**_ (hybridLocalMethodPointer)
- static String **MP**_ (hybridLocalModelPointer)
- static String **MP**_ (idMethod)
- static String **MP**_ (logFile)
- static String **MP**_ (modelPointer)
- static String **MP**_ (pstudyFilename)
- static String **MP**_ (subMethodName)
- static String **MP**_ (subMethodPointer)
- static String **MP**_ (subModelPointer)
- static StringArray **MP**_ (hybridMethodNames)
- static StringArray **MP**_ (hybridMethodPointers)
- static StringArray **MP**_ (hybridModelPointers)
- static StringArray **MP**_ (linearEqScaleTypes)
- static StringArray **MP**_ (linearIneqScaleTypes)
- static StringArray **MP**_ (miscOptions)
- static bool **MP**_ (approxExportAnnotated)
- static bool **MP**_ (approxImportActive)
- static bool **MP**_ (approxImportAnnotated)
- static bool **MP**_ (backfillFlag)
- static bool **MP**_ (calibrateSigmaFlag)
- static bool **MP**_ (constantPenalty)
- static bool **MP**_ (crossValidation)
- static bool **MP**_ (expansionFlag)
- static bool **MP**_ (fixedSeedFlag)
- static bool **MP**_ (fixedSequenceFlag)
- static bool **MP**_ (latinizeFlag)
- static bool **MP**_ (mainEffectsFlag)
- static bool **MP**_ (methodScaling)
- static bool **MP**_ (methodUseDerivsFlag)
- static bool **MP**_ (mutationAdaptive)
- static bool **MP**_ (normalizedCoeffs)
- static bool **MP**_ (printPopFlag)
- static bool **MP**_ (pstudyFileActive)
- static bool **MP**_ (pstudyFileAnnotated)
- static bool **MP**_ (randomizeOrderFlag)
- static bool **MP**_ (regressDiag)
- static bool **MP**_ (showAllEval)
- static bool **MP**_ (showMiscOptions)
- static bool **MP**_ (speculativeFlag)
- static bool **MP**_ (tensorGridFlag)

- static bool **MP_** (surrBasedGlobalReplacePts)
- static bool **MP_** (surrBasedLocalLayerBypass)
- static bool **MP_** (vbdFlag)
- static bool **MP_** (volQualityFlag)
- static short **MP_** (expansionType)
- static short **MP_** (nestingOverride)
- static short **MP_** (refinementType)
- static int **MP_** (batchSize)
- static int **MP_** (concurrentRandomJobs)
- static int **MP_** (contractAfterFail)
- static int **MP_** (covarianceType)
- static int **MP_** (crossoverChainPairs)
- static int **MP_** (emulatorOrder)
- static int **MP_** (emulatorSamples)
- static int **MP_** (expandAfterSuccess)
- static int **MP_** (iteratorServers)
- static int **MP_** (jumpStep)
- static int **MP_** (maxFunctionEvaluations)
- static int **MP_** (maxIterations)
- static int **MP_** (mutationRange)
- static int **MP_** (newSolnsGenerated)
- static int **MP_** (numChains)
- static int **MP_** (numCR)
- static int **MP_** (numSamples)
- static int **MP_** (numSteps)
- static int **MP_** (numSymbols)
- static int **MP_** (numTrials)
- static int **MP_** (populationSize)
- static int **MP_** (previousSamples)
- static int **MP_** (procsPerIterator)
- static int **MP_** (randomSeed)
- static int **MP_** (refineSamples)
- static int **MP_** (searchSchemeSize)
- static int **MP_** (totalPatternSize)
- static int **MP_** (verifyLevel)
- static size_t **MP_** (numDesigns)
- static size_t **MP_** (numFinalSolutions)
- static size_t **MP_** (numGenerations)
- static size_t **MP_** (numOffspring)
- static size_t **MP_** (numParents)
- static Method_mp_type **MP2s** (covarianceControl, DIAGONAL_COVARIANCE)
- static Method_mp_type **MP2s** (covarianceControl, FULL_COVARIANCE)
- static Method_mp_type **MP2s** (distributionType, COMPLEMENTARY)
- static Method_mp_type **MP2s** (distributionType, CUMULATIVE)
- static Method_mp_type **MP2s** (emulatorType, GP_EMULATOR)
- static Method_mp_type **MP2s** (emulatorType, KRIGING_EMULATOR)

- static Method_mp_type **MP2s** (emulatorType, PCE_EMULATOR)
- static Method_mp_type **MP2s** (emulatorType, SC_EMULATOR)
- static Method_mp_type **MP2s** (emulatorType, VPS_EMULATOR)
- static Method_mp_type **MP2p** (expansionBasisType, ADAPTED_BASIS_EXPANDING_FRONT)
- static Method_mp_type **MP2p** (expansionBasisType, ADAPTED_BASIS_GENERALIZED)
- static Method_mp_type **MP2p** (expansionBasisType, HIERARCHICAL_INTERPOLANT)
- static Method_mp_type **MP2p** (expansionBasisType, NODAL_INTERPOLANT)
- static Method_mp_type **MP2p** (expansionBasisType, TENSOR_PRODUCT_BASIS)
- static Method_mp_type **MP2p** (expansionBasisType, TOTAL_ORDER_BASIS)
- static Method_mp_type **MP2s** (expansionType, ASKEY_U)
- static Method_mp_type **MP2s** (expansionType, STD_NORMAL_U)
- static Method_mp_type **MP2p** (growthOverride, RESTRICTED)
- static Method_mp_type **MP2p** (growthOverride, UNRESTRICTED)
- static Method_mp_type **MP2s** (iteratorScheduling, MASTER_SCHEDULING)
- static Method_mp_type **MP2s** (iteratorScheduling, PEER_SCHEDULING)
- static Method_mp_type **MP2s** (lsRegressionType, EQ_CON_LS)
- static Method_mp_type **MP2s** (lsRegressionType, SVD_LS)
- static Method_mp_type **MP2o** (meritFn, ArgaezTapia)
- static Method_mp_type **MP2o** (meritFn, NormFmu)
- static Method_mp_type **MP2o** (meritFn, VanShanno)
- static Method_mp_type **MP2s** (methodOutput, DEBUG_OUTPUT)
- static Method_mp_type **MP2s** (methodOutput, NORMAL_OUTPUT)
- static Method_mp_type **MP2s** (methodOutput, QUIET_OUTPUT)
- static Method_mp_type **MP2s** (methodOutput, SILENT_OUTPUT)
- static Method_mp_type **MP2s** (methodOutput, VERBOSE_OUTPUT)
- static Method_mp_type **MP2p** (nestingOverride, NESTED)
- static Method_mp_type **MP2p** (nestingOverride, NON_NESTED)
- static Method_mp_type **MP2p** (refinementControl, DIMENSION_ADAPTIVE_CONTROL_GENERALIZED)
- static Method_mp_type **MP2p** (refinementControl, DIMENSION_ADAPTIVE_CONTROL_DECAY)
- static Method_mp_type **MP2p** (refinementControl, DIMENSION_ADAPTIVE_CONTROL_SOBOL)
- static Method_mp_type **MP2p** (refinementControl, LOCAL_ADAPTIVE_CONTROL)
- static Method_mp_type **MP2p** (refinementControl, UNIFORM_CONTROL)
- static Method_mp_type **MP2p** (refinementType, P_REFINEMENT)
- static Method_mp_type **MP2p** (refinementType, H_REFINEMENT)
- static Method_mp_type **MP2p** (regressionType, BASIS_PURSUIT)
- static Method_mp_type **MP2p** (regressionType, BASIS_PURSUIT_DENOISING)
- static Method_mp_type **MP2p** (regressionType, DEFAULT_LEAST_SQ_REGRESSION)
- static Method_mp_type **MP2p** (regressionType, LASSO_REGRESSION)
- static Method_mp_type **MP2p** (regressionType, LEAST_ANGLE_REGRESSION)
- static Method_mp_type **MP2p** (regressionType, ORTHOG_LEAST_INTERPOLATION)
- static Method_mp_type **MP2p** (regressionType, ORTHOG_MATCH_PURSUIT)
- static Method_mp_type **MP2s** (responseLevelTarget, GEN_RELIABILITIES)
- static Method_mp_type **MP2s** (responseLevelTarget, PROBABILITIES)
- static Method_mp_type **MP2s** (responseLevelTarget, RELIABILITIES)
- static Method_mp_type **MP2s** (responseLevelTargetReduce, SYSTEM_PARALLEL)

- static Method_mp_type **MP2s** (responseLevelTargetReduce, SYSTEM_SERIES)
- static Method_mp_type **MP2s** (surrBasedLocalAcceptLogic, FILTER)
- static Method_mp_type **MP2s** (surrBasedLocalAcceptLogic, TR_RATIO)
- static Method_mp_type **MP2s** (surrBasedLocalConstrRelax, HOMOTOPY)
- static Method_mp_type **MP2s** (surrBasedLocalMeritFn, ADAPTIVE_PENALTY_MERIT)
- static Method_mp_type **MP2s** (surrBasedLocalMeritFn, AUGMENTED_LAGRANGIAN_MERIT)
- static Method_mp_type **MP2s** (surrBasedLocalMeritFn, LAGRANGIAN_MERIT)
- static Method_mp_type **MP2s** (surrBasedLocalMeritFn, PENALTY_MERIT)
- static Method_mp_type **MP2s** (surrBasedLocalSubProbCon, LINEARIZED_CONSTRAINTS)
- static Method_mp_type **MP2s** (surrBasedLocalSubProbCon, NO_CONSTRAINTS)
- static Method_mp_type **MP2s** (surrBasedLocalSubProbCon, ORIGINAL_CONSTRAINTS)
- static Method_mp_type **MP2s** (surrBasedLocalSubProbObj, AUGMENTED_LAGRANGIAN_OBJECTIVE)
- static Method_mp_type **MP2s** (surrBasedLocalSubProbObj, LAGRANGIAN_OBJECTIVE)
- static Method_mp_type **MP2s** (surrBasedLocalSubProbObj, ORIGINAL_PRIMARY)
- static Method_mp_type **MP2s** (surrBasedLocalSubProbObj, SINGLE_OBJECTIVE)
- static Method_mp_utype **MP2s** (integrationRefine, AIS)
- static Method_mp_utype **MP2s** (integrationRefine, IS)
- static Method_mp_utype **MP2s** (integrationRefine, MMAIS)
- static Method_mp_utype **MP2s** (methodName, ASYNCH_PATTERN_SEARCH)
- static Method_mp_utype **MP2s** (methodName, COLINY_BETA)
- static Method_mp_utype **MP2s** (methodName, COLINY_COBYLA)
- static Method_mp_utype **MP2s** (methodName, COLINY_DIRECT)
- static Method_mp_utype **MP2s** (methodName, COLINY_EA)
- static Method_mp_utype **MP2s** (methodName, COLINY_PATTERN_SEARCH)
- static Method_mp_utype **MP2s** (methodName, COLINY_SOLIS_WETS)
- static Method_mp_utype **MP2s** (methodName, CONMIN_FRCG)
- static Method_mp_utype **MP2s** (methodName, CONMIN_MFD)
- static Method_mp_utype **MP2s** (methodName, DACE)
- static Method_mp_utype **MP2s** (methodName, DOT_BFGS)
- static Method_mp_utype **MP2s** (methodName, DOT_FRCG)
- static Method_mp_utype **MP2s** (methodName, DOT_MMFD)
- static Method_mp_utype **MP2s** (methodName, DOT_SLP)
- static Method_mp_utype **MP2s** (methodName, DOT_SQP)
- static Method_mp_utype **MP2s** (methodName, EFFICIENT_GLOBAL)
- static Method_mp_utype **MP2s** (methodName, FSU_CVT)
- static Method_mp_utype **MP2s** (methodName, FSU_HALTON)
- static Method_mp_utype **MP2s** (methodName, FSU_HAMMERSLEY)
- static Method_mp_utype **MP2s** (methodName, HYBRID)
- static Method_mp_utype **MP2s** (methodName, MESH_ADAPTIVE_SEARCH)
- static Method_mp_utype **MP2s** (methodName, MOGA)
- static Method_mp_utype **MP2s** (methodName, MULTI_START)
- static Method_mp_utype **MP2s** (methodName, NCSU_DIRECT)
- static Method_mp_utype **MP2s** (methodName, NL2SOL)
- static Method_mp_utype **MP2s** (methodName, NLPQL_SQP)
- static Method_mp_utype **MP2s** (methodName, NLSSOL_SQP)

- static Method_mp_utype **MP2s** (methodName, ADAPTIVE_SAMPLING)
- static Method_mp_utype **MP2s** (methodName, BAYES_CALIBRATION)
- static Method_mp_utype **MP2s** (methodName, EFFICIENT_SUBSPACE)
- static Method_mp_utype **MP2s** (methodName, GENIE_DIRECT)
- static Method_mp_utype **MP2s** (methodName, GENIE_OPT_DARTS)
- static Method_mp_utype **MP2s** (methodName, GPAIS)
- static Method_mp_utype **MP2s** (methodName, GLOBAL_EVIDENCE)
- static Method_mp_utype **MP2s** (methodName, GLOBAL_INTERVAL_EST)
- static Method_mp_utype **MP2s** (methodName, GLOBAL_RELIABILITY)
- static Method_mp_utype **MP2s** (methodName, IMPORTANCE_SAMPLING)
- static Method_mp_utype **MP2s** (methodName, LOCAL_EVIDENCE)
- static Method_mp_utype **MP2s** (methodName, LOCAL_INTERVAL_EST)
- static Method_mp_utype **MP2s** (methodName, LOCAL_RELIABILITY)
- static Method_mp_utype **MP2s** (methodName, POF_DARTS)
- static Method_mp_utype **MP2s** (methodName, POLYNOMIAL_CHAOS)
- static Method_mp_utype **MP2s** (methodName, RANDOM_SAMPLING)
- static Method_mp_utype **MP2s** (methodName, STOCH_COLLOCATION)
- static Method_mp_utype **MP2s** (methodName, NONLINEAR_CG)
- static Method_mp_utype **MP2s** (methodName, NPSOL_SQP)
- static Method_mp_utype **MP2s** (methodName, OPTPP_CG)
- static Method_mp_utype **MP2s** (methodName, OPTPP_FD_NEWTON)
- static Method_mp_utype **MP2s** (methodName, OPTPP_G_NEWTON)
- static Method_mp_utype **MP2s** (methodName, OPTPP_NEWTON)
- static Method_mp_utype **MP2s** (methodName, OPTPP_PDS)
- static Method_mp_utype **MP2s** (methodName, OPTPP_Q_NEWTON)
- static Method_mp_utype **MP2s** (methodName, PARETO_SET)
- static Method_mp_utype **MP2s** (methodName, PSUADE_MOAT)
- static Method_mp_utype **MP2s** (methodName, RICHARDSON_EXTRAP)
- static Method_mp_utype **MP2s** (methodName, SOGA)
- static Method_mp_utype **MP2s** (methodName, SURROGATE_BASED_GLOBAL)
- static Method_mp_utype **MP2s** (methodName, SURROGATE_BASED_LOCAL)
- static Method_mp_utype **MP2s** (methodName, VECTOR_PARAMETER_STUDY)
- static Method_mp_utype **MP2s** (methodName, LIST_PARAMETER_STUDY)
- static Method_mp_utype **MP2s** (methodName, CENTERED_PARAMETER_STUDY)
- static Method_mp_utype **MP2s** (methodName, MULTIDIM_PARAMETER_STUDY)
- static Method_mp_utype **MP2s** (reliabilitySearchType, AMV_PLUS_U)
- static Method_mp_utype **MP2s** (reliabilitySearchType, AMV_PLUS_X)
- static Method_mp_utype **MP2s** (reliabilitySearchType, AMV_U)
- static Method_mp_utype **MP2s** (reliabilitySearchType, AMV_X)
- static Method_mp_utype **MP2s** (reliabilitySearchType, EGRA_U)
- static Method_mp_utype **MP2s** (reliabilitySearchType, EGRA_X)
- static Method_mp_utype **MP2s** (reliabilitySearchType, NO_APPROX)
- static Method_mp_utype **MP2s** (reliabilitySearchType, TANA_U)
- static Method_mp_utype **MP2s** (reliabilitySearchType, TANA_X)
- static Method_mp_utype **MP2s** (sampleType, SUBMETHOD_INCREMENTAL_LHS)
- static Method_mp_utype **MP2s** (sampleType, SUBMETHOD_INCREMENTAL_RANDOM)

- static Method_mp_utype **MP2s** (sampleType, SUBMETHOD_LHS)
- static Method_mp_utype **MP2s** (sampleType, SUBMETHOD_RANDOM)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_COLLABORATIVE)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_EMBEDDED)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_SEQUENTIAL)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_DREAM)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_GPMSA)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_QUESO)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_NIP)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_SQP)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_EA)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_EGO)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_SBO)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_LHS)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_RANDOM)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_OA_LHS)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_OAS)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_BOX_BEHNKEN)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_CENTRAL_COMPOSITE)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_GRID)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_CONVERGE_ORDER)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_CONVERGE_QOI)
- static Method_mp_utype **MP2s** (subMethod, SUBMETHOD_ESTIMATE_ORDER)
- static IntSet **MP_** (surrogateFnIndices)
- static Model_mp_lit **MP2** (approxPointReuse, all)
- static Model_mp_lit **MP2** (approxPointReuse, none)
- static Model_mp_lit **MP2** (approxPointReuse, region)
- static Model_mp_lit **MP2** (marsInterpolation, linear)
- static Model_mp_lit **MP2** (marsInterpolation, cubic)
- static Model_mp_lit **MP2** (modelType, nested)
- static Model_mp_lit **MP2** (modelType, single)
- static Model_mp_lit **MP2** (modelType, surrogate)
- static Model_mp_lit **MP2** (surrogateType, hierarchical)
- static Model_mp_lit **MP2** (surrogateType, global_gaussian)
- static Model_mp_lit **MP2** (surrogateType, global_kriging)
- static Model_mp_lit **MP2** (surrogateType, global_mars)
- static Model_mp_lit **MP2** (surrogateType, global_moving_least_squares)
- static Model_mp_lit **MP2** (surrogateType, global_neural_network)
- static Model_mp_lit **MP2** (surrogateType, global_polynomial)
- static Model_mp_lit **MP2** (surrogateType, global_radial_basis)
- static Model_mp_lit **MP2** (surrogateType, local_taylor)
- static Model_mp_lit **MP2** (surrogateType, multipoint_tana)
- static Model_mp_lit **MP2** (trendOrder, constant)
- static Model_mp_lit **MP2** (trendOrder, linear)
- static Model_mp_lit **MP2** (trendOrder, reduced_quadratic)
- static Model_mp_lit **MP2** (trendOrder, quadratic)

- static Model_mp_ord **MP2s** (approxCorrectionOrder, 0)
- static Model_mp_ord **MP2s** (approxCorrectionOrder, 1)
- static Model_mp_ord **MP2s** (approxCorrectionOrder, 2)
- static Model_mp_ord **MP2s** (polynomialOrder, 1)
- static Model_mp_ord **MP2s** (polynomialOrder, 2)
- static Model_mp_ord **MP2s** (polynomialOrder, 3)
- static Model_mp_type **MP2s** (approxCorrectionType, ADDITIVE_CORRECTION)
- static Model_mp_type **MP2s** (approxCorrectionType, COMBINED_CORRECTION)
- static Model_mp_type **MP2s** (approxCorrectionType, MULTIPLICATIVE_CORRECTION)
- static Model_mp_type **MP2s** (pointsManagement, MINIMUM_POINTS)
- static Model_mp_type **MP2s** (pointsManagement, RECOMMENDED_POINTS)
- static Model_mp_type **MP2s** (subMethodScheduling, MASTER_SCHEDULING)
- static Model_mp_type **MP2s** (subMethodScheduling, PEER_SCHEDULING)
- static Real **MP_** (annRange)
- static Real **MP_** (krigingNugget)
- static Real **MP_** (percentFold)
- static RealVector **MP_** (krigingCorrelations)
- static RealVector **MP_** (primaryRespCoeffs)
- static RealVector **MP_** (secondaryRespCoeffs)
- static String **MP_** (approxChallengeFile)
- static String **MP_** (approxExportModelFile)
- static String **MP_** (idModel)
- static String **MP_** (interfacePointer)
- static String **MP_** (krigingOptMethod)
- static String **MP_** (lowFidelityModelPointer)
- static String **MP_** (optionalInterfRespPointer)
- static String **MP_** (responsesPointer)
- static String **MP_** (truthModelPointer)
- static String **MP_** (variablesPointer)
- static StringArray **MP_** (primaryVarMaps)
- static StringArray **MP_** (secondaryVarMaps)
- static StringArray **MP_** (diagMetrics)
- static bool **MP_** (approxChallengeActive)
- static bool **MP_** (approxChallengeAnnotated)
- static bool **MP_** (crossValidateFlag)
- static bool **MP_** (hierarchicalTags)
- static bool **MP_** (modelUseDerivsFlag)
- static bool **MP_** (pressFlag)
- static bool **MP_** (pointSelection)
- static short **MP_** (annNodes)
- static short **MP_** (annRandomWeight)
- static short **MP_** (krigingFindNugget)
- static short **MP_** (krigingMaxTrials)
- static short **MP_** (marsMaxBases)
- static short **MP_** (mlsPolyOrder)
- static short **MP_** (mlsWeightFunction)

- static short **MP_** (rbfBases)
- static short **MP_** (rbfMaxPts)
- static short **MP_** (rbfMaxSubsets)
- static short **MP_** (rbfMinPartition)
- static int **MP_** (numFolds)
- static int **MP_** (pointsTotal)
- static int **MP_** (subMethodProcs)
- static int **MP_** (subMethodServers)
- static IntSet **MP_** (idAnalyticGrads)
- static IntSet **MP_** (idAnalyticHessians)
- static IntSet **MP_** (idNumericalGrads)
- static IntSet **MP_** (idNumericalHessians)
- static IntSet **MP_** (idQuasiHessians)
- static IntVector **MP_** (fieldLengths)
- static IntVector **MP_** (numCoordsPerField)
- static RealVector **MP_** (coordsPerField)
- static RealVector **MP_** (expConfigVars)
- static RealVector **MP_** (expObservations)
- static RealVector **MP_** (expStdDeviations)
- static RealVector **MP_** (primaryRespFnWeights)
- static RealVector **MP_** (nonlinearEqTargets)
- static RealVector **MP_** (nonlinearIneqLowerBnds)
- static RealVector **MP_** (nonlinearIneqUpperBnds)
- static RealVector **MP_** (fdGradStepSize)
- static RealVector **MP_** (fdHessStepSize)
- static RealVector **MP_** (primaryRespFnScales)
- static RealVector **MP_** (nonlinearEqScales)
- static RealVector **MP_** (nonlinearIneqScales)
- static Resp_mp_lit **MP2** (gradientType, analytic)
- static Resp_mp_lit **MP2** (gradientType, mixed)
- static Resp_mp_lit **MP2** (gradientType, none)
- static Resp_mp_lit **MP2** (gradientType, numerical)
- static Resp_mp_lit **MP2** (hessianType, analytic)
- static Resp_mp_lit **MP2** (hessianType, mixed)
- static Resp_mp_lit **MP2** (hessianType, none)
- static Resp_mp_lit **MP2** (hessianType, numerical)
- static Resp_mp_lit **MP2** (hessianType, quasi)
- static Resp_mp_lit **MP2** (intervalType, central)
- static Resp_mp_lit **MP2** (intervalType, forward)
- static Resp_mp_lit **MP2** (methodSource, dakota)
- static Resp_mp_lit **MP2** (methodSource, vendor)
- static Resp_mp_lit **MP2** (fdGradStepType, absolute)
- static Resp_mp_lit **MP2** (fdGradStepType, bounds)
- static Resp_mp_lit **MP2** (fdGradStepType, relative)
- static Resp_mp_lit **MP2** (fdHessStepType, absolute)
- static Resp_mp_lit **MP2** (fdHessStepType, bounds)

- static Resp_mp_lit **MP2** (fdHessStepType, relative)
- static Resp_mp_lit **MP2** (quasiHessianType, bfgs)
- static Resp_mp_lit **MP2** (quasiHessianType, damped_bfgs)
- static Resp_mp_lit **MP2** (quasiHessianType, sr1)
- static String **MP_** (configDataFileName)
- static String **MP_** (coordDataFileName)
- static String **MP_** (expDataFileName)
- static String **MP_** (fieldCoordDataFileName)
- static String **MP_** (fieldDataFileName)
- static String **MP_** (idResponses)
- static String **MP_** (sigmaDataFileName)
- static StringArray **MP_** (nonlinearEqScaleTypes)
- static StringArray **MP_** (nonlinearIneqScaleTypes)
- static StringArray **MP_** (primaryRespFnScaleTypes)
- static StringArray **MP_** (primaryRespFnSense)
- static StringArray **MP_** (responseLabels)
- static StringArray **MP_** (sigmaType)
- static bool **MP_** (centralHess)
- static bool **MP_** (expDataFileAnnotated)
- static bool **MP_** (ignoreBounds)
- static size_t **MP_** (numExpStdDeviations)
- static size_t **MP_** (numExpConfigVars)
- static size_t **MP_** (numExperiments)
- static size_t **MP_** (numFieldLeastSqTerms)
- static size_t **MP_** (numFieldObjectiveFunctions)
- static size_t **MP_** (numFieldResponseFunctions)
- static size_t **MP_** (numLeastSqTerms)
- static size_t **MP_** (numNonlinearEqConstraints)
- static size_t **MP_** (numNonlinearIneqConstraints)
- static size_t **MP_** (numObjectiveFunctions)
- static size_t **MP_** (numResponseFunctions)
- static size_t **MP_** (numScalarLeastSqTerms)
- static size_t **MP_** (numScalarObjectiveFunctions)
- static size_t **MP_** (numScalarResponseFunctions)
- static String **MP_** (errorFile)
- static String **MP_** (outputFile)
- static String **MP_** (postRunInput)
- static String **MP_** (postRunOutput)
- static String **MP_** (preRunInput)
- static String **MP_** (preRunOutput)
- static String **MP_** (readRestart)
- static String **MP_** (resultsOutputFile)
- static String **MP_** (runInput)
- static String **MP_** (runOutput)
- static String **MP_** (tabularDataFile)
- static String **MP_** (topMethodPointer)

- static String **MP_** (writeRestart)
- static bool **MP_** (checkFlag)
- static bool **MP_** (graphicsFlag)
- static bool **MP_** (postRunFlag)
- static bool **MP_** (preRunFlag)
- static bool **MP_** (resultsOutputFlag)
- static bool **MP_** (runFlag)
- static bool **MP_** (tabularDataFlag)
- static int **MP_** (outputPrecision)
- static int **MP_** (stopRestart)
- static size_t **MP_** (numBetaUncVars)
- static size_t **MP_** (numBinomialUncVars)
- static size_t **MP_** (numContinuousDesVars)
- static size_t **MP_** (numContinuousIntervalUncVars)
- static size_t **MP_** (numContinuousStateVars)
- static size_t **MP_** (numDiscreteDesRangeVars)
- static size_t **MP_** (numDiscreteDesSetIntVars)
- static size_t **MP_** (numDiscreteDesSetStrVars)
- static size_t **MP_** (numDiscreteDesSetRealVars)
- static size_t **MP_** (numDiscreteIntervalUncVars)
- static size_t **MP_** (numDiscreteStateRangeVars)
- static size_t **MP_** (numDiscreteStateSetIntVars)
- static size_t **MP_** (numDiscreteStateSetStrVars)
- static size_t **MP_** (numDiscreteStateSetRealVars)
- static size_t **MP_** (numDiscreteUncSetIntVars)
- static size_t **MP_** (numDiscreteUncSetStrVars)
- static size_t **MP_** (numDiscreteUncSetRealVars)
- static size_t **MP_** (numExponentialUncVars)
- static size_t **MP_** (numFrechetUncVars)
- static size_t **MP_** (numGammaUncVars)
- static size_t **MP_** (numGeometricUncVars)
- static size_t **MP_** (numGumbelUncVars)
- static size_t **MP_** (numHistogramBinUncVars)
- static size_t **MP_** (numHistogramPtIntUncVars)
- static size_t **MP_** (numHistogramPtStrUncVars)
- static size_t **MP_** (numHistogramPtRealUncVars)
- static size_t **MP_** (numHyperGeomUncVars)
- static size_t **MP_** (numLognormalUncVars)
- static size_t **MP_** (numLoguniformUncVars)
- static size_t **MP_** (numNegBinomialUncVars)
- static size_t **MP_** (numNormalUncVars)
- static size_t **MP_** (numPoissonUncVars)
- static size_t **MP_** (numTriangularUncVars)
- static size_t **MP_** (numUniformUncVars)
- static size_t **MP_** (numWeibullUncVars)
- static IntVector **VP_** (ddsi)

- static IntVector **VP**_ (DIlb)
- static IntVector **MP**_ (discreteDesignRangeLowerBnds)
- static IntVector **MP**_ (discreteDesignRangeUpperBnds)
- static IntVector **MP**_ (discreteDesignRangeVars)
- static IntVector **MP**_ (discreteDesignSetIntVars)
- static IntVector **MP**_ (discreteIntervalUncVars)
- static IntVector **MP**_ (discreteStateRangeLowerBnds)
- static IntVector **MP**_ (discreteStateRangeUpperBnds)
- static IntVector **MP**_ (discreteStateRangeVars)
- static IntVector **MP**_ (discreteStateSetIntVars)
- static IntVector **MP**_ (discreteUncSetIntVars)
- static IntVector **VP**_ (DIub)
- static IntVector **MP**_ (histogramPointIntUncVars)
- static IntVector **VP**_ (hpia)
- static IntVector **VP**_ (dssi)
- static IntVector **VP**_ (dusi)
- static IntArray **VP**_ (nddsi)
- static IntArray **VP**_ (nddss)
- static IntArray **VP**_ (nddsr)
- static IntArray **VP**_ (ndssi)
- static IntArray **VP**_ (ndsss)
- static IntArray **VP**_ (ndssr)
- static IntArray **VP**_ (ndusi)
- static IntArray **VP**_ (nduss)
- static IntArray **VP**_ (ndusr)
- static IntArray **VP**_ (nhbp)
- static IntArray **VP**_ (nhpip)
- static IntArray **VP**_ (nhpsp)
- static IntArray **VP**_ (nhprp)
- static IntArray **VP**_ (nCI)
- static IntArray **VP**_ (nDI)
- static RealVector **MP**_ (betaUncLowerBnds)
- static RealVector **MP**_ (betaUncUpperBnds)
- static RealVector **MP**_ (betaUncVars)
- static RealVector **MP**_ (binomialUncProbPerTrial)
- static RealVector **MP**_ (continuousDesignLowerBnds)
- static RealVector **MP**_ (continuousDesignUpperBnds)
- static RealVector **MP**_ (continuousDesignVars)
- static RealVector **MP**_ (continuousDesignScales)
- static RealVector **MP**_ (continuousIntervalUncVars)
- static RealVector **MP**_ (continuousStateLowerBnds)
- static RealVector **MP**_ (continuousStateUpperBnds)
- static RealVector **MP**_ (continuousStateVars)
- static RealVector **MP**_ (discreteDesignSetRealVars)
- static RealVector **MP**_ (discreteStateSetRealVars)
- static RealVector **MP**_ (discreteUncSetRealVars)

- static RealVector **MP_** (frechetUncBetas)
- static RealVector **MP_** (frechetUncVars)
- static RealVector **MP_** (geometricUncProbPerTrial)
- static RealVector **MP_** (gumbelUncBetas)
- static RealVector **MP_** (gumbelUncVars)
- static RealVector **MP_** (histogramBinUncVars)
- static RealVector **MP_** (histogramPointRealUncVars)
- static RealVector **MP_** (negBinomialUncProbPerTrial)
- static RealVector **MP_** (normalUncLowerBnds)
- static RealVector **MP_** (normalUncMeans)
- static RealVector **MP_** (normalUncUpperBnds)
- static RealVector **MP_** (normalUncVars)
- static RealVector **MP_** (triangularUncModes)
- static RealVector **MP_** (triangularUncVars)
- static RealVector **MP_** (uniformUncVars)
- static RealVector **MP_** (weibullUncVars)
- static RealVector **VP_** (ddsr)
- static RealVector **VP_** (dssr)
- static RealVector **VP_** (dusr)
- static RealVector **VP_** (CIlb)
- static RealVector **VP_** (CIub)
- static RealVector **VP_** (CIp)
- static RealVector **VP_** (DIp)
- static RealVector **VP_** (DSIp)
- static RealVector **VP_** (DSSp)
- static RealVector **VP_** (DSRp)
- static RealVector **VP_** (hba)
- static RealVector **VP_** (hbo)
- static RealVector **VP_** (hbc)
- static RealVector **VP_** (hpic)
- static RealVector **VP_** (hpsc)
- static RealVector **VP_** (hpra)
- static RealVector **VP_** (hprc)
- static RealVector **VP_** (ucm)
- static String **MP_** (idVariables)
- static StringArray **MP_** (continuousDesignLabels)
- static StringArray **MP_** (continuousDesignScaleTypes)
- static StringArray **MP_** (continuousStateLabels)
- static StringArray **MP_** (discreteDesignRangeLabels)
- static StringArray **MP_** (discreteDesignSetIntLabels)
- static StringArray **MP_** (discreteDesignSetStrLabels)
- static StringArray **MP_** (discreteDesignSetRealLabels)
- static StringArray **MP_** (discreteStateRangeLabels)
- static StringArray **MP_** (discreteStateSetIntLabels)
- static StringArray **MP_** (discreteStateSetStrLabels)
- static StringArray **MP_** (discreteStateSetRealLabels)

- static StringArray **MP_** (discreteDesignSetStrVars)
- static StringArray **MP_** (discreteUncSetStrVars)
- static StringArray **MP_** (discreteStateSetStrVars)
- static StringArray **MP_** (histogramPointStrUncVars)
- static StringArray **VP_** (hpsa)
- static StringArray **VP_** (ddss)
- static StringArray **VP_** (duss)
- static StringArray **VP_** (dsss)
- static BitArray **MP_** (discreteDesignSetIntCat)
- static BitArray **MP_** (discreteDesignSetRealCat)
- static BitArray **MP_** (discreteStateSetIntCat)
- static BitArray **MP_** (discreteStateSetRealCat)
- static BitArray **MP_** (discreteUncSetIntCat)
- static BitArray **MP_** (discreteUncSetRealCat)
- static Var_brv **MP2s** (betaUncAlphas, 0.)
- static Var_brv **MP2s** (betaUncBetas, 0.)
- static Var_brv **MP2s** (exponentialUncBetas, 0.)
- static Var_brv **MP2s** (exponentialUncVars, 0.)
- static Var_brv **MP2s** (frechetUncAlphas, 2.)
- static Var_brv **MP2s** (gammaUncAlphas, 0.)
- static Var_brv **MP2s** (gammaUncBetas, 0.)
- static Var_brv **MP2s** (gammaUncVars, 0.)
- static Var_brv **MP2s** (gumbelUncAlphas, 0.)
- static Var_brv **MP2s** (lognormalUncErrFacts, 1.)
- static Var_brv **MP2s** (lognormalUncLambdas, 0.)
- static Var_brv **MP2s** (lognormalUncLowerBnds, 0.)
- static Var_brv **MP2s** (lognormalUncMeans, 0.)
- static Var_brv **MP2s** (lognormalUncStdDevs, 0.)
- static Var_brv **MP2s** (lognormalUncUpperBnds, DBL_MAX)
- static Var_brv **MP2s** (lognormalUncVars, 0.)
- static Var_brv **MP2s** (lognormalUncZetas, 0.)
- static Var_brv **MP2s** (loguniformUncLowerBnds, 0.)
- static Var_brv **MP2s** (loguniformUncUpperBnds, DBL_MAX)
- static Var_brv **MP2s** (loguniformUncVars, 0.)
- static Var_brv **MP2s** (normalUncStdDevs, 0.)
- static Var_brv **MP2s** (poissonUncLambdas, 0.)
- static Var_brv **MP2s** (triangularUncLowerBnds,-DBL_MAX)
- static Var_brv **MP2s** (triangularUncUpperBnds, DBL_MAX)
- static Var_brv **MP2s** (uniformUncLowerBnds,-DBL_MAX)
- static Var_brv **MP2s** (uniformUncUpperBnds, DBL_MAX)
- static Var_brv **MP2s** (weibullUncAlphas, 0.)
- static Var_brv **MP2s** (weibullUncBetas, 0.)
- static Var_biv **MP2s** (binomialUncNumTrials, 0)
- static Var_biv **MP2s** (binomialUncVars, 0)
- static Var_biv **MP2s** (geometricUncVars, 0)
- static Var_biv **MP2s** (hyperGeomUncNumDrawn, 0)

- static Var_biv **MP2s** (hyperGeomUncSelectedPop, 0)
- static Var_biv **MP2s** (hyperGeomUncTotalPop, 0)
- static Var_biv **MP2s** (hyperGeomUncVars, 0)
- static Var_biv **MP2s** (negBinomialUncNumTrials, 0)
- static Var_biv **MP2s** (negBinomialUncVars, 0)
- static Var_biv **MP2s** (poissonUncVars, 0)
- static Var_mp_type **Vtype** (varsDomain, MIXED_DOMAIN)
- static Var_mp_type **Vtype** (varsDomain, RELAXED_DOMAIN)
- static Var_mp_type **Vtype** (varsView, ALL_VIEW)
- static Var_mp_type **Vtype** (varsView, DESIGN_VIEW)
- static Var_mp_type **Vtype** (varsView, UNCERTAIN_VIEW)
- static Var_mp_type **Vtype** (varsView, ALEATORY_UNCERTAIN_VIEW)
- static Var_mp_type **Vtype** (varsView, EPISTEMIC_UNCERTAIN_VIEW)
- static Var_mp_type **Vtype** (varsView, STATE_VIEW)
- template<class ContainerT >
  void flatten_num_array (const std::vector< ContainerT > &input_array, IntArray **pia)

  Free convenience function that flatten sizes of an array of std containers; takes an array of containers and returns an IntArray containing the sizes of each container in the input array. Note: Did not specialize for vector<RealVector> as no current use cases.
- void **dn2f_** (int *n, int *p, Real *x, Calcrj, int *iv, int *liv, int *lv, Real *v, int *ui, void *ur, Vf)
- void **dn2fb_** (int *n, int *p, Real *x, Real *b, Calcrj, int *iv, int *liv, int *lv, Real *v, int *ui, void *ur, Vf)
- void **dn2g_** (int *n, int *p, Real *x, Calcrj, Calcrj, int *iv, int *liv, int *lv, Real *v, int *ui, void *ur, Vf)
- void **dn2gb_** (int *n, int *p, Real *x, Real *b, Calcrj, Calcrj, int *iv, int *liv, int *lv, Real *v, int *ui, void *ur, Vf)
- void **divset_** (int *, int *, int *, int *, Real *)
- double **dr7mdc_** (int *)
- static void **Rswapchk** (Nl2Misc *q)
- static int **hasnaninf** (const double *d, int n)
- NLPQLPOptimizer * **new_NLPQLPOptimizer** (ProblemDescDB &problem_db)
- NLPQLPOptimizer * **new_NLPQLPOptimizer** (Model &model)
- NLPQLPOptimizer * **new_NLPQLPOptimizer** (ProblemDescDB &problem_db, Model &model)
- NPSOLOptimizer * **new_NPSOLOptimizer** (ProblemDescDB &problem_db)
- NPSOLOptimizer * **new_NPSOLOptimizer1** (Model &model)
- NPSOLOptimizer * **new_NPSOLOptimizer2** (Model &model, const int &derivative_level, const Real &conv_tol)
- NPSOLOptimizer * **new_NPSOLOptimizer3** (const RealVector &initial_point, const RealVector &var_lower_bnds, const RealVector &var_upper_bnds, const RealMatrix &lin_ineq_coeffs, const RealVector &lin_ineq_lower_bnds, const RealVector &lin_ineq_upper_bnds, const RealMatrix &lin_eq_coeffs, const RealVector &lin_eq_targets, const RealVector &nonlin_ineq_lower_bnds, const RealVector &nonlin_ineq_upper_bnds, const RealVector &nonlin_eq_targets, void(*user_obj_eval)(int &, int &, double *, double &, double *, int &), void(*user_con_eval)(int &, int &, int &, int &, int *, double *, double *, double *, int &), const int &derivative_level, const Real &conv_tol)
- NPSOLOptimizer * **new_NPSOLOptimizer** (ProblemDescDB &problem_db, Model &model)
- NPSOLOptimizer * **new_NPSOLOptimizer** (Model &model)
- NPSOLOptimizer * **new_NPSOLOptimizer** (Model &model, const int &, const Real &)

- NPSOLOptimizer * **new_NPSOLOptimizer** (const RealVector &initial_point, const RealVector &var_lower_bnds, const RealVector &var_upper_bnds, const RealMatrix &lin_ineq_coeffs, const RealVector &lin_ineq_lower_bnds, const RealVector &lin_ineq_upper_bnds, const RealMatrix &lin_eq_coeffs, const RealVector &lin_eq_targets, const RealVector &nonlin_ineq_lower_bnds, const RealVector &nonlin_ineq_upper_bnds, const RealVector &nonlin_eq_targets, void(*user_obj_eval)(int &, int &, double *, double &, double *, int &), void(*user_con_eval)(int &, int &, int &, int &, int *, double *, double *, double *, int &), const int &derivative_level, const Real &conv_tol)
- void start_dakota_heartbeat (int)
- void **dak_sigcatch** (int sig)
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, ParallelLevel &pl)

  MPIUnpackBuffer extraction operator for ParallelLevel. Calls read(MPIUnpackBuffer&).
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const ParallelLevel &pl)

  MPIPackBuffer insertion operator for ParallelLevel. Calls write(MPIPackBuffer&).
- std::istream & operator>> (std::istream &s, ParamResponsePair &pair)

  std::istream extraction operator for ParamResponsePair
- std::ostream & operator<< (std::ostream &s, const ParamResponsePair &pair)

  std::ostream insertion operator for ParamResponsePair
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, ParamResponsePair &pair)

  MPIUnpackBuffer extraction operator for ParamResponsePair.
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const ParamResponsePair &pair)

  MPIPackBuffer insertion operator for ParamResponsePair.
- bool operator== (const ParamResponsePair &pair1, const ParamResponsePair &pair2)

  equality operator for ParamResponsePair
- bool operator!= (const ParamResponsePair &pair1, const ParamResponsePair &pair2)

  inequality operator for ParamResponsePair
- static void * **binsearch** (void *kw, size_t kwsize, size_t n, const char *key)
- static const char * **Begins** (const String &entry_name, const char *s)
- static void **Bad_name** (String entry_name, const char *where)
- static void **Locked_db** ()
- static void **Null_rep** (const char *who)
- static void **Null_rep1** (const char *who)
- MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, ProgramOptions &progopt)

  MPIUnpackBuffer extraction operator.
- MPIPackBuffer & operator<< (MPIPackBuffer &s, const ProgramOptions &progopt)

  MPIPackBuffer insertion operator.
- bool set_compare (const ParamResponsePair &database_pr, const ActiveSet &search_set)

  search function for a particular ParamResponsePair within a PRPList based on ActiveSet content (request vector and derivative variables vector)
- bool id_vars_exact_compare (const ParamResponsePair &database_pr, const ParamResponsePair &search_pr)

  search function for a particular ParamResponsePair within a PRPMultiIndex
- std::size_t hash_value (const ParamResponsePair &prp)

  hash_value for ParamResponsePairs stored in a PRPMultiIndex
- PRPCacheHIter hashedCacheBegin (PRPCache &prp_cache)

  hashed definition of cache begin

- PRPCacheHIter hashedCacheEnd (PRPCache &prp_cache)

  hashed definition of cache end
- PRPQueueHIter hashedQueueBegin (PRPQueue &prp_queue)

  hashed definition of queue begin
- PRPQueueHIter hashedQueueEnd (PRPQueue &prp_queue)

  hashed definition of queue end
- PRPCacheHIter lookup_by_val (PRPMultiIndexCache &prp_cache, const ParamResponsePair &search_pr)

  find a ParamResponsePair based on the interface id, variables, and ActiveSet search data within search_pr.
- PRPCacheHIter lookup_by_val (PRPMultiIndexCache &prp_cache, const String &search_interface_id, const Variables &search_vars, const ActiveSet &search_set)

  find a ParamResponsePair within a PRPMultiIndexCache based on the interface id, variables, and ActiveSet search data
- PRPCacheOIter **lookup_by_nearby_val** (PRPMultiIndexCache &prp_cache, const String &search_interface_id, const Variables &search_vars, const ActiveSet &search_set, Real tol)
- PRPCacheOIter lookup_by_ids (PRPMultiIndexCache &prp_cache, const IntStringPair &search_ids)

  find a ParamResponsePair within a PRPMultiIndexCache based on search_ids (i.e. std::pair<eval_id,interface_id>) search data
- PRPCacheOIter **lookup_by_ids** (PRPMultiIndexCache &prp_cache, const IntStringPair &search_ids, const ParamResponsePair &search_pr)
- PRPQueueHIter lookup_by_val (PRPMultiIndexQueue &prp_queue, const ParamResponsePair &search_pr)

  find a ParamResponsePair based on the interface id, variables, and ActiveSet search data within search_pr.
- PRPQueueHIter lookup_by_val (PRPMultiIndexQueue &prp_queue, const String &search_interface_id, const Variables &search_vars, const ActiveSet &search_set)

  find a ParamResponsePair within a PRPMultiIndexQueue based on interface id, variables, and ActiveSet search data
- PRPQueueOIter lookup_by_eval_id (PRPMultiIndexQueue &prp_queue, int search_id)

  find a ParamResponsePair within a PRPMultiIndexQueue based on search_id (i.e. integer eval_id) search data
- void print_restart (int argc, char **argv, String print_dest)

  print a restart file
- void print_restart_pdb (int argc, char **argv, String print_dest)

  print a restart file (PDB format)
- void print_restart_tabular (int argc, char **argv, String print_dest)

  print a restart file (tabular format)
- void read_neutral (int argc, char **argv)

  read a restart file (neutral file format)
- void repair_restart (int argc, char **argv, String identifier_type)

  repair a restart file by removing corrupted evaluations
- void concatenate_restart (int argc, char **argv)

  concatenate multiple restart files
- static HANDLE * **wait_setup** (std::map< pid_t, int > *M, size_t *pn)
- static int **wait_for_one** (size_t n, HANDLE *h, int req1, size_t *pi)
- int salinas_main (int argc, char *argv[ ], MPI_Comm *comm)

  subroutine interface to SALINAS simulation code
- std::string **get_cwd_str** ()
- std::vector< std::string > get_pathext ()
- bool contains (const bfs::path &dir_path, const std::string &file_name, boost::filesystem::path &complete_filepath)

Variables

- PRPCache data_pairs

  contains all parameter/response pairs.
- short abort_mode = ABORT_EXITS

  by default Dakota exits or calls MPI_Abort on errors
- std::ostream ∗ dakota_cout = &std::cout

  `DAKOTA stdout initially points to`
  < std::cout, but may be redirected to a tagged ofstream if there are < concurrent iterators.
- std::ostream ∗ dakota_cerr = &std::cerr

  `DAKOTA stderr initially points to`
  < std::cerr, but may be redirected to a tagged ofstream if there are < concurrent iterators.
- ResultsManager iterator_results_db

  Global results database for iterator results.
- int write_precision = 10

  used in ostream data output functions < (restart_util.cpp overrides default value)
- MPIManager dummy_mpi_mgr

  dummy MPIManager for ref initialization
- ProgramOptions dummy_prg_opt

  dummy ProgramOptions for ref initialization
- OutputManager dummy_out_mgr

  dummy OutputManager for ref initialization
- ParallelLibrary dummy_lib

  dummy ParallelLibrary for ref initialization
- ProblemDescDB dummy_db

  dummy ProblemDescDB for ref initialization
- int mc_ptr_int = 0

  global pointer for ModelCenter API
- int dc_ptr_int = 0

  global pointer for ModelCenter eval DB
- ProblemDescDB ∗ Dak_pddb

  set by ProblemDescDB, for use in parsing
- const size_t _NPOS = ∼(size_t)0

  special value returned by index() when entry not found
- const bool TABULAR_IFACE_ID = true

  whether to write/read interface IDs in tabular data
- Interface dummy_interface

  `dummy Interface object used for mandatory`
  < reference initialization or default virtual < function return by reference when a real < Interface instance is unavailable
- Model dummy_model

  `dummy Model object used for mandatory reference`
  < initialization or default virtual function < return by reference when a real Model instance < is unavailable
- Iterator dummy_iterator

`dummy Iterator object used for mandatory`
< reference initialization or default virtual < function return by reference when a real < Iterator instance is unavailable

- Dakota_funcs ∗ **DF**
- Dakota_funcs **DakFuncs0**
- const char ∗ FIELD_NAMES [ ]
- const int NUMBER_OF_FIELDS = 23
- static GuiKeyWord kw_1 [2]
- static GuiKeyWord **kw_2** [2]
- static GuiKeyWord **kw_3** [1]
- static GuiKeyWord **kw_4** [1]
- static GuiKeyWord **kw_5** [2]
- static GuiKeyWord **kw_6** [2]
- static GuiKeyWord **kw_7** [15]
- static GuiKeyWord **kw_8** [1]
- static GuiKeyWord **kw_9** [4]
- static GuiKeyWord **kw_10** [1]
- static GuiKeyWord **kw_11** [4]
- static GuiKeyWord **kw_12** [1]
- static GuiKeyWord **kw_13** [8]
- static GuiKeyWord **kw_14** [9]
- static GuiKeyWord **kw_15** [12]
- static GuiKeyWord **kw_16** [2]
- static GuiKeyWord **kw_17** [2]
- static GuiKeyWord **kw_18** [3]
- static GuiKeyWord **kw_19** [2]
- static GuiKeyWord **kw_20** [2]
- static GuiKeyWord **kw_21** [9]
- static GuiKeyWord **kw_22** [2]
- static GuiKeyWord **kw_23** [1]
- static GuiKeyWord **kw_24** [1]
- static GuiKeyWord **kw_25** [2]
- static GuiKeyWord **kw_26** [4]
- static GuiKeyWord **kw_27** [2]
- static GuiKeyWord **kw_28** [3]
- static GuiKeyWord **kw_29** [3]
- static GuiKeyWord **kw_30** [2]
- static GuiKeyWord **kw_31** [3]
- static GuiKeyWord **kw_32** [2]
- static GuiKeyWord **kw_33** [15]
- static GuiKeyWord **kw_34** [7]
- static GuiKeyWord **kw_35** [2]
- static GuiKeyWord **kw_36** [19]
- static GuiKeyWord **kw_37** [2]
- static GuiKeyWord **kw_38** [3]
- static GuiKeyWord **kw_39** [5]

- static GuiKeyWord **kw_40** [1]
- static GuiKeyWord **kw_41** [1]
- static GuiKeyWord **kw_42** [4]
- static GuiKeyWord **kw_43** [6]
- static GuiKeyWord **kw_44** [2]
- static GuiKeyWord **kw_45** [2]
- static GuiKeyWord **kw_46** [2]
- static GuiKeyWord **kw_47** [2]
- static GuiKeyWord **kw_48** [2]
- static GuiKeyWord **kw_49** [2]
- static GuiKeyWord **kw_50** [3]
- static GuiKeyWord **kw_51** [6]
- static GuiKeyWord **kw_52** [2]
- static GuiKeyWord **kw_53** [3]
- static GuiKeyWord **kw_54** [5]
- static GuiKeyWord **kw_55** [1]
- static GuiKeyWord **kw_56** [1]
- static GuiKeyWord **kw_57** [4]
- static GuiKeyWord **kw_58** [4]
- static GuiKeyWord **kw_59** [9]
- static GuiKeyWord **kw_60** [4]
- static GuiKeyWord **kw_61** [7]
- static GuiKeyWord **kw_62** [8]
- static GuiKeyWord **kw_63** [2]
- static GuiKeyWord **kw_64** [12]
- static GuiKeyWord **kw_65** [3]
- static GuiKeyWord **kw_66** [2]
- static GuiKeyWord **kw_67** [3]
- static GuiKeyWord **kw_68** [2]
- static GuiKeyWord **kw_69** [5]
- static GuiKeyWord **kw_70** [4]
- static GuiKeyWord **kw_71** [15]
- static GuiKeyWord **kw_72** [3]
- static GuiKeyWord **kw_73** [2]
- static GuiKeyWord **kw_74** [2]
- static GuiKeyWord **kw_75** [18]
- static GuiKeyWord **kw_76** [14]
- static GuiKeyWord **kw_77** [12]
- static GuiKeyWord **kw_78** [10]
- static GuiKeyWord **kw_79** [1]
- static GuiKeyWord **kw_80** [15]
- static GuiKeyWord **kw_81** [15]
- static GuiKeyWord **kw_82** [2]
- static GuiKeyWord **kw_83** [2]
- static GuiKeyWord **kw_84** [3]
- static GuiKeyWord **kw_85** [7]

- static GuiKeyWord **kw_86** [9]
- static GuiKeyWord **kw_87** [3]
- static GuiKeyWord **kw_88** [1]
- static GuiKeyWord **kw_89** [9]
- static GuiKeyWord **kw_90** [1]
- static GuiKeyWord **kw_91** [11]
- static GuiKeyWord **kw_92** [2]
- static GuiKeyWord **kw_93** [3]
- static GuiKeyWord **kw_94** [2]
- static GuiKeyWord **kw_95** [3]
- static GuiKeyWord **kw_96** [2]
- static GuiKeyWord **kw_97** [11]
- static GuiKeyWord **kw_98** [2]
- static GuiKeyWord **kw_99** [2]
- static GuiKeyWord **kw_100** [3]
- static GuiKeyWord **kw_101** [2]
- static GuiKeyWord **kw_102** [2]
- static GuiKeyWord **kw_103** [2]
- static GuiKeyWord **kw_104** [3]
- static GuiKeyWord **kw_105** [5]
- static GuiKeyWord **kw_106** [12]
- static GuiKeyWord **kw_107** [2]
- static GuiKeyWord **kw_108** [2]
- static GuiKeyWord **kw_109** [2]
- static GuiKeyWord **kw_110** [3]
- static GuiKeyWord **kw_111** [5]
- static GuiKeyWord **kw_112** [8]
- static GuiKeyWord **kw_113** [2]
- static GuiKeyWord **kw_114** [1]
- static GuiKeyWord **kw_115** [1]
- static GuiKeyWord **kw_116** [2]
- static GuiKeyWord **kw_117** [3]
- static GuiKeyWord **kw_118** [2]
- static GuiKeyWord **kw_119** [3]
- static GuiKeyWord **kw_120** [2]
- static GuiKeyWord **kw_121** [2]
- static GuiKeyWord **kw_122** [16]
- static GuiKeyWord **kw_123** [2]
- static GuiKeyWord **kw_124** [1]
- static GuiKeyWord **kw_125** [2]
- static GuiKeyWord **kw_126** [1]
- static GuiKeyWord **kw_127** [1]
- static GuiKeyWord **kw_128** [5]
- static GuiKeyWord **kw_129** [1]
- static GuiKeyWord **kw_130** [2]
- static GuiKeyWord **kw_131** [8]

- static GuiKeyWord **kw_132** [2]
- static GuiKeyWord **kw_133** [3]
- static GuiKeyWord **kw_134** [2]
- static GuiKeyWord **kw_135** [12]
- static GuiKeyWord **kw_136** [3]
- static GuiKeyWord **kw_137** [3]
- static GuiKeyWord **kw_138** [2]
- static GuiKeyWord **kw_139** [1]
- static GuiKeyWord **kw_140** [1]
- static GuiKeyWord **kw_141** [2]
- static GuiKeyWord **kw_142** [3]
- static GuiKeyWord **kw_143** [2]
- static GuiKeyWord **kw_144** [7]
- static GuiKeyWord **kw_145** [3]
- static GuiKeyWord **kw_146** [5]
- static GuiKeyWord **kw_147** [4]
- static GuiKeyWord **kw_148** [10]
- static GuiKeyWord **kw_149** [1]
- static GuiKeyWord **kw_150** [2]
- static GuiKeyWord **kw_151** [4]
- static GuiKeyWord **kw_152** [2]
- static GuiKeyWord **kw_153** [7]
- static GuiKeyWord **kw_154** [16]
- static GuiKeyWord **kw_155** [2]
- static GuiKeyWord **kw_156** [5]
- static GuiKeyWord **kw_157** [3]
- static GuiKeyWord **kw_158** [1]
- static GuiKeyWord **kw_159** [6]
- static GuiKeyWord **kw_160** [3]
- static GuiKeyWord **kw_161** [2]
- static GuiKeyWord **kw_162** [1]
- static GuiKeyWord **kw_163** [3]
- static GuiKeyWord **kw_164** [1]
- static GuiKeyWord **kw_165** [2]
- static GuiKeyWord **kw_166** [4]
- static GuiKeyWord **kw_167** [22]
- static GuiKeyWord **kw_168** [1]
- static GuiKeyWord **kw_169** [1]
- static GuiKeyWord **kw_170** [7]
- static GuiKeyWord **kw_171** [2]
- static GuiKeyWord **kw_172** [5]
- static GuiKeyWord **kw_173** [10]
- static GuiKeyWord **kw_174** [1]
- static GuiKeyWord **kw_175** [2]
- static GuiKeyWord **kw_176** [2]
- static GuiKeyWord **kw_177** [2]

- static GuiKeyWord **kw_178** [3]
- static GuiKeyWord **kw_179** [2]
- static GuiKeyWord **kw_180** [11]
- static GuiKeyWord **kw_181** [1]
- static GuiKeyWord **kw_182** [2]
- static GuiKeyWord **kw_183** [4]
- static GuiKeyWord **kw_184** [2]
- static GuiKeyWord **kw_185** [3]
- static GuiKeyWord **kw_186** [2]
- static GuiKeyWord **kw_187** [3]
- static GuiKeyWord **kw_188** [1]
- static GuiKeyWord **kw_189** [1]
- static GuiKeyWord **kw_190** [2]
- static GuiKeyWord **kw_191** [2]
- static GuiKeyWord **kw_192** [1]
- static GuiKeyWord **kw_193** [17]
- static GuiKeyWord **kw_194** [3]
- static GuiKeyWord **kw_195** [6]
- static GuiKeyWord **kw_196** [2]
- static GuiKeyWord **kw_197** [6]
- static GuiKeyWord **kw_198** [3]
- static GuiKeyWord **kw_199** [2]
- static GuiKeyWord **kw_200** [4]
- static GuiKeyWord **kw_201** [3]
- static GuiKeyWord **kw_202** [2]
- static GuiKeyWord **kw_203** [5]
- static GuiKeyWord **kw_204** [2]
- static GuiKeyWord **kw_205** [30]
- static GuiKeyWord **kw_206** [1]
- static GuiKeyWord **kw_207** [4]
- static GuiKeyWord **kw_208** [1]
- static GuiKeyWord **kw_209** [13]
- static GuiKeyWord **kw_210** [2]
- static GuiKeyWord **kw_211** [2]
- static GuiKeyWord **kw_212** [3]
- static GuiKeyWord **kw_213** [2]
- static GuiKeyWord **kw_214** [2]
- static GuiKeyWord **kw_215** [4]
- static GuiKeyWord **kw_216** [2]
- static GuiKeyWord **kw_217** [4]
- static GuiKeyWord **kw_218** [2]
- static GuiKeyWord **kw_219** [28]
- static GuiKeyWord **kw_220** [2]
- static GuiKeyWord **kw_221** [13]
- static GuiKeyWord **kw_222** [12]
- static GuiKeyWord **kw_223** [11]

- static GuiKeyWord **kw_224** [3]
- static GuiKeyWord **kw_225** [4]
- static GuiKeyWord **kw_226** [16]
- static GuiKeyWord **kw_227** [5]
- static GuiKeyWord **kw_228** [2]
- static GuiKeyWord **kw_229** [1]
- static GuiKeyWord **kw_230** [10]
- static GuiKeyWord **kw_231** [4]
- static GuiKeyWord **kw_232** [5]
- static GuiKeyWord **kw_233** [2]
- static GuiKeyWord **kw_234** [2]
- static GuiKeyWord **kw_235** [2]
- static GuiKeyWord **kw_236** [2]
- static GuiKeyWord **kw_237** [4]
- static GuiKeyWord **kw_238** [20]
- static GuiKeyWord **kw_239** [15]
- static GuiKeyWord **kw_240** [7]
- static GuiKeyWord **kw_241** [2]
- static GuiKeyWord **kw_242** [7]
- static GuiKeyWord **kw_243** [1]
- static GuiKeyWord **kw_244** [4]
- static GuiKeyWord **kw_245** [6]
- static GuiKeyWord **kw_246** [13]
- static GuiKeyWord **kw_247** [4]
- static GuiKeyWord **kw_248** [90]
- static GuiKeyWord **kw_249** [1]
- static GuiKeyWord **kw_250** [2]
- static GuiKeyWord **kw_251** [7]
- static GuiKeyWord **kw_252** [2]
- static GuiKeyWord **kw_253** [1]
- static GuiKeyWord **kw_254** [3]
- static GuiKeyWord **kw_255** [6]
- static GuiKeyWord **kw_256** [2]
- static GuiKeyWord **kw_257** [2]
- static GuiKeyWord **kw_258** [2]
- static GuiKeyWord **kw_259** [3]
- static GuiKeyWord **kw_260** [2]
- static GuiKeyWord **kw_261** [4]
- static GuiKeyWord **kw_262** [7]
- static GuiKeyWord **kw_263** [2]
- static GuiKeyWord **kw_264** [3]
- static GuiKeyWord **kw_265** [2]
- static GuiKeyWord **kw_266** [3]
- static GuiKeyWord **kw_267** [3]
- static GuiKeyWord **kw_268** [5]
- static GuiKeyWord **kw_269** [4]

- static GuiKeyWord **kw_270** [5]
- static GuiKeyWord **kw_271** [3]
- static GuiKeyWord **kw_272** [21]
- static GuiKeyWord **kw_273** [6]
- static GuiKeyWord **kw_274** [3]
- static GuiKeyWord **kw_275** [2]
- static GuiKeyWord **kw_276** [2]
- static GuiKeyWord **kw_277** [5]
- static GuiKeyWord **kw_278** [7]
- static GuiKeyWord **kw_279** [6]
- static GuiKeyWord **kw_280** [9]
- static GuiKeyWord **kw_281** [5]
- static GuiKeyWord **kw_282** [6]
- static GuiKeyWord **kw_283** [8]
- static GuiKeyWord **kw_284** [19]
- static GuiKeyWord **kw_285** [4]
- static GuiKeyWord **kw_286** [10]
- static GuiKeyWord **kw_287** [2]
- static GuiKeyWord **kw_288** [1]
- static GuiKeyWord **kw_289** [2]
- static GuiKeyWord **kw_290** [8]
- static GuiKeyWord **kw_291** [4]
- static GuiKeyWord **kw_292** [6]
- static GuiKeyWord **kw_293** [8]
- static GuiKeyWord **kw_294** [15]
- static GuiKeyWord **kw_295** [4]
- static GuiKeyWord **kw_296** [4]
- static GuiKeyWord **kw_297** [8]
- static GuiKeyWord **kw_298** [7]
- static GuiKeyWord **kw_299** [1]
- static GuiKeyWord **kw_300** [2]
- static GuiKeyWord **kw_301** [19]
- static GuiKeyWord **kw_302** [6]
- static GuiKeyWord **kw_303** [11]
- static GuiKeyWord **kw_304** [5]
- static GuiKeyWord **kw_305** [12]
- static GuiKeyWord **kw_306** [10]
- static GuiKeyWord **kw_307** [8]
- static GuiKeyWord **kw_308** [8]
- static GuiKeyWord **kw_309** [7]
- static GuiKeyWord **kw_310** [7]
- static GuiKeyWord **kw_311** [6]
- static GuiKeyWord **kw_312** [3]
- static GuiKeyWord **kw_313** [9]
- static GuiKeyWord **kw_314** [8]
- static GuiKeyWord **kw_315** [7]

- static GuiKeyWord **kw_316** [7]
- static GuiKeyWord **kw_317** [6]
- static GuiKeyWord **kw_318** [3]
- static GuiKeyWord **kw_319** [9]
- static GuiKeyWord **kw_320** [9]
- static GuiKeyWord **kw_321** [8]
- static GuiKeyWord **kw_322** [3]
- static GuiKeyWord **kw_323** [5]
- static GuiKeyWord **kw_324** [7]
- static GuiKeyWord **kw_325** [7]
- static GuiKeyWord **kw_326** [4]
- static GuiKeyWord **kw_327** [7]
- static GuiKeyWord **kw_328** [11]
- static GuiKeyWord **kw_329** [6]
- static GuiKeyWord **kw_330** [6]
- static GuiKeyWord **kw_331** [6]
- static GuiKeyWord **kw_332** [3]
- static GuiKeyWord **kw_333** [5]
- static GuiKeyWord **kw_334** [2]
- static GuiKeyWord **kw_335** [4]
- static GuiKeyWord **kw_336** [11]
- static GuiKeyWord **kw_337** [7]
- static GuiKeyWord **kw_338** [5]
- static GuiKeyWord **kw_339** [11]
- static GuiKeyWord **kw_340** [3]
- static GuiKeyWord **kw_341** [9]
- static GuiKeyWord **kw_342** [7]
- static GuiKeyWord **kw_343** [7]
- static GuiKeyWord **kw_344** [34]
- static GuiKeyWord **kw_345** [6]
- static KeyWord kw_1 [2]
- static KeyWord **kw_2** [2]
- static KeyWord **kw_3** [1]
- static KeyWord **kw_4** [1]
- static KeyWord **kw_5** [2]
- static KeyWord **kw_6** [2]
- static KeyWord **kw_7** [15]
- static KeyWord **kw_8** [1]
- static KeyWord **kw_9** [4]
- static KeyWord **kw_10** [1]
- static KeyWord **kw_11** [4]
- static KeyWord **kw_12** [1]
- static KeyWord **kw_13** [8]
- static KeyWord **kw_14** [9]
- static KeyWord **kw_15** [12]
- static KeyWord **kw_16** [2]

- static KeyWord **kw_17** [2]
- static KeyWord **kw_18** [3]
- static KeyWord **kw_19** [2]
- static KeyWord **kw_20** [2]
- static KeyWord **kw_21** [9]
- static KeyWord **kw_22** [1]
- static KeyWord **kw_23** [2]
- static KeyWord **kw_24** [2]
- static KeyWord **kw_25** [1]
- static KeyWord **kw_26** [1]
- static KeyWord **kw_27** [2]
- static KeyWord **kw_28** [4]
- static KeyWord **kw_29** [4]
- static KeyWord **kw_30** [2]
- static KeyWord **kw_31** [3]
- static KeyWord **kw_32** [3]
- static KeyWord **kw_33** [2]
- static KeyWord **kw_34** [3]
- static KeyWord **kw_35** [2]
- static KeyWord **kw_36** [11]
- static KeyWord **kw_37** [9]
- static KeyWord **kw_38** [7]
- static KeyWord **kw_39** [2]
- static KeyWord **kw_40** [11]
- static KeyWord **kw_41** [2]
- static KeyWord **kw_42** [3]
- static KeyWord **kw_43** [5]
- static KeyWord **kw_44** [1]
- static KeyWord **kw_45** [1]
- static KeyWord **kw_46** [4]
- static KeyWord **kw_47** [6]
- static KeyWord **kw_48** [2]
- static KeyWord **kw_49** [2]
- static KeyWord **kw_50** [2]
- static KeyWord **kw_51** [2]
- static KeyWord **kw_52** [2]
- static KeyWord **kw_53** [3]
- static KeyWord **kw_54** [2]
- static KeyWord **kw_55** [3]
- static KeyWord **kw_56** [4]
- static KeyWord **kw_57** [2]
- static KeyWord **kw_58** [3]
- static KeyWord **kw_59** [5]
- static KeyWord **kw_60** [1]
- static KeyWord **kw_61** [1]
- static KeyWord **kw_62** [4]

- static KeyWord **kw_63** [2]
- static KeyWord **kw_64** [8]
- static KeyWord **kw_65** [4]
- static KeyWord **kw_66** [5]
- static KeyWord **kw_67** [3]
- static KeyWord **kw_68** [2]
- static KeyWord **kw_69** [4]
- static KeyWord **kw_70** [2]
- static KeyWord **kw_71** [8]
- static KeyWord **kw_72** [3]
- static KeyWord **kw_73** [2]
- static KeyWord **kw_74** [3]
- static KeyWord **kw_75** [2]
- static KeyWord **kw_76** [5]
- static KeyWord **kw_77** [4]
- static KeyWord **kw_78** [11]
- static KeyWord **kw_79** [2]
- static KeyWord **kw_80** [3]
- static KeyWord **kw_81** [2]
- static KeyWord **kw_82** [2]
- static KeyWord **kw_83** [12]
- static KeyWord **kw_84** [8]
- static KeyWord **kw_85** [4]
- static KeyWord **kw_86** [3]
- static KeyWord **kw_87** [1]
- static KeyWord **kw_88** [14]
- static KeyWord **kw_89** [7]
- static KeyWord **kw_90** [2]
- static KeyWord **kw_91** [2]
- static KeyWord **kw_92** [3]
- static KeyWord **kw_93** [7]
- static KeyWord **kw_94** [5]
- static KeyWord **kw_95** [3]
- static KeyWord **kw_96** [1]
- static KeyWord **kw_97** [8]
- static KeyWord **kw_98** [1]
- static KeyWord **kw_99** [11]
- static KeyWord **kw_100** [2]
- static KeyWord **kw_101** [3]
- static KeyWord **kw_102** [2]
- static KeyWord **kw_103** [3]
- static KeyWord **kw_104** [2]
- static KeyWord **kw_105** [7]
- static KeyWord **kw_106** [2]
- static KeyWord **kw_107** [2]
- static KeyWord **kw_108** [3]

- static KeyWord **kw_109** [2]
- static KeyWord **kw_110** [2]
- static KeyWord **kw_111** [2]
- static KeyWord **kw_112** [3]
- static KeyWord **kw_113** [5]
- static KeyWord **kw_114** [8]
- static KeyWord **kw_115** [2]
- static KeyWord **kw_116** [2]
- static KeyWord **kw_117** [2]
- static KeyWord **kw_118** [3]
- static KeyWord **kw_119** [5]
- static KeyWord **kw_120** [7]
- static KeyWord **kw_121** [2]
- static KeyWord **kw_122** [1]
- static KeyWord **kw_123** [1]
- static KeyWord **kw_124** [3]
- static KeyWord **kw_125** [2]
- static KeyWord **kw_126** [3]
- static KeyWord **kw_127** [2]
- static KeyWord **kw_128** [3]
- static KeyWord **kw_129** [2]
- static KeyWord **kw_130** [2]
- static KeyWord **kw_131** [14]
- static KeyWord **kw_132** [2]
- static KeyWord **kw_133** [3]
- static KeyWord **kw_134** [1]
- static KeyWord **kw_135** [2]
- static KeyWord **kw_136** [1]
- static KeyWord **kw_137** [1]
- static KeyWord **kw_138** [5]
- static KeyWord **kw_139** [1]
- static KeyWord **kw_140** [2]
- static KeyWord **kw_141** [6]
- static KeyWord **kw_142** [2]
- static KeyWord **kw_143** [3]
- static KeyWord **kw_144** [2]
- static KeyWord **kw_145** [8]
- static KeyWord **kw_146** [3]
- static KeyWord **kw_147** [3]
- static KeyWord **kw_148** [2]
- static KeyWord **kw_149** [1]
- static KeyWord **kw_150** [1]
- static KeyWord **kw_151** [2]
- static KeyWord **kw_152** [3]
- static KeyWord **kw_153** [2]
- static KeyWord **kw_154** [7]

- static KeyWord **kw_155** [3]
- static KeyWord **kw_156** [5]
- static KeyWord **kw_157** [4]
- static KeyWord **kw_158** [10]
- static KeyWord **kw_159** [1]
- static KeyWord **kw_160** [2]
- static KeyWord **kw_161** [4]
- static KeyWord **kw_162** [2]
- static KeyWord **kw_163** [5]
- static KeyWord **kw_164** [8]
- static KeyWord **kw_165** [2]
- static KeyWord **kw_166** [5]
- static KeyWord **kw_167** [3]
- static KeyWord **kw_168** [1]
- static KeyWord **kw_169** [6]
- static KeyWord **kw_170** [7]
- static KeyWord **kw_171** [3]
- static KeyWord **kw_172** [2]
- static KeyWord **kw_173** [1]
- static KeyWord **kw_174** [3]
- static KeyWord **kw_175** [1]
- static KeyWord **kw_176** [2]
- static KeyWord **kw_177** [4]
- static KeyWord **kw_178** [8]
- static KeyWord **kw_179** [1]
- static KeyWord **kw_180** [1]
- static KeyWord **kw_181** [5]
- static KeyWord **kw_182** [2]
- static KeyWord **kw_183** [5]
- static KeyWord **kw_184** [10]
- static KeyWord **kw_185** [1]
- static KeyWord **kw_186** [2]
- static KeyWord **kw_187** [2]
- static KeyWord **kw_188** [2]
- static KeyWord **kw_189** [3]
- static KeyWord **kw_190** [2]
- static KeyWord **kw_191** [7]
- static KeyWord **kw_192** [1]
- static KeyWord **kw_193** [2]
- static KeyWord **kw_194** [4]
- static KeyWord **kw_195** [2]
- static KeyWord **kw_196** [2]
- static KeyWord **kw_197** [1]
- static KeyWord **kw_198** [3]
- static KeyWord **kw_199** [1]
- static KeyWord **kw_200** [2]

- static KeyWord **kw_201** [3]
- static KeyWord **kw_202** [1]
- static KeyWord **kw_203** [1]
- static KeyWord **kw_204** [2]
- static KeyWord **kw_205** [2]
- static KeyWord **kw_206** [1]
- static KeyWord **kw_207** [17]
- static KeyWord **kw_208** [3]
- static KeyWord **kw_209** [6]
- static KeyWord **kw_210** [2]
- static KeyWord **kw_211** [6]
- static KeyWord **kw_212** [3]
- static KeyWord **kw_213** [2]
- static KeyWord **kw_214** [4]
- static KeyWord **kw_215** [3]
- static KeyWord **kw_216** [2]
- static KeyWord **kw_217** [3]
- static KeyWord **kw_218** [2]
- static KeyWord **kw_219** [25]
- static KeyWord **kw_220** [1]
- static KeyWord **kw_221** [4]
- static KeyWord **kw_222** [1]
- static KeyWord **kw_223** [8]
- static KeyWord **kw_224** [2]
- static KeyWord **kw_225** [2]
- static KeyWord **kw_226** [3]
- static KeyWord **kw_227** [2]
- static KeyWord **kw_228** [2]
- static KeyWord **kw_229** [4]
- static KeyWord **kw_230** [2]
- static KeyWord **kw_231** [4]
- static KeyWord **kw_232** [2]
- static KeyWord **kw_233** [23]
- static KeyWord **kw_234** [2]
- static KeyWord **kw_235** [3]
- static KeyWord **kw_236** [4]
- static KeyWord **kw_237** [2]
- static KeyWord **kw_238** [4]
- static KeyWord **kw_239** [3]
- static KeyWord **kw_240** [3]
- static KeyWord **kw_241** [4]
- static KeyWord **kw_242** [7]
- static KeyWord **kw_243** [5]
- static KeyWord **kw_244** [2]
- static KeyWord **kw_245** [1]
- static KeyWord **kw_246** [8]

- static KeyWord **kw_247** [3]
- static KeyWord **kw_248** [5]
- static KeyWord **kw_249** [2]
- static KeyWord **kw_250** [2]
- static KeyWord **kw_251** [2]
- static KeyWord **kw_252** [2]
- static KeyWord **kw_253** [4]
- static KeyWord **kw_254** [6]
- static KeyWord **kw_255** [5]
- static KeyWord **kw_256** [7]
- static KeyWord **kw_257** [2]
- static KeyWord **kw_258** [7]
- static KeyWord **kw_259** [1]
- static KeyWord **kw_260** [4]
- static KeyWord **kw_261** [6]
- static KeyWord **kw_262** [13]
- static KeyWord **kw_263** [4]
- static KeyWord **kw_264** [90]
- static KeyWord **kw_265** [1]
- static KeyWord **kw_266** [2]
- static KeyWord **kw_267** [7]
- static KeyWord **kw_268** [2]
- static KeyWord **kw_269** [1]
- static KeyWord **kw_270** [3]
- static KeyWord **kw_271** [6]
- static KeyWord **kw_272** [2]
- static KeyWord **kw_273** [2]
- static KeyWord **kw_274** [2]
- static KeyWord **kw_275** [3]
- static KeyWord **kw_276** [2]
- static KeyWord **kw_277** [4]
- static KeyWord **kw_278** [7]
- static KeyWord **kw_279** [2]
- static KeyWord **kw_280** [3]
- static KeyWord **kw_281** [2]
- static KeyWord **kw_282** [3]
- static KeyWord **kw_283** [3]
- static KeyWord **kw_284** [5]
- static KeyWord **kw_285** [4]
- static KeyWord **kw_286** [5]
- static KeyWord **kw_287** [3]
- static KeyWord **kw_288** [21]
- static KeyWord **kw_289** [6]
- static KeyWord **kw_290** [3]
- static KeyWord **kw_291** [1]
- static KeyWord **kw_292** [2]

- static KeyWord **kw_293** [2]
- static KeyWord **kw_294** [5]
- static KeyWord **kw_295** [7]
- static KeyWord **kw_296** [6]
- static KeyWord **kw_297** [9]
- static KeyWord **kw_298** [5]
- static KeyWord **kw_299** [6]
- static KeyWord **kw_300** [8]
- static KeyWord **kw_301** [19]
- static KeyWord **kw_302** [4]
- static KeyWord **kw_303** [8]
- static KeyWord **kw_304** [3]
- static KeyWord **kw_305** [2]
- static KeyWord **kw_306** [1]
- static KeyWord **kw_307** [2]
- static KeyWord **kw_308** [8]
- static KeyWord **kw_309** [4]
- static KeyWord **kw_310** [6]
- static KeyWord **kw_311** [8]
- static KeyWord **kw_312** [15]
- static KeyWord **kw_313** [4]
- static KeyWord **kw_314** [4]
- static KeyWord **kw_315** [7]
- static KeyWord **kw_316** [1]
- static KeyWord **kw_317** [2]
- static KeyWord **kw_318** [19]
- static KeyWord **kw_319** [6]
- static KeyWord **kw_320** [11]
- static KeyWord **kw_321** [5]
- static KeyWord **kw_322** [12]
- static KeyWord **kw_323** [10]
- static KeyWord **kw_324** [8]
- static KeyWord **kw_325** [8]
- static KeyWord **kw_326** [7]
- static KeyWord **kw_327** [7]
- static KeyWord **kw_328** [6]
- static KeyWord **kw_329** [3]
- static KeyWord **kw_330** [9]
- static KeyWord **kw_331** [8]
- static KeyWord **kw_332** [7]
- static KeyWord **kw_333** [7]
- static KeyWord **kw_334** [6]
- static KeyWord **kw_335** [3]
- static KeyWord **kw_336** [9]
- static KeyWord **kw_337** [9]
- static KeyWord **kw_338** [8]

- static KeyWord **kw_339** [3]
- static KeyWord **kw_340** [5]
- static KeyWord **kw_341** [7]
- static KeyWord **kw_342** [7]
- static KeyWord **kw_343** [4]
- static KeyWord **kw_344** [7]
- static KeyWord **kw_345** [11]
- static KeyWord **kw_346** [6]
- static KeyWord **kw_347** [6]
- static KeyWord **kw_348** [6]
- static KeyWord **kw_349** [3]
- static KeyWord **kw_350** [5]
- static KeyWord **kw_351** [2]
- static KeyWord **kw_352** [4]
- static KeyWord **kw_353** [11]
- static KeyWord **kw_354** [7]
- static KeyWord **kw_355** [5]
- static KeyWord **kw_356** [11]
- static KeyWord **kw_357** [3]
- static KeyWord **kw_358** [9]
- static KeyWord **kw_359** [7]
- static KeyWord **kw_360** [7]
- static KeyWord **kw_361** [34]
- static KeyWord **kw_362** [6]
- FILE ∗ **nidrin**
- const size_t NIDR_MAX_ERROR_LEN = 8192

  maximum error length is roughly 100 lines at 80 char; using fixed error length instead of investing in converting to vsnprintf (C++11)
- static const char ∗ **aln_scaletypes** [ ] = { ”auto”, ”log”, ”none”, 0 }
- static Var_uinfo **CAUVLbl** [CAUVar_Nkinds]
- static Var_uinfo **DAUIVLbl** [DAUIVar_Nkinds]
- static Var_uinfo **DAUSVLbl** [DAUSVar_Nkinds]
- static Var_uinfo **DAURVLbl** [DAURVar_Nkinds]
- static Var_uinfo **CEUVLbl** [CEUVar_Nkinds]
- static Var_uinfo **DEUIVLbl** [DEUIVar_Nkinds]
- static Var_uinfo **DEUSVLbl** [DEUSVar_Nkinds]
- static Var_uinfo **DEURVLbl** [DEURVar_Nkinds]
- static Var_uinfo **DiscSetLbl** [DiscSetVar_Nkinds]
- static VarLabelChk DesignAndStateLabelsCheck [ ]

  Variables label array designations for design and state. All non-uncertain variables need to be in this array. Used in check_variables_node to check lengths and make_variable_defaults to build labels.
- static VLreal VLUncertainReal [NUM_UNC_REAL_CONT]

  Variables labels/bounds/values check array for real-valued uncertain variables; one array entry per contiguous container. These associate the individual variables given by, e.g., CAUVLbl, with the contiguous container in which they are stored.
- static VLint VLUncertainInt [NUM_UNC_INT_CONT]

Variables labels/bounds/values check array for integer-valued uncertain variables; one array entry per contiguous container. These associate the individual variables given by, e.g., DAUIVLbl, with the contiguous container in which they are stored.

- static VLstr VLUncertainStr [NUM_UNC_STR_CONT]

  Variables labels/bounds/values check array for string-valued uncertain variables; one array entry per contiguous container. These associate the individual variables given by, e.g., DAUSVLbl, with the contiguous container in which they are stored.
- static int VLR_aleatory [NUM_UNC_REAL_CONT] = { 1, 0, 1, 0 }

  which uncertain real check array containers are aleatory (true = 1)
- static int VLI_aleatory [NUM_UNC_INT_CONT] = { 1, 0 }

  which uncertain integer check array containers are aleatory (true = 1)
- static int VLS_aleatory [NUM_UNC_STR_CONT] = { 1, 0 }

  which uncertain string check array containers are aleatory (true = 1)
- static Var_check **var_mp_check_cv** [ ]
- static Var_check **var_mp_check_dset** [ ]
- static Var_check **var_mp_check_cau** [ ]
- static Var_check **var_mp_check_daui** [ ]
- static Var_check **var_mp_check_daus** [ ]
- static Var_check **var_mp_check_daur** [ ]
- static Var_check **var_mp_check_ceu** [ ]
- static Var_check **var_mp_check_deui** [ ]
- static Var_check **var_mp_check_deus** [ ]
- static Var_check **var_mp_check_deur** [ ]
- static Var_rcheck var_mp_cbound [ ]

  This is used within check_variables_node(): Var_RealBoundIPCheck() is applied to validate bounds and initial points.
- static Var_icheck var_mp_drange [ ]

  This is used in check_variables_node(): Var_IntBoundIPCheck() is applied to validate bounds and initial points, and in make_variable_defaults(): Vgen_∗ is called to infer bounds.
- static time_t **start_time**
- const char ∗ SCI_FIELD_NAMES [ ]
- const int SCI_NUMBER_OF_FIELDS = 26
- const int LARGE_SCALE = 100

  a (perhaps arbitrary) definition of large scale; choose a large-scale algorithm if numVars >= LARGE_SCALE
- const double POW_VAL = 1.0

  offset used text_book exponent: 1.0 is nominal, 1.4 used for B&B testing

12.1.1 Detailed Description

The primary namespace for DAKOTA. The Dakota namespace encapsulates the core classes of the DAKOTA framework and prevents name clashes with third-party libraries from methods and packages. The C++ source files defining these core classes reside in Dakota/src as ∗.[ch]pp.

Work directory TODO

Doc: we will search for drivers in PATH, workdir (.), RUNDIR

Remove legacy utilities (once concepts migrated)

- In general review cases with race conditions such as single dir getting created / removed for each eval.

- Verify creation/removal in parallel runs (eval_comm_rank 0?); are there scenarios where we should create once ahead of time?
- Enforce tagging when asynch is possible
- Challenge of shared vs. distinct filesystems
- Verify template files exist at parse and that workdir parent exists
- Verify behavior when directory exists
- Allow recursive copy to descend to overwrite leaf nodes when directories already exist
- Old code setting permissions mask 0700
- Workdir with multiple analysis components (per analysis)
- Workdirs shared for each unique concurrent eval (not per eval ID)
- Evaluate environment variables
- How to manage drivers that want just param names in the work dir?!? I think arg_adjust is removing the directory args that aren't needed.
- Consider making the class members for directories and files bfs::paths
- Behavior of file_save when directory not saved
- Error checking: directory was created directory path is a directory directory has rwx for user population worked try/catch around all fs operations
- Verify correct handling of relative vs. absolute files/dirs
- Enforce that first argument must be an executable program for all drivers; at least for fork
- Historical behaviors / features to consider Template dirs on PATH: likely no longer

  Allowed FOO=zorch and would set that in the environment; could allow separate env var specification; otherwise likely remove
- TODO: pass environment to exec as separate pointer

TESTING NEEDS

- Allow nested quotes in driver, at least one level: analysis_driver = 'ad.sh "-opt foo -opt1 goo"' p.in.1 r.out.1
- Env vars will be carried along for now, not expanded before eval; set some helpful env vars before the eval.

12.1.2 Typedef Documentation

typedef bmi::multi_index_container<Dakota::ParamResponsePair, bmi::indexed_by< bmi::ordered_non-_unique<bmi::tag<ordered>, bmi::const_mem_fun<Dakota::ParamResponsePair, const IntStringPair&, &Dakota::ParamResponsePair::eval_interface_ids> >, bmi::hashed_non_unique<bmi::tag<hashed>, bmi::identity<Dakota::ParamResponsePair>, partial_prp_hash, partial_prp_equality> > > PRPMultiIndexCache

Boost Multi-Index Container for globally caching ParamResponsePairs.

For a global cache, both evaluation and interface id's are used for tagging ParamResponsePair records.

typedef bmi::multi_index_container<Dakota::ParamResponsePair, bmi::indexed_by< bmi::ordered_unique<bmi::tag<ordered>, bmi::const_mem_fun<Dakota::ParamResponsePair, int, &Dakota::ParamResponsePair::eval_id> >, bmi::hashed_non_unique<bmi::tag<hashed>, bmi::identity<Dakota::ParamResponsePair>, partial_prp_hash, partial_prp_equality> > > PRPMultiIndexQueue

Boost Multi-Index Container for locally queueing ParamResponsePairs.

For a local queue, interface id's are expected to be consistent, such that evaluation id's are sufficient for tracking particular evaluations.

12.1.3 Enumeration Type Documentation

anonymous enum

Sub-methods, including sampling, inference algorithm, opt algorithm types.

Enumerator

SUBMETHOD_COLLABORATIVE Type of hybrid meta-iterator:

12.1.4 Function Documentation

CommandShell & flush (CommandShell & *shell*)

convenient shell manipulator function to "flush" the shell

global convenience function for manipulating the shell; invokes the class member flush function.

References CommandShell::flush().

Referenced by SysCallApplicInterface::spawn_analysis_to_shell(), SysCallApplicInterface::spawn_evaluation_to_shell(), SysCallApplicInterface::spawn_input_filter_to_shell(), and SysCallApplicInterface::spawn_output_filter_to_shell().

void register_signal_handlers ( )

Tie various signal handlers to Dakota's abort_handler function.

Global function to register signal handlers at top-level.

References abort_handler().

Referenced by main().

void mpi_debug_hold ( )

Global function to hold Dakota processes to help with MPI debugging.

See details in code for details, depending on MPI implementation in use.

Referenced by main().

T Dakota::abort_handler_t (int *code*)

Templatized abort_handler_t method that allows for convenient return from methods that otherwise have no sensible return from error clauses. Usage: MyType& method() { return abort_handler<MyType&>(-1); }

References abort_handler().

bool Dakota::operator!= (const ActiveSet & *set1*, const ActiveSet & *set2*) [inline]

inequality operator for ActiveSet
inequality operator

bool Dakota::operator== (const Model & *m1*, const Model & *m2*) [inline]

equality operator for Envelope is true if same letter instance
equality operator (detect same letter instance)
References Model::modelRep.

bool Dakota::operator!= (const Model & *m1*, const Model & *m2*) [inline]

inequality operator for Envelope is true if different letter instance
inequality operator (detect different letter instances)
References Model::modelRep.

bool Dakota::operator== (const Response & *resp1*, const Response & *resp2*) [inline]

equality operator for Response
equality operator

bool Dakota::operator!= (const Response & *resp1*, const Response & *resp2*) [inline]

inequality operator for Response
inequality operator

bool Dakota::operator!= (const Variables & *vars1*, const Variables & *vars2*) [inline]

inequality operator for Variables
strict inequality operator

void Dakota::write_ordered (std::ostream & *s*, const SizetArray & *comp_totals*, const Teuchos::SerialDenseVector< OrdinalType, ScalarType1 > & *c_vector*, const Teuchos::SerialDenseVector< OrdinalType, ScalarType2 > & *di_vector*, const Teuchos::SerialDenseVector< OrdinalType, ScalarType3 > & *ds_vector*, const Teuchos::SerialDenseVector< OrdinalType, ScalarType4 > & *dr_vector*) [inline]

free function to write Variables data vectors in input spec ordering
written for arbitrary types, but typical use will be ScalarType1 = Real, ScalarType2 = int, ScalarType3 = string, and ScalarType4 = int or Real.
Referenced by ParamStudy::pre_run().

void Dakota::write_ordered (std::ostream & *s*, const SizetArray & *comp_totals*, const Teuchos::SerialDenseVector< OrdinalType, ScalarType1 > & *c_vector*, const Teuchos::SerialDenseVector< OrdinalType, ScalarType2 > & *di_vector*, const boost::multi_array< ScalarType3, 1 > & *ds_array*, const Teuchos::SerialDenseVector< OrdinalType, ScalarType4 > & *dr_vector*) [inline]

free function to write Variables data vectors in input spec ordering
written for arbitrary types, but typical use will be ScalarType1 = Real, ScalarType2 = int, ScalarType3 = string, and ScalarType4 = int or Real.

Real Dakota::getdist (const RealVector & *x1*, const RealVector & *x2*)

Gets the Euclidean distance between x1 and x2
Referenced by mindist(), and mindistindx().

Real Dakota::mindist (const RealVector & *x*, const RealMatrix & *xset*, int *except*)

Returns the minimum distance between the point x and the points in the set xset (compares against all points in xset except point "except"): if except is not needed, pass 0.
References getdist().
Referenced by getRmax(), and NonDAdaptImpSampling::select_rep_points().

Real Dakota::mindistindx (const RealVector & *x*, const RealMatrix & *xset*, const IntArray & *indx*)

Gets the min distance between x and points in the set xset defined by the nindx values in indx.
References getdist().
Referenced by GaussProcApproximation::pointsel_add_sel().

Real Dakota::getRmax (const RealMatrix & *xset*)

Gets the maximum of the min distance between each point and the rest of the set.
References mindist().
Referenced by GaussProcApproximation::pointsel_add_sel().

int Dakota::start_grid_computing (char * *analysis_driver_script*, char * *params_file*, char * *results_file*)

sample function prototype for launching grid computing

int Dakota::stop_grid_computing ( )

sample function prototype for terminating grid computing

int Dakota::perform_analysis (char * *iteration_num*)

sample function prototype for submitting a grid evaluation

string Dakota::asstring (const T & *val*)

Creates a string from the argument *val* using an ostringstream.
This only gets used in this file and is only ever called with ints so no error checking is in place.

Parameters

val	The value of type T to convert to a string.

Returns

The string representation of *val* created using an ostringstream.

Referenced by JEGAOptimizer::LoadTheConstraints().

void start_dakota_heartbeat (int *seconds*)

Heartbeat function provided by dakota_filesystem_utils; pass output interval in seconds, or -1 to use $DAKOTA_HEARTBEAT

Referenced by OutputManager::OutputManager().

bool Dakota::operator== (const ParamResponsePair & *pair1*, const ParamResponsePair & *pair2*) `[inline]`

equality operator for ParamResponsePair

equality operator

References ParamResponsePair::evalInterfaceIds, ParamResponsePair::prPairParameters, and ParamResponsePair::prPairResponse.

bool Dakota::operator!= (const ParamResponsePair & *pair1*, const ParamResponsePair & *pair2*) `[inline]`

inequality operator for ParamResponsePair

inequality operator

bool Dakota::set_compare (const ParamResponsePair & *database_pr*, const ActiveSet & *search_set*) `[inline]`

search function for a particular ParamResponsePair within a PRPList based on ActiveSet content (request vector and derivative variables vector)

a global function to compare the ActiveSet of a particular database_pr (presumed to be in the global history list) with a passed in ActiveSet (search_set).

References ParamResponsePair::active_set(), ActiveSet::derivative_vector(), and ActiveSet::request_vector().

Referenced by lookup_by_val().

bool Dakota::id_vars_exact_compare (const ParamResponsePair & *database_pr*, const ParamResponsePair & *search_pr*) `[inline]`

search function for a particular ParamResponsePair within a PRPMultiIndex

a global function to compare the interface id and variables of a particular database_pr (presumed to be in the global history list) with a passed in key of interface id and variables provided by search_pr.

References ParamResponsePair::interface_id(), and ParamResponsePair::prp_parameters().

Referenced by partial_prp_equality::operator()().

PRPCacheHIter Dakota::lookup_by_val (PRPMultiIndexCache & *prp_cache*, const ParamResponsePair & *search_pr*) `[inline]`

find a ParamResponsePair based on the interface id, variables, and ActiveSet search data within search_pr.

Lookup occurs in two steps: (1) PRPMultiIndexCache lookup based on strict equality in interface id and variables, and (2) set_compare() post-processing based on ActiveSet subset logic.

References ParamResponsePair::active_set(), and set_compare().

Referenced by ApplicationInterface::duplication_detect(), Model::estimate_derivatives(), SurrBasedLocalMinimizer::find_center_approx(), Optimizer::local_objective_recast_retrieve(), lookup_by_val(), SNLLLeastSq::post_run(), SurrBasedMinimizer::print_results(), LeastSq::print_results(), Optimizer::print_results(), DiscrepancyCorrection::search_db(), and NonDLocalReliability::update_mpp_search_data().

PRPQueueHIter Dakota::lookup_by_val (PRPMultiIndexQueue & *prp_queue,* const ParamResponsePair & *search_pr*) `[inline]`

find a ParamResponsePair based on the interface id, variables, and ActiveSet search data within search_pr.

Lookup occurs in two steps: (1) PRPMultiIndexQueue lookup based on strict equality in interface id and variables, and (2) set_compare() post-processing based on ActiveSet subset logic.

References ParamResponsePair::active_set(), and set_compare().

void print_restart (int *argc,* char ** *argv,* String *print_dest*)

print a restart file

Usage: "dakota_restart_util print dakota.rst"

"dakota_restart_util to_neutral dakota.rst dakota.neu"

Prints all evals. in full precision to either stdout or a neutral file. The former is useful for ensuring that duplicate detection is successful in a restarted run (e.g., starting a new method from the previous best), and the latter is used for translating binary files between platforms.

References abort_handler(), ParamResponsePair::eval_id(), ParamResponsePair::write_annotated(), and write_precision.

Referenced by main().

void print_restart_pdb (int *argc,* char ** *argv,* String *print_dest*)

print a restart file (PDB format)

Usage: "dakota_restart_util to_pdb dakota.rst dakota.pdb"

Unrolls all data associated with a particular tag for all evaluations and then writes this data in a tabular format (e.g., to a PDB database or MATLAB/TECPLOT data file).

References abort_handler(), Variables::continuous_variables(), Variables::discrete_int_variables(), Variables::discrete_real_variables(), and Response::function_values().

Referenced by main().

void print_restart_tabular (int *argc,* char ** *argv,* String *print_dest*)

print a restart file (tabular format)

Usage: "dakota_restart_util to_tabular dakota.rst dakota.txt"

Unrolls all data associated with a particular tag for all evaluations and then writes this data in a tabular format (e.g., to a PDB database or MATLAB/TECPLOT data file).

References abort_handler(), Variables::acv(), Variables::adiv(), Variables::adrv(), Variables::adsv(), Variables::all_continuous_variable_labels(), Variables::all_discrete_int_variable_labels(), Variables::all_discrete_real_variable_labels(), Variables::all_discrete_string_variable_labels(), Response::function_labels(), ParamResponsePair::interface_id(), ParamResponsePair::prp_parameters(), ParamResponsePair::prp_response(), ParamResponsePair::write_tabular(), and ParamResponsePair::write_tabular_labels().

Referenced by main().

void read_neutral (int *argc,* char ** *argv*)

read a restart file (neutral file format)

Usage: "dakota_restart_util from_neutral dakota.neu dakota.rst"

Reads evaluations from a neutral file. This is used for translating binary files between platforms.

References ParamResponsePair::read_annotated().

Referenced by main().

void repair_restart (int *argc*, char ** *argv*, String *identifier_type*)

repair a restart file by removing corrupted evaluations

Usage: "dakota_restart_util remove 0.0 dakota_old.rst dakota_new.rst"

"dakota_restart_util remove_ids 2 7 13 dakota_old.rst dakota_new.rst"

Repairs a restart file by removing corrupted evaluations. The identifier for evaluation removal can be either a double precision number (all evaluations having a matching response function value are removed) or a list of integers (all evaluations with matching evaluation ids are removed).

References abort_handler(), Response::active_set_request_vector(), contains(), ParamResponsePair::eval_id(), Response::function_values(), and ParamResponsePair::prp_response().

Referenced by main().

void concatenate_restart (int *argc*, char ** *argv*)

concatenate multiple restart files

Usage: "dakota_restart_util cat dakota_1.rst ... dakota_n.rst dakota_new.rst"

Combines multiple restart files into a single restart database.

References abort_handler().

Referenced by main().

std::vector<std::string> Dakota::get_pathext ( )

Utility function for executable file search algorithms

Referenced by WorkdirHelper::which().

bool Dakota::contains (const bfs::path & *dir_path*, const std::string & *file_name*, boost::filesystem::path & *complete_filepath*) `[inline]`

Utility function for "which" sets complete_filepath from dir_path/file_name combo

12.1.5 Variable Documentation

short abort_mode = ABORT_EXITS

by default Dakota exits or calls MPI_Abort on errors

whether dakota exits/aborts or throws on errors

Referenced by abort_throw_or_exit(), Environment::exit_mode(), and PythonInterface::python_run().

Dakota_funcs DakFuncs0

Initial value:

```
= {
        fprintf,
        abort_handler,
        dlsolver_option,
        continuous_lower_bounds1,
        continuous_upper_bounds1,
        nonlinear_ineq_constraint_lower_bounds1,
        nonlinear_ineq_constraint_upper_bounds1,
        nonlinear_eq_constraint_targets1,
        linear_ineq_constraint_lower_bounds1,
        linear_ineq_constraint_upper_bounds1,
        linear_eq_constraint_targets1,
        linear_ineq_constraint_coeffs1,
        linear_eq_constraint_coeffs1,
```

```
        ComputeResponses1,
        GetFuncs1,
        GetGrads1,
        GetContVars1,
        SetBestContVars1,
        SetBestDiscVars1,
        SetBestRespFns1,
        Get_Real1,
        Get_Int1,
        Get_Bool1
        }
```

const char* FIELD_NAMES[ ]

Initial value:

```
= { "numFns", "numVars", "numACV", "numADIV",
                              "numADRV", "numDerivVars", "xC", "xDI",
                              "xDR", "xCLabels", "xDILabels",
                              "xDRLabels", "directFnASV", "directFnDVV",
                              "fnFlag", "gradFlag", "hessFlag",
                              "fnVals",  "fnGrads",  "fnHessians",
                              "fnLabels", "failure", "currEvalId" }
```

fields to pass to Matlab in Dakota structure

Referenced by MatlabInterface::matlab_engine_run(), and MatlabInterface::MatlabInterface().

const int NUMBER_OF_FIELDS = 23

number of fields in above structure

Referenced by MatlabInterface::matlab_engine_run(), and MatlabInterface::MatlabInterface().

GuiKeyWord kw_1[2] [static]

Initial value:

```
= {
                {"input",11,0,1,0,29},
                {"output",11,0,2,0,31}
                }
```

1335 distinct keywords (plus 207 aliases)

GuiKeyWord kw_2[2] [static]

Initial value:

```
= {
                {"input",11,0,1,0,17},
                {"output",11,0,2,0,19}
                }
```

GuiKeyWord kw_3[1] [static]

Initial value:

```
= {
                {"stop_restart",0x29,0,1,0,11}
                }
```

GuiKeyWord kw_4[1] [static]

Initial value:

```
= {
                {"results_output_file",11,0,1,0,43,0,0.,0.,0.,0,"{File name for results output}
        EnvCommands.html#EnvOutput"}
                }
```

GuiKeyWord kw_5[2] [static]

Initial value:

```
= {
                {"input",11,0,1,0,23},
                {"output",11,0,2,0,25}
                }
```

GuiKeyWord kw_6[2] [static]

Initial value:

```
= {
                {"tabular_data_file",11,0,1,0,37},
                {"tabular_graphics_file",3,0,1,0,36}
                }
```

GuiKeyWord kw_7[15] [static]

Initial value:

```
= {
                {"check",8,0,1,0,3},
                {"error_file",11,0,3,0,7},
                {"graphics",8,0,9,0,33,0,0.,0.,0.,0,"{Graphics flag} EnvCommands.html#EnvOutput"},
                {"method_pointer",3,0,13,0,44},
                {"output_file",11,0,2,0,5},
                {"output_precision",0x29,0,11,0,39,0,0.,0.,0.,0,"{Numeric output precision}
        EnvCommands.html#EnvOutput"},
                {"post_run",8,2,8,0,27,kw_1},
                {"pre_run",8,2,6,0,15,kw_2},
                {"read_restart",11,1,4,0,9,kw_3},
                {"results_output",8,1,12,0,41,kw_4,0.,0.,0.,0,"{Enable results output}
        EnvCommands.html#EnvOutput"},
                {"run",8,2,7,0,21,kw_5},
                {"tabular_data",8,2,10,0,35,kw_6},
                {"tabular_graphics_data",0,2,10,0,34,kw_6},
                {"top_method_pointer",11,0,13,0,45,0,0.,0.,0.,0,"{Method pointer}
        EnvCommands.html#EnvMethPtr"},
                {"write_restart",11,0,5,0,13}
                }
```

GuiKeyWord kw_8[1] [static]

Initial value:

```
= {
                {"cache_tolerance",10,0,1,0,2415}
                }
```

GuiKeyWord kw_9[4] [static]

Initial value:

```
= {
                {"active_set_vector",8,0,1,0,2409},
                {"evaluation_cache",8,0,2,0,2411},
                {"restart_file",8,0,4,0,2417},
                {"strict_cache_equality",8,1,3,0,2413,kw_8}
                }
```

GuiKeyWord kw_10[1] [static]

Initial value:

```
= {
                {"processors_per_analysis",0x19,0,1,0,2385,0,0.,0.,0.,0,"{Number of processors per analysis
        server} InterfCommands.html#InterfApplicDF"}
                }
```

GuiKeyWord kw_11[4] [static]

Initial value:

```
= {
                {"abort",8,0,1,1,2399,0,0.,0.,0.,0,"@[CHOOSE failure mitigation]"},
                {"continuation",8,0,1,1,2405},
                {"recover",14,0,1,1,2403},
                {"retry",9,0,1,1,2401}
                }
```

GuiKeyWord kw_12[1] [static]

Initial value:

```
= {
                {"numpy",8,0,1,0,2391,0,0.,0.,0.,0,"{Python NumPy dataflow}
        InterfCommands.html#InterfApplicMSP"}
                }
```

GuiKeyWord kw_13[8] [static]

Initial value:

```
= {
                {"copy_files",15,0,5,0,2379},
                {"dir_save",0,0,3,0,2374},
                {"dir_tag",0,0,2,0,2372},
                {"directory_save",8,0,3,0,2375,0,0.,0.,0.,0,"{Save work directory}
        InterfCommands.html#InterfApplicF"},
                {"directory_tag",8,0,2,0,2373,0,0.,0.,0.,0,"{Tag work directory}
        InterfCommands.html#InterfApplicF"},
                {"link_files",15,0,4,0,2377},
                {"named",11,0,1,0,2371,0,0.,0.,0.,0,"{Name of work directory}
        InterfCommands.html#InterfApplicF"},
                {"replace",8,0,6,0,2381,0,0.,0.,0.,0,"{Replace existing files}
        InterfCommands.html#InterfApplicF"}
                }
```

GuiKeyWord kw_14[9] [static]

Initial value:

```
= {
                {"allow_existing_results",8,0,3,0,2359,0,0.,0.,0.,0,"{Allow existing results files}
        InterfCommands.html#InterfApplicF"},
                {"aprepro",8,0,5,0,2363,0,0.,0.,0.,0,"{Aprepro parameters file format}
        InterfCommands.html#InterfApplicF"},
                {"dprepro",0,0,5,0,2362},
                {"file_save",8,0,7,0,2367,0,0.,0.,0.,0,"{Parameters and results file saving}
        InterfCommands.html#InterfApplicF"},
                {"file_tag",8,0,6,0,2365,0,0.,0.,0.,0,"{Parameters and results file tagging}
        InterfCommands.html#InterfApplicF"},
                {"parameters_file",11,0,1,0,2355,0,0.,0.,0.,0,"{Parameters file name}
        InterfCommands.html#InterfApplicF"},
                {"results_file",11,0,2,0,2357,0,0.,0.,0.,0,"{Results file name}
        InterfCommands.html#InterfApplicF"},
                {"verbatim",8,0,4,0,2361,0,0.,0.,0.,0,"{Verbatim driver/filter invocation syntax}
        InterfCommands.html#InterfApplicF"},
                {"work_directory",8,8,8,0,2369,kw_13,0.,0.,0.,0,"{Create work directory}
        InterfCommands.html#InterfApplicF"}
                }
```

GuiKeyWord kw_15[12] [static]

Initial value:

```
= {
                {"analysis_components",15,0,1,0,2345,0,0.,0.,0.,0,"{Additional identifiers for use by the
        analysis_drivers} InterfCommands.html#InterfApplic"},
                {"deactivate",8,4,6,0,2407,kw_9,0.,0.,0.,0,"{Feature deactivation}
        InterfCommands.html#InterfApplic"},
                {"direct",8,1,4,1,2383,kw_10,0.,0.,0.,0,"[CHOOSE interface type]{Direct function interface
        } InterfCommands.html#InterfApplicDF"},
                {"failure_capture",8,4,5,0,2397,kw_11,0.,0.,0.,0,"{Failure capturing}
        InterfCommands.html#InterfApplic"},
                {"fork",8,9,4,1,2353,kw_14,0.,0.,0.,0,"@{Fork interface } InterfCommands.html#InterfApplicF
       "},
                {"grid",8,0,4,1,2395,0,0.,0.,0.,0,"{Grid interface } InterfCommands.html#InterfApplicG"},
                {"input_filter",11,0,2,0,2347,0,0.,0.,0.,0,"{Input filter} InterfCommands.html#InterfApplic
       "},
                {"matlab",8,0,4,1,2387,0,0.,0.,0.,0,"{Matlab interface }
        InterfCommands.html#InterfApplicMSP"},
                {"output_filter",11,0,3,0,2349,0,0.,0.,0.,0,"{Output filter}
        InterfCommands.html#InterfApplic"},
                {"python",8,1,4,1,2389,kw_12,0.,0.,0.,0,"{Python interface }
        InterfCommands.html#InterfApplicMSP"},
                {"scilab",8,0,4,1,2393,0,0.,0.,0.,0,"{Scilab interface }
        InterfCommands.html#InterfApplicMSP"},
                {"system",8,9,4,1,2351,kw_14}
                }
```

GuiKeyWord kw_16[2] [static]

Initial value:

```
= {
                {"master",8,0,1,1,2449},
                {"peer",8,0,1,1,2451}
                }
```

GuiKeyWord kw_17[2] [static]

Initial value:

```
= {
                {"dynamic",8,0,1,1,2425},
                {"static",8,0,1,1,2427}
                }
```

GuiKeyWord kw_18[3] [static]

Initial value:

```
= {
                {"analysis_concurrency",0x19,0,3,0,2429,0,0.,0.,0.,0,"{Asynchronous analysis concurrency}
        InterfCommands.html#InterfIndControl"},
                {"evaluation_concurrency",0x19,0,1,0,2421,0,0.,0.,0.,0,"{Asynchronous evaluation
        concurrency} InterfCommands.html#InterfIndControl"},
                {"local_evaluation_scheduling",8,2,2,0,2423,kw_17,0.,0.,0.,0,"{Local evaluation scheduling}
        InterfCommands.html#InterfIndControl"}
                }
```

GuiKeyWord kw_19[2] [static]

Initial value:

```
= {
                {"dynamic",8,0,1,1,2439},
                {"static",8,0,1,1,2441}
                }
```

GuiKeyWord kw_20[2] [static]

Initial value:

```
= {
                {"master",8,0,1,1,2435},
                {"peer",8,2,1,1,2437,kw_19,0.,0.,0.,0,"{Peer scheduling of evaluations}
        InterfCommands.html#InterfIndControl"}
                }
```

GuiKeyWord kw_21[9] [static]

Initial value:

```
= {
                {"algebraic_mappings",11,0,2,0,2341,0,0.,0.,0.,0,"{Algebraic mappings file}
        InterfCommands.html#InterfAlgebraic"},
                {"analysis_drivers",15,12,3,0,2343,kw_15,0.,0.,0.,0,"{Analysis drivers}
        InterfCommands.html#InterfApplic"},
                {"analysis_scheduling",8,2,9,0,2447,kw_16,0.,0.,0.,0,"{Message passing configuration for
        scheduling of analyses} InterfCommands.html#InterfIndControl"},
                {"analysis_servers",0x19,0,8,0,2445,0,0.,0.,0.,0,"{Number of analysis servers}
        InterfCommands.html#InterfIndControl"},
                {"asynchronous",8,3,4,0,2419,kw_18,0.,0.,0.,0,"{Asynchronous interface usage}
        InterfCommands.html#InterfIndControl"},
                {"evaluation_scheduling",8,2,6,0,2433,kw_20,0.,0.,0.,0,"{Message passing configuration for
        scheduling of evaluations} InterfCommands.html#InterfIndControl"},
                {"evaluation_servers",0x19,0,5,0,2431,0,0.,0.,0.,0,"{Number of evaluation servers}
        InterfCommands.html#InterfIndControl"},
                {"id_interface",11,0,1,0,2339,0,0.,0.,0.,0,"{Interface set identifier}
        InterfCommands.html#InterfIndControl"},
                {"processors_per_evaluation",0x19,0,7,0,2443,0,0.,0.,0.,0,"{Number of processors per
        evaluation server} InterfCommands.html#InterfIndControl"}
                }
```

GuiKeyWord kw_22[2] [static]

Initial value:

```
= {
                {"complementary",8,0,1,1,1193},
                {"cumulative",8,0,1,1,1191}
                }
```

GuiKeyWord kw_23[1] [static]

Initial value:

```
= {
                {"num_gen_reliability_levels",13,0,1,0,1201,0,0.,0.,0.,0,"{Number of generalized
        reliability levels} MethodCommands.html#MethodNonD"}
                }
```

GuiKeyWord kw_24[1] [static]

Initial value:

```
= {
                {"num_probability_levels",13,0,1,0,1197,0,0.,0.,0.,0,"{Number of probability levels}
        MethodCommands.html#MethodNonD"}
                }
```

GuiKeyWord kw_25[2] [static]

Initial value:

```
= {
                {"mt19937",8,0,1,1,1205},
                {"rnum2",8,0,1,1,1207}
                }
```

GuiKeyWord kw_26[4] [static]

Initial value:

```
= {
                {"constant_liar",8,0,1,1,1065},
                {"distance_penalty",8,0,1,1,1061},
                {"naive",8,0,1,1,1059},
                {"topology",8,0,1,1,1063}
                }
```

GuiKeyWord kw_27[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,1079},
                {"freeform",8,0,1,0,1081}
                }
```

GuiKeyWord kw_28[3] [static]

Initial value:

```
= {
                {"distance",8,0,1,1,1053},
                {"gradient",8,0,1,1,1055},
                {"predicted_variance",8,0,1,1,1051}
                }
```

GuiKeyWord kw_29[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,1075},
                {"annotated",8,0,1,0,1071},
                {"freeform",8,0,1,0,1073}
                }
```

GuiKeyWord kw_30[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,1097},
                {"series",8,0,1,1,1095}
                }
```

GuiKeyWord kw_31[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,1091},
                {"probabilities",8,0,1,1,1089},
                {"system",8,2,2,0,1093,kw_30}
                }
```

GuiKeyWord kw_32[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,1087,kw_31},
                {"num_response_levels",13,0,1,0,1085}
                }
```

GuiKeyWord kw_33[15] [static]

Initial value:

```
= {
                {"batch_selection",8,4,3,0,1057,kw_26,0.,0.,0.,0,"{Batch selection strategy}
        MethodCommands.html#MethodNonDAdaptive"},
                {"batch_size",9,0,4,0,1067,0,0.,0.,0.,0,"{Batch size (number of points added each
        iteration)} MethodCommands.html#MethodNonDAdaptive"},
                {"distribution",8,2,12,0,1189,kw_22,0.,0.,0.,0,"{Distribution type}
        MethodCommands.html#MethodNonD"},
                {"emulator_samples",9,0,1,0,1047,0,0.,0.,0.,0,"{Number of samples on the emulator to
        generate a new true sample each iteration} MethodCommands.html#MethodNonDAdaptive"},
                {"export_points_file",11,2,6,0,1077,kw_27,0.,0.,0.,0,"{File name for exporting
        approximation-based samples from evaluating the GP} MethodCommands.html#MethodNonDAdaptive"},
                {"fitness_metric",8,3,2,0,1049,kw_28,0.,0.,0.,0,"{Fitness metric}
        MethodCommands.html#MethodNonDAdaptive"},
                {"gen_reliability_levels",14,1,14,0,1199,kw_23,0.,0.,0.,0,"{Generalized reliability levels}
        MethodCommands.html#MethodNonD"},
                {"import_points_file",11,3,5,0,1069,kw_29,0.,0.,0.,0,"{File name for points to be imported
        as the basis for the initial GP} MethodCommands.html#MethodNonDAdaptive"},
                {"misc_options",15,0,8,0,1099},
                {"model_pointer",11,0,9,0,1667},
                {"probability_levels",14,1,13,0,1195,kw_24,0.,0.,0.,0,"{Probability levels}
        MethodCommands.html#MethodNonD"},
                {"response_levels",14,2,7,0,1083,kw_32},
                {"rng",8,2,15,0,1203,kw_25,0.,0.,0.,0,"{Random number generator}
        MethodCommands.html#MethodNonDMC"},
                {"samples",9,0,10,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC
       "},
                {"seed",0x19,0,11,0,1433,0,0.,0.,0.,0,"{Refinement seed}
        MethodCommands.html#MethodNonDLocalRel"}
                }
```

GuiKeyWord kw_34[7] [static]

Initial value:

```
= {
                {"merit1",8,0,1,1,335,0,0.,0.,0.,0,"[CHOOSE merit function]"},
                {"merit1_smooth",8,0,1,1,337},
                {"merit2",8,0,1,1,339},
                {"merit2_smooth",8,0,1,1,341,0,0.,0.,0.,0,"@"},
                {"merit2_squared",8,0,1,1,343},
                {"merit_max",8,0,1,1,331},
                {"merit_max_smooth",8,0,1,1,333}
                }
```

GuiKeyWord kw_35[2] [static]

Initial value:

```
= {
                {"blocking",8,0,1,1,325,0,0.,0.,0.,0,"[CHOOSE synchronization]"},
                {"nonblocking",8,0,1,1,327,0,0.,0.,0.,0,"@"}
                }
```

GuiKeyWord kw_36[19] [static]

Initial value:

```
= {
                {"constraint_penalty",10,0,7,0,345,0,0.,0.,0.,0,"{Constraint penalty}
        MethodCommands.html#MethodAPPSDC"},
                {"contraction_factor",10,0,2,0,317,0,0.,0.,0.,0,"{Pattern contraction factor}
        MethodCommands.html#MethodAPPSDC"},
                {"initial_delta",10,0,1,0,315,0,0.,0.,0.,0,"{Initial offset value}
        MethodCommands.html#MethodAPPSDC"},
                {"linear_equality_constraint_matrix",14,0,15,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,17,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,18,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,16,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,10,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,11,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,13,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,14,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,12,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"merit_function",8,7,6,0,329,kw_34,0.,0.,0.,0,"{Merit function}
        MethodCommands.html#MethodAPPSDC"},
                {"model_pointer",11,0,9,0,1667},
                {"smoothing_factor",10,0,8,0,347,0,0.,0.,0.,0,"{Smoothing factor}
        MethodCommands.html#MethodAPPSDC"},
                {"solution_accuracy",2,0,4,0,320},
                {"solution_target",10,0,4,0,321,0,0.,0.,0.,0,"{Solution target}
        MethodCommands.html#MethodAPPSDC"},
                {"synchronization",8,2,5,0,323,kw_35,0.,0.,0.,0,"{Evaluation synchronization}
        MethodCommands.html#MethodAPPSDC"},
                {"threshold_delta",10,0,3,0,319,0,0.,0.,0.,0,"{Threshold for offset values}
        MethodCommands.html#MethodAPPSDC"}
                }
```

GuiKeyWord kw_37[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,1359},
                {"freeform",8,0,1,0,1361}
                }
```

GuiKeyWord kw_38[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,1355},
                {"annotated",8,0,1,0,1351},
                {"freeform",8,0,1,0,1353}
                }
```

GuiKeyWord kw_39[5] [static]

Initial value:

```
= {
                {"dakota",8,0,1,1,1345},
                {"emulator_samples",9,0,2,0,1347},
                {"export_points_file",11,2,4,0,1357,kw_37},
                {"import_points_file",11,3,3,0,1349,kw_38},
                {"surfpack",8,0,1,1,1343}
                }
```

GuiKeyWord kw_40[1] [static]

Initial value:

```
= {
                {"sparse_grid_level",13,0,1,0,1365}
                }
```

GuiKeyWord kw_41[1] [static]

Initial value:

```
= {
                {"sparse_grid_level",13,0,1,0,1369}
                }
```

GuiKeyWord kw_42[4] [static]

Initial value:

```
= {
                {"gaussian_process",8,5,1,1,1341,kw_39},
                {"kriging",0,5,1,1,1340,kw_39},
                {"pce",8,1,1,1,1363,kw_40},
                {"sc",8,1,1,1,1367,kw_41}
                }
```

GuiKeyWord kw_43[6] [static]

Initial value:

```
= {
                {"chains",0x29,0,1,0,1329,0,3.,0.,0.,0,"{Number of chains}
        MethodCommands.html#MethodNonDBayesCalib"},
                {"crossover_chain_pairs",0x29,0,3,0,1333,0,0.,0.,0.,0,"{Number of chain pairs used in
        crossover } MethodCommands.html#MethodNonDBayesCalib"},
                {"emulator",8,4,6,0,1339,kw_42},
                {"gr_threshold",0x1a,0,4,0,1335,0,0.,0.,0.,0,"{Gelman-Rubin Threshold for convergence}
        MethodCommands.html#MethodNonDBayesCalib"},
                {"jump_step",0x29,0,5,0,1337,0,0.,0.,0.,0,"{Jump-Step }
        MethodCommands.html#MethodNonDBayesCalib"},
                {"num_cr",0x29,0,2,0,1331,0,1.,0.,0.,0,"{Number of candidate points used in burn-in
        adaptation} MethodCommands.html#MethodNonDBayesCalib"}
                }
```

GuiKeyWord kw_44[2] [static]

Initial value:

```
= {
                {"adaptive",8,0,1,1,1315},
                {"hastings",8,0,1,1,1313}
                }
```

GuiKeyWord kw_45[2] [static]

Initial value:

```
= {
                {"delayed",8,0,1,1,1309},
                {"standard",8,0,1,1,1307}
                }
```

GuiKeyWord kw_46[2] [static]

Initial value:

```
= {
                {"metropolis",8,2,2,0,1311,kw_44,0.,0.,0.,0,"{Metropolis type for the MCMC algorithm }
        MethodCommands.html#MethodNonDBayesCalib"},
                {"rejection",8,2,1,0,1305,kw_45}
                }
```

GuiKeyWord kw_47[2] [static]

Initial value:

```
= {
                {"dram",8,2,1,1,1303,kw_46},
                {"multilevel",8,0,1,1,1317}
                }
```

GuiKeyWord kw_48[2] [static]

Initial value:

```
= {
                {"mt19937",8,0,1,1,1321},
                {"rnum2",8,0,1,1,1323}
                }
```

GuiKeyWord kw_49[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,1297},
                {"freeform",8,0,1,0,1299}
                }
```

GuiKeyWord kw_50[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,1293},
                {"annotated",8,0,1,0,1289},
                {"freeform",8,0,1,0,1291}
                }
```

GuiKeyWord kw_51[6] [static]

Initial value:

```
= {
                {"emulator_samples",9,0,1,1,1285},
                {"export_points_file",11,2,3,0,1295,kw_49},
                {"import_points_file",11,3,2,0,1287,kw_50},
                {"mcmc_type",8,2,4,0,1301,kw_47},
                {"proposal_covariance_scale",14,0,6,0,1325,0,0.,0.,0.,0,"{Proposal covariance scaling}
        MethodCommands.html#MethodNonDBayesCalib"},
                {"rng",8,2,5,0,1319,kw_48,0.,0.,0.,0,"{Random seed generator}
        MethodCommands.html#MethodNonDBayesCalib"}
                }
```

GuiKeyWord kw_52[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,1271},
                {"freeform",8,0,1,0,1273}
                }
```

GuiKeyWord kw_53[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,1267},
                {"annotated",8,0,1,0,1263},
                {"freeform",8,0,1,0,1265}
                }
```

GuiKeyWord kw_54[5] [static]

Initial value:

```
= {
                {"dakota",8,0,1,1,1257},
                {"emulator_samples",9,0,2,0,1259},
                {"export_points_file",11,2,4,0,1269,kw_52},
                {"import_points_file",11,3,3,0,1261,kw_53},
                {"surfpack",8,0,1,1,1255}
                }
```

GuiKeyWord kw_55[1] [static]

Initial value:

```
= {
                {"sparse_grid_level",13,0,1,0,1277}
                }
```

GuiKeyWord kw_56[1] [static]

Initial value:

```
= {
                {"sparse_grid_level",13,0,1,0,1281}
                }
```

GuiKeyWord kw_57[4] [static]

Initial value:

```
= {
                {"gaussian_process",8,5,1,1,1253,kw_54},
                {"kriging",0,5,1,1,1252,kw_54},
                {"pce",8,1,1,1,1275,kw_55},
                {"sc",8,1,1,1,1279,kw_56}
                }
```

GuiKeyWord kw_58[4] [static]

Initial value:

```
= {
                {"emulator",8,4,1,0,1251,kw_57},
                {"mcmc_type",8,2,2,0,1301,kw_47},
                {"proposal_covariance_scale",14,0,4,0,1325,0,0.,0.,0.,0,"{Proposal covariance scaling}
        MethodCommands.html#MethodNonDBayesCalib"},
                {"rng",8,2,3,0,1319,kw_48,0.,0.,0.,0,"{Random seed generator}
        MethodCommands.html#MethodNonDBayesCalib"}
                }
```

GuiKeyWord kw_59[9] [static]

Initial value:

```
= {
                {"calibrate_sigma",8,0,4,0,1375,0,0.,0.,0.,0,"{Calibrate sigma flag}
        MethodCommands.html#MethodNonDBayesCalib"},
                {"dream",8,6,1,1,1327,kw_43},
                {"gpmsa",8,6,1,1,1283,kw_51},
                {"likelihood_scale",10,0,3,0,1373,0,0.,0.,0.,0,"{Likelihood scale factor}
        MethodCommands.html#MethodNonDBayesCalib"},
                {"model_pointer",11,0,5,0,1667},
                {"queso",8,4,1,1,1249,kw_58},
                {"samples",9,0,6,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
       },
                {"seed",0x19,0,7,0,1433,0,0.,0.,0.,0,"{Refinement seed}
        MethodCommands.html#MethodNonDLocalRel"},
                {"use_derivatives",8,0,2,0,1371}
                }
```

GuiKeyWord kw_60[4] [static]

Initial value:

```
= {
                {"deltas_per_variable",5,0,2,2,1650},
                {"model_pointer",11,0,3,0,1667},
                {"step_vector",14,0,1,1,1649,0,0.,0.,0.,0,"{Step vector} MethodCommands.html#MethodPSCPS"},
                {"steps_per_variable",13,0,2,2,1651,0,0.,0.,0.,0,"{Number of steps per variable}
        MethodCommands.html#MethodPSCPS"}
                }
```

GuiKeyWord kw_61[7] [static]

Initial value:

```
= {
                {"beta_solver_name",11,0,1,1,633},
                {"misc_options",15,0,6,0,641,0,0.,0.,0.,0,"{Specify miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"model_pointer",11,0,2,0,1667},
                {"seed",0x19,0,4,0,637,0,0.,0.,0.,0,"{Random seed for stochastic pattern search}
        MethodCommands.html#MethodSCOLIBPS"},
                {"show_misc_options",8,0,5,0,639,0,0.,0.,0.,0,"{Show miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"solution_accuracy",2,0,3,0,634},
                {"solution_target",10,0,3,0,635,0,0.,0.,0.,0,"{Desired solution target}
        MethodCommands.html#MethodSCOLIBDC"}
                }
```

GuiKeyWord kw_62[8] [static]

Initial value:

```
= {
                {"initial_delta",10,0,6,0,551,0,0.,0.,0.,0,"{Initial offset value}
        MethodCommands.html#MethodSCOLIBPS"},
                {"misc_options",15,0,5,0,641,0,0.,0.,0.,0,"{Specify miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"model_pointer",11,0,1,0,1667},
                {"seed",0x19,0,3,0,637,0,0.,0.,0.,0,"{Random seed for stochastic pattern search}
        MethodCommands.html#MethodSCOLIBPS"},
                {"show_misc_options",8,0,4,0,639,0,0.,0.,0.,0,"{Show miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"solution_accuracy",2,0,2,0,634},
                {"solution_target",10,0,2,0,635,0,0.,0.,0.,0,"{Desired solution target}
        MethodCommands.html#MethodSCOLIBDC"},
                {"threshold_delta",10,0,7,0,553,0,0.,0.,0.,0,"{Threshold for offset values}
        MethodCommands.html#MethodSCOLIBPS"}
                }
```

GuiKeyWord kw_63[2] [static]

Initial value:

```
= {
                {"all_dimensions",8,0,1,1,561},
                {"major_dimension",8,0,1,1,559}
                }
```

GuiKeyWord kw_64[12] [static]

Initial value:

```
= {
                {"constraint_penalty",10,0,6,0,571,0,0.,0.,0.,0,"{Constraint penalty}
        MethodCommands.html#MethodSCOLIBDIR"},
                {"division",8,2,1,0,557,kw_63,0.,0.,0.,0,"{Box subdivision approach}
        MethodCommands.html#MethodSCOLIBDIR"},
                {"global_balance_parameter",10,0,2,0,563,0,0.,0.,0.,0,"{Global search balancing parameter}
        MethodCommands.html#MethodSCOLIBDIR"},
                {"local_balance_parameter",10,0,3,0,565,0,0.,0.,0.,0,"{Local search balancing parameter}
        MethodCommands.html#MethodSCOLIBDIR"},
                {"max_boxsize_limit",10,0,4,0,567,0,0.,0.,0.,0,"{Maximum boxsize limit}
        MethodCommands.html#MethodSCOLIBDIR"},
                {"min_boxsize_limit",10,0,5,0,569,0,0.,0.,0.,0,"{Minimum boxsize limit}
```

```
MethodCommands.html#MethodSCOLIBDIR"},
        {"misc_options",15,0,11,0,641,0,0.,0.,0.,0,"{Specify miscellaneous options}
MethodCommands.html#MethodSCOLIBDC"},
        {"model_pointer",11,0,7,0,1667},
        {"seed",0x19,0,9,0,637,0,0.,0.,0.,0,"{Random seed for stochastic pattern search}
MethodCommands.html#MethodSCOLIBPS"},
        {"show_misc_options",8,0,10,0,639,0,0.,0.,0.,0,"{Show miscellaneous options}
MethodCommands.html#MethodSCOLIBDC"},
        {"solution_accuracy",2,0,8,0,634},
        {"solution_target",10,0,8,0,635,0,0.,0.,0.,0,"{Desired solution target}
MethodCommands.html#MethodSCOLIBDC"}
        }
```

GuiKeyWord kw_65[3] [static]

Initial value:

```
= {
        {"blend",8,0,1,1,607},
        {"two_point",8,0,1,1,605},
        {"uniform",8,0,1,1,609}
        }
```

GuiKeyWord kw_66[2] [static]

Initial value:

```
= {
        {"linear_rank",8,0,1,1,587},
        {"merit_function",8,0,1,1,589}
        }
```

GuiKeyWord kw_67[3] [static]

Initial value:

```
= {
        {"flat_file",11,0,1,1,583},
        {"simple_random",8,0,1,1,579},
        {"unique_random",8,0,1,1,581}
        }
```

GuiKeyWord kw_68[2] [static]

Initial value:

```
= {
        {"mutation_range",9,0,2,0,625,0,0.,0.,0.,0,"{Mutation range}
MethodCommands.html#MethodSCOLIBEA"},
        {"mutation_scale",10,0,1,0,623,0,0.,0.,0.,0,"{Mutation scale}
MethodCommands.html#MethodSCOLIBEA"}
        }
```

GuiKeyWord kw_69[5] [static]

Initial value:

```
= {
                {"non_adaptive",8,0,2,0,627,0,0.,0.,0.,0,"{Non-adaptive mutation flag}
        MethodCommands.html#MethodSCOLIBEA"},
                {"offset_cauchy",8,2,1,1,619,kw_68},
                {"offset_normal",8,2,1,1,617,kw_68},
                {"offset_uniform",8,2,1,1,621,kw_68},
                {"replace_uniform",8,0,1,1,615}
                }
```

GuiKeyWord kw_70[4] [static]

Initial value:

```
= {
                {"chc",9,0,1,1,595,0,0.,0.,0.,0,"{CHC replacement type} MethodCommands.html#MethodSCOLIBEA"
       },
                {"elitist",9,0,1,1,597,0,0.,0.,0.,0,"{Elitist replacement type}
        MethodCommands.html#MethodSCOLIBEA"},
                {"new_solutions_generated",9,0,2,0,599,0,0.,0.,0.,0,"{New solutions generated}
        MethodCommands.html#MethodSCOLIBEA"},
                {"random",9,0,1,1,593,0,0.,0.,0.,0,"{Random replacement type}
        MethodCommands.html#MethodSCOLIBEA"}
                }
```

GuiKeyWord kw_71[15] [static]

Initial value:

```
= {
                {"constraint_penalty",10,0,9,0,629},
                {"crossover_rate",10,0,5,0,601,0,0.,0.,0.,0,"{Crossover rate}
        MethodCommands.html#MethodSCOLIBEA"},
                {"crossover_type",8,3,6,0,603,kw_65,0.,0.,0.,0,"{Crossover type}
        MethodCommands.html#MethodSCOLIBEA"},
                {"fitness_type",8,2,3,0,585,kw_66,0.,0.,0.,0,"{Fitness type}
        MethodCommands.html#MethodSCOLIBEA"},
                {"initialization_type",8,3,2,0,577,kw_67,0.,0.,0.,0,"{Initialization type}
        MethodCommands.html#MethodSCOLIBEA"},
                {"misc_options",15,0,14,0,641,0,0.,0.,0.,0,"{Specify miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"model_pointer",11,0,10,0,1667},
                {"mutation_rate",10,0,7,0,611,0,0.,0.,0.,0,"{Mutation rate}
        MethodCommands.html#MethodSCOLIBEA"},
                {"mutation_type",8,5,8,0,613,kw_69,0.,0.,0.,0,"{Mutation type}
        MethodCommands.html#MethodSCOLIBEA"},
                {"population_size",0x19,0,1,0,575,0,0.,0.,0.,0,"{Number of population members}
        MethodCommands.html#MethodSCOLIBEA"},
                {"replacement_type",8,4,4,0,591,kw_70,0.,0.,0.,0,"{Replacement type}
        MethodCommands.html#MethodSCOLIBEA"},
                {"seed",0x19,0,12,0,637,0,0.,0.,0.,0,"{Random seed for stochastic pattern search}
        MethodCommands.html#MethodSCOLIBPS"},
                {"show_misc_options",8,0,13,0,639,0,0.,0.,0.,0,"{Show miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"solution_accuracy",2,0,11,0,634},
                {"solution_target",10,0,11,0,635,0,0.,0.,0.,0,"{Desired solution target}
        MethodCommands.html#MethodSCOLIBDC"}
                }
```

GuiKeyWord kw_72[3] [static]

Initial value:

```
= {
                {"adaptive_pattern",8,0,1,1,525},
                {"basic_pattern",8,0,1,1,527},
                {"multi_step",8,0,1,1,523}
                }
```

GuiKeyWord kw_73[2] [static]

Initial value:

```
= {
                {"coordinate",8,0,1,1,513},
                {"simplex",8,0,1,1,515}
                }
```

GuiKeyWord kw_74[2] [static]

Initial value:

```
= {
                {"blocking",8,0,1,1,531},
                {"nonblocking",8,0,1,1,533}
                }
```

GuiKeyWord kw_75[18] [static]

Initial value:

```
= {
                {"constant_penalty",8,0,1,0,505,0,0.,0.,0.,0,"{Control of dynamic penalty}
        MethodCommands.html#MethodSCOLIBPS"},
                {"constraint_penalty",10,0,17,0,547,0,0.,0.,0.,0,"{Constraint penalty}
        MethodCommands.html#MethodSCOLIBPS"},
                {"contraction_factor",10,0,16,0,545,0,0.,0.,0.,0,"{Pattern contraction factor}
        MethodCommands.html#MethodSCOLIBPS"},
                {"expand_after_success",9,0,3,0,509,0,0.,0.,0.,0,"{Number of consecutive improvements
        before expansion} MethodCommands.html#MethodSCOLIBPS"},
                {"exploratory_moves",8,3,7,0,521,kw_72,0.,0.,0.,0,"{Exploratory moves selection}
        MethodCommands.html#MethodSCOLIBPS"},
                {"initial_delta",10,0,14,0,551,0,0.,0.,0.,0,"{Initial offset value}
        MethodCommands.html#MethodSCOLIBPS"},
                {"misc_options",15,0,13,0,641,0,0.,0.,0.,0,"{Specify miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"model_pointer",11,0,9,0,1667},
                {"no_expansion",8,0,2,0,507,0,0.,0.,0.,0,"{No expansion flag}
        MethodCommands.html#MethodSCOLIBPS"},
                {"pattern_basis",8,2,4,0,511,kw_73,0.,0.,0.,0,"{Pattern basis selection}
        MethodCommands.html#MethodSCOLIBPS"},
                {"seed",0x19,0,11,0,637,0,0.,0.,0.,0,"{Random seed for stochastic pattern search}
        MethodCommands.html#MethodSCOLIBPS"},
                {"show_misc_options",8,0,12,0,639,0,0.,0.,0.,0,"{Show miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"solution_accuracy",2,0,10,0,634},
                {"solution_target",10,0,10,0,635,0,0.,0.,0.,0,"{Desired solution target}
        MethodCommands.html#MethodSCOLIBDC"},
                {"stochastic",8,0,5,0,517,0,0.,0.,0.,0,"{Stochastic pattern search}
        MethodCommands.html#MethodSCOLIBPS"},
                {"synchronization",8,2,8,0,529,kw_74,0.,0.,0.,0,"{Evaluation synchronization}
        MethodCommands.html#MethodSCOLIBPS"},
                {"threshold_delta",10,0,15,0,553,0,0.,0.,0.,0,"{Threshold for offset values}
        MethodCommands.html#MethodSCOLIBPS"},
                {"total_pattern_size",9,0,6,0,519,0,0.,0.,0.,0,"{Total number of points in pattern}
        MethodCommands.html#MethodSCOLIBPS"}
                }
```

GuiKeyWord kw_76[14] [static]

Initial value:

```
= {
                {"constant_penalty",8,0,4,0,543,0,0.,0.,0.,0,"{Control of dynamic penalty}
        MethodCommands.html#MethodSCOLIBSW"},
                {"constraint_penalty",10,0,13,0,547,0,0.,0.,0.,0,"{Constraint penalty}
        MethodCommands.html#MethodSCOLIBPS"},
                {"contract_after_failure",9,0,1,0,537,0,0.,0.,0.,0,"{Number of consecutive failures before
        contraction} MethodCommands.html#MethodSCOLIBSW"},
                {"contraction_factor",10,0,12,0,545,0,0.,0.,0.,0,"{Pattern contraction factor}
        MethodCommands.html#MethodSCOLIBPS"},
                {"expand_after_success",9,0,3,0,541,0,0.,0.,0.,0,"{Number of consecutive improvements
        before expansion} MethodCommands.html#MethodSCOLIBSW"},
                {"initial_delta",10,0,10,0,551,0,0.,0.,0.,0,"{Initial offset value}
        MethodCommands.html#MethodSCOLIBPS"},
                {"misc_options",15,0,9,0,641,0,0.,0.,0.,0,"{Specify miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"model_pointer",11,0,5,0,1667},
                {"no_expansion",8,0,2,0,539,0,0.,0.,0.,0,"{No expansion flag}
        MethodCommands.html#MethodSCOLIBSW"},
                {"seed",0x19,0,7,0,637,0,0.,0.,0.,0,"{Random seed for stochastic pattern search}
        MethodCommands.html#MethodSCOLIBPS"},
                {"show_misc_options",8,0,8,0,639,0,0.,0.,0.,0,"{Show miscellaneous options}
        MethodCommands.html#MethodSCOLIBDC"},
                {"solution_accuracy",2,0,6,0,634},
                {"solution_target",10,0,6,0,635,0,0.,0.,0.,0,"{Desired solution target}
        MethodCommands.html#MethodSCOLIBDC"},
                {"threshold_delta",10,0,11,0,553,0,0.,0.,0.,0,"{Threshold for offset values}
        MethodCommands.html#MethodSCOLIBPS"}
                }
```

GuiKeyWord kw_77[12] [static]

Initial value:

```
= {
                {"frcg",8,0,1,1,249},
                {"linear_equality_constraint_matrix",14,0,8,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,10,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,11,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,9,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,3,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,4,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,6,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,7,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,5,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"mfd",8,0,1,1,251},
                {"model_pointer",11,0,2,0,1667}
                }
```

GuiKeyWord kw_78[10] [static]

Initial value:

```
= {
                {"linear_equality_constraint_matrix",14,0,7,0,495,0,0.,0.,0.,0,"{Linear equality
```

```
coefficient matrix} MethodCommands.html#MethodMin"},
         {"linear_equality_scale_types",15,0,9,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
MethodCommands.html#MethodMin"},
         {"linear_equality_scales",14,0,10,0,501,0,0.,0.,0.,0,"{Linear equality scales}
MethodCommands.html#MethodMin"},
         {"linear_equality_targets",14,0,8,0,497,0,0.,0.,0.,0,"{Linear equality targets}
MethodCommands.html#MethodMin"},
         {"linear_inequality_constraint_matrix",14,0,2,0,485,0,0.,0.,0.,0,"{Linear inequality
coefficient matrix} MethodCommands.html#MethodMin"},
         {"linear_inequality_lower_bounds",14,0,3,0,487,0,0.,0.,0.,0,"{Linear inequality lower
bounds} MethodCommands.html#MethodMin"},
         {"linear_inequality_scale_types",15,0,5,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
types} MethodCommands.html#MethodMin"},
         {"linear_inequality_scales",14,0,6,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
MethodCommands.html#MethodMin"},
         {"linear_inequality_upper_bounds",14,0,4,0,489,0,0.,0.,0.,0,"{Linear inequality upper
bounds} MethodCommands.html#MethodMin"},
         {"model_pointer",11,0,1,0,1667}
         }
```

GuiKeyWord kw_79[1] [static]

Initial value:

```
= {
         {"drop_tolerance",10,0,1,0,1399}
         }
```

GuiKeyWord kw_80[15] [static]

Initial value:

```
= {
         {"box_behnken",8,0,1,1,1389,0,0.,0.,0.,0,"[CHOOSE DACE type]"},
         {"central_composite",8,0,1,1,1391},
         {"fixed_seed",8,0,5,0,1401,0,0.,0.,0.,0,"{Fixed seed flag} MethodCommands.html#MethodDDACE"
    },
         {"grid",8,0,1,1,1379},
         {"lhs",8,0,1,1,1385},
         {"main_effects",8,0,2,0,1393,0,0.,0.,0.,0,"{Main effects} MethodCommands.html#MethodDDACE"}
    ,
         {"model_pointer",11,0,7,0,1667},
         {"oa_lhs",8,0,1,1,1387},
         {"oas",8,0,1,1,1383},
         {"quality_metrics",8,0,3,0,1395,0,0.,0.,0.,0,"{Quality metrics}
MethodCommands.html#MethodDDACE"},
         {"random",8,0,1,1,1381},
         {"samples",9,0,8,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
    },
         {"seed",0x19,0,9,0,1433,0,0.,0.,0.,0,"{Refinement seed}
MethodCommands.html#MethodNonDLocalRel"},
         {"symbols",9,0,6,0,1403,0,0.,0.,0.,0,"{Number of symbols} MethodCommands.html#MethodDDACE"}
    ,
         {"variance_based_decomp",8,1,4,0,1397,kw_79,0.,0.,0.,0,"{Variance based decomposition}
MethodCommands.html#MethodDDACE"}
         }
```

GuiKeyWord kw_81[15] [static]

Initial value:

```
= {
         {"bfgs",8,0,1,1,237},
         {"frcg",8,0,1,1,233},
         {"linear_equality_constraint_matrix",14,0,8,0,495,0,0.,0.,0.,0,"{Linear equality
```

```
coefficient matrix} MethodCommands.html#MethodMin"},
         {"linear_equality_scale_types",15,0,10,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
MethodCommands.html#MethodMin"},
         {"linear_equality_scales",14,0,11,0,501,0,0.,0.,0.,0,"{Linear equality scales}
MethodCommands.html#MethodMin"},
         {"linear_equality_targets",14,0,9,0,497,0,0.,0.,0.,0,"{Linear equality targets}
MethodCommands.html#MethodMin"},
         {"linear_inequality_constraint_matrix",14,0,3,0,485,0,0.,0.,0.,0,"{Linear inequality
coefficient matrix} MethodCommands.html#MethodMin"},
         {"linear_inequality_lower_bounds",14,0,4,0,487,0,0.,0.,0.,0,"{Linear inequality lower
bounds} MethodCommands.html#MethodMin"},
         {"linear_inequality_scale_types",15,0,6,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
types} MethodCommands.html#MethodMin"},
         {"linear_inequality_scales",14,0,7,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
MethodCommands.html#MethodMin"},
         {"linear_inequality_upper_bounds",14,0,5,0,489,0,0.,0.,0.,0,"{Linear inequality upper
bounds} MethodCommands.html#MethodMin"},
         {"mmfd",8,0,1,1,235},
         {"model_pointer",11,0,2,0,1667},
         {"slp",8,0,1,1,239},
         {"sqp",8,0,1,1,241}
         }
```

GuiKeyWord kw_82[2] [static]

Initial value:

```
= {
         {"annotated",8,0,1,0,701},
         {"freeform",8,0,1,0,703}
         }
```

GuiKeyWord kw_83[2] [static]

Initial value:

```
= {
         {"dakota",8,0,1,1,687},
         {"surfpack",8,0,1,1,685}
         }
```

GuiKeyWord kw_84[3] [static]

Initial value:

```
= {
         {"active_only",8,0,2,0,697},
         {"annotated",8,0,1,0,693},
         {"freeform",8,0,1,0,695}
         }
```

GuiKeyWord kw_85[7] [static]

Initial value:

```
= {
         {"export_points_file",11,2,4,0,699,kw_82,0.,0.,0.,0,"{File name for exporting
approximation-based samples from evaluating the GP} MethodCommands.html#MethodEG"},
         {"gaussian_process",8,2,1,0,683,kw_83,0.,0.,0.,0,"{GP selection}
MethodCommands.html#MethodEG"},
         {"import_points_file",11,3,3,0,691,kw_84,0.,0.,0.,0,"{File name for points to be imported
as the basis for the initial GP} MethodCommands.html#MethodEG"},
```

```
                {"kriging",0,2,1,0,682,kw_83},
                {"model_pointer",11,0,6,0,1667},
                {"seed",0x19,0,5,0,705,0,0.,0.,0.,0,"{Random seed} MethodCommands.html#MethodEG"},
                {"use_derivatives",8,0,2,0,689,0,0.,0.,0.,0,"{Derivative usage}
        MethodCommands.html#MethodEG"}
                }
```

GuiKeyWord kw_86[9] [static]

Initial value:

```
= {
                {"batch_size",9,0,2,0,1139},
                {"distribution",8,2,6,0,1189,kw_22,0.,0.,0.,0,"{Distribution type}
        MethodCommands.html#MethodNonD"},
                {"emulator_samples",9,0,1,0,1137},
                {"gen_reliability_levels",14,1,8,0,1199,kw_23,0.,0.,0.,0,"{Generalized reliability levels}
        MethodCommands.html#MethodNonD"},
                {"model_pointer",11,0,3,0,1667},
                {"probability_levels",14,1,7,0,1195,kw_24,0.,0.,0.,0,"{Probability levels}
        MethodCommands.html#MethodNonD"},
                {"rng",8,2,9,0,1203,kw_25,0.,0.,0.,0,"{Random number generator}
        MethodCommands.html#MethodNonDMC"},
                {"samples",9,0,4,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
       },
                {"seed",0x19,0,5,0,1433,0,0.,0.,0.,0,"{Refinement seed}
        MethodCommands.html#MethodNonDLocalRel"}
                }
```

GuiKeyWord kw_87[3] [static]

Initial value:

```
= {
                {"grid",8,0,1,1,1419,0,0.,0.,0.,0,"[CHOOSE trial type]"},
                {"halton",8,0,1,1,1421},
                {"random",8,0,1,1,1423,0,0.,0.,0.,0,"@"}
                }
```

GuiKeyWord kw_88[1] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,1,0,1413}
                }
```

GuiKeyWord kw_89[9] [static]

Initial value:

```
= {
                {"fixed_seed",8,0,4,0,1415,0,0.,0.,0.,0,"{Fixed seed flag}
        MethodCommands.html#MethodFSUDACE"},
                {"latinize",8,0,1,0,1407,0,0.,0.,0.,0,"{Latinization of samples}
        MethodCommands.html#MethodFSUDACE"},
                {"model_pointer",11,0,7,0,1667},
                {"num_trials",9,0,6,0,1425,0,0.,0.,0.,0,"{Number of trials  }
        MethodCommands.html#MethodFSUDACE"},
                {"quality_metrics",8,0,2,0,1409,0,0.,0.,0.,0,"{Quality metrics}
        MethodCommands.html#MethodFSUDACE"},
                {"samples",9,0,8,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
```

```
},
          {"seed",0x19,0,9,0,1433,0,0.,0.,0.,0,"{Refinement seed}
 MethodCommands.html#MethodNonDLocalRel"},
          {"trial_type",8,3,5,0,1417,kw_87,0.,0.,0.,0,"{Trial type} MethodCommands.html#MethodFSUDACE
"},
          {"variance_based_decomp",8,1,3,0,1411,kw_88,0.,0.,0.,0,"{Variance based decomposition}
 MethodCommands.html#MethodFSUDACE"}
          }
```

GuiKeyWord kw_90[1] `[static]`

Initial value:

```
= {
          {"drop_tolerance",10,0,1,0,1615}
          }
```

GuiKeyWord kw_91[11] `[static]`

Initial value:

```
= {
          {"fixed_sequence",8,0,6,0,1619,0,0.,0.,0.,0,"{Fixed sequence flag}
 MethodCommands.html#MethodFSUDACE"},
          {"halton",8,0,1,1,1605,0,0.,0.,0.,0,"[CHOOSE sequence type]"},
          {"hammersley",8,0,1,1,1607},
          {"latinize",8,0,2,0,1609,0,0.,0.,0.,0,"{Latinization of samples}
 MethodCommands.html#MethodFSUDACE"},
          {"model_pointer",11,0,10,0,1667},
          {"prime_base",13,0,9,0,1625,0,0.,0.,0.,0,"{Prime bases for sequences}
 MethodCommands.html#MethodFSUDACE"},
          {"quality_metrics",8,0,3,0,1611,0,0.,0.,0.,0,"{Quality metrics}
 MethodCommands.html#MethodFSUDACE"},
          {"samples",9,0,5,0,1617,0,0.,0.,0.,0,"{Number of samples taken in the MCMC sampling}
 MethodCommands.html#MethodNonDBayesCalib"},
          {"sequence_leap",13,0,8,0,1623,0,0.,0.,0.,0,"{Sequence leaping indices}
 MethodCommands.html#MethodFSUDACE"},
          {"sequence_start",13,0,7,0,1621,0,0.,0.,0.,0,"{Sequence starting indices}
 MethodCommands.html#MethodFSUDACE"},
          {"variance_based_decomp",8,1,4,0,1613,kw_90,0.,0.,0.,0,"{Variance based decomposition}
 MethodCommands.html#MethodFSUDACE"}
          }
```

GuiKeyWord kw_92[2] `[static]`

Initial value:

```
= {
          {"annotated",8,0,1,0,1025},
          {"freeform",8,0,1,0,1027}
          }
```

GuiKeyWord kw_93[3] `[static]`

Initial value:

```
= {
          {"active_only",8,0,2,0,1021},
          {"annotated",8,0,1,0,1017},
          {"freeform",8,0,1,0,1019}
          }
```

GuiKeyWord kw_94[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,1043},
                {"series",8,0,1,1,1041}
                }
```

GuiKeyWord kw_95[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,1037},
                {"probabilities",8,0,1,1,1035},
                {"system",8,2,2,0,1039,kw_94}
                }
```

GuiKeyWord kw_96[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,1033,kw_95},
                {"num_response_levels",13,0,1,0,1031}
                }
```

GuiKeyWord kw_97[11] [static]

Initial value:

```
= {
                {"distribution",8,2,8,0,1189,kw_22,0.,0.,0.,0,"{Distribution type}
        MethodCommands.html#MethodNonD"},
                {"emulator_samples",9,0,1,0,1013},
                {"export_points_file",11,2,3,0,1023,kw_92,0.,0.,0.,0,"{File name for exporting
        approximation-based samples from evaluating the emulator} MethodCommands.html#MethodNonDBayesCalib"},
                {"gen_reliability_levels",14,1,10,0,1199,kw_23,0.,0.,0.,0,"{Generalized reliability levels}
        MethodCommands.html#MethodNonD"},
                {"import_points_file",11,3,2,0,1015,kw_93,0.,0.,0.,0,"{File name for points to be imported
        as the basis for the initial emulator} MethodCommands.html#MethodNonDBayesCalib"},
                {"model_pointer",11,0,5,0,1667},
                {"probability_levels",14,1,9,0,1195,kw_24,0.,0.,0.,0,"{Probability levels}
        MethodCommands.html#MethodNonD"},
                {"response_levels",14,2,4,0,1029,kw_96},
                {"rng",8,2,11,0,1203,kw_25,0.,0.,0.,0,"{Random number generator}
        MethodCommands.html#MethodNonDMC"},
                {"samples",9,0,6,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
       },
                {"seed",0x19,0,7,0,1433,0,0.,0.,0.,0,"{Refinement seed}
        MethodCommands.html#MethodNonDLocalRel"}
                }
```

GuiKeyWord kw_98[2] [static]

Initial value:

```
= {
                {"model_pointer",11,0,2,0,1667},
                {"seed",0x19,0,1,0,679,0,0.,0.,0.,0,"{Random seed} MethodCommands.html#MethodNonDMC"}
                }
```

GuiKeyWord kw_99[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,1187},
                {"series",8,0,1,1,1185}
                }
```

GuiKeyWord kw_100[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,1181},
                {"probabilities",8,0,1,1,1179},
                {"system",8,2,2,0,1183,kw_99}
                }
```

GuiKeyWord kw_101[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,1177,kw_100},
                {"num_response_levels",13,0,1,0,1175}
                }
```

GuiKeyWord kw_102[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,1165},
                {"freeform",8,0,1,0,1167}
                }
```

GuiKeyWord kw_103[2] [static]

Initial value:

```
= {
                {"dakota",8,0,1,1,1151},
                {"surfpack",8,0,1,1,1149}
                }
```

GuiKeyWord kw_104[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,1161},
                {"annotated",8,0,1,0,1157},
                {"freeform",8,0,1,0,1159}
                }
```

GuiKeyWord kw_105[5] [static]

Initial value:

```
= {
                {"export_points_file",11,2,4,0,1163,kw_102},
                {"gaussian_process",8,2,1,0,1147,kw_103},
                {"import_points_file",11,3,3,0,1155,kw_104,0.,0.,0.,0,"{File containing points to evaluate}
        MethodCommands.html#MethodPSLPS"},
                {"kriging",0,2,1,0,1146,kw_103},
                {"use_derivatives",8,0,2,0,1153}
                }
```

GuiKeyWord kw_106[12] [static]

Initial value:

```
= {
                {"distribution",8,2,6,0,1189,kw_22,0.,0.,0.,0,"{Distribution type}
        MethodCommands.html#MethodNonD"},
                {"ea",8,0,1,0,1169},
                {"ego",8,5,1,0,1145,kw_105},
                {"gen_reliability_levels",14,1,8,0,1199,kw_23,0.,0.,0.,0,"{Generalized reliability levels}
        MethodCommands.html#MethodNonD"},
                {"lhs",8,0,1,0,1171},
                {"model_pointer",11,0,3,0,1667},
                {"probability_levels",14,1,7,0,1195,kw_24,0.,0.,0.,0,"{Probability levels}
        MethodCommands.html#MethodNonD"},
                {"response_levels",14,2,2,0,1173,kw_101},
                {"rng",8,2,9,0,1203,kw_25,0.,0.,0.,0,"{Random number generator}
        MethodCommands.html#MethodNonDMC"},
                {"samples",9,0,4,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
       },
                {"sbo",8,5,1,0,1143,kw_105},
                {"seed",0x19,0,5,0,1433,0,0.,0.,0.,0,"{Refinement seed}
        MethodCommands.html#MethodNonDLocalRel"}
                }
```

GuiKeyWord kw_107[2] [static]

Initial value:

```
= {
                {"mt19937",8,0,1,1,1243},
                {"rnum2",8,0,1,1,1245}
                }
```

GuiKeyWord kw_108[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,1233},
                {"freeform",8,0,1,0,1235}
                }
```

GuiKeyWord kw_109[2] [static]

Initial value:

```
= {
                {"dakota",8,0,1,1,1219},
                {"surfpack",8,0,1,1,1217}
                }
```

GuiKeyWord kw_110[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,1229},
                {"annotated",8,0,1,0,1225},
                {"freeform",8,0,1,0,1227}
                }
```

GuiKeyWord kw_111[5] [static]

Initial value:

```
= {
                {"export_points_file",11,2,4,0,1231,kw_108,0.,0.,0.,0,"{File name for exporting
        approximation-based samples from evaluating the GP} MethodCommands.html#MethodNonDGlobalIntervalEst"},
                {"gaussian_process",8,2,1,0,1215,kw_109,0.,0.,0.,0,"{EGO GP selection}
        MethodCommands.html#MethodNonDGlobalIntervalEst"},
                {"import_points_file",11,3,3,0,1223,kw_110,0.,0.,0.,0,"{File name for points to be imported
        as the basis for the initial GP} MethodCommands.html#MethodNonDGlobalIntervalEst"},
                {"kriging",0,2,1,0,1214,kw_109},
                {"use_derivatives",8,0,2,0,1221,0,0.,0.,0.,0,"{Derivative usage}
        MethodCommands.html#MethodNonDGlobalIntervalEst"}
                }
```

GuiKeyWord kw_112[8] [static]

Initial value:

```
= {
                {"ea",8,0,1,0,1237},
                {"ego",8,5,1,0,1213,kw_111},
                {"lhs",8,0,1,0,1239},
                {"model_pointer",11,0,3,0,1667},
                {"rng",8,2,2,0,1241,kw_107,0.,0.,0.,0,"{Random seed generator}
        MethodCommands.html#MethodNonDGlobalIntervalEst"},
                {"samples",9,0,4,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
       },
                {"sbo",8,5,1,0,1211,kw_111},
                {"seed",0x19,0,5,0,1433,0,0.,0.,0.,0,"{Refinement seed}
        MethodCommands.html#MethodNonDLocalRel"}
                }
```

GuiKeyWord kw_113[2] [static]

Initial value:

```
= {
                {"complementary",8,0,1,1,1593},
                {"cumulative",8,0,1,1,1591}
                }
```

GuiKeyWord kw_114[1] [static]

Initial value:

```
= {
                {"num_gen_reliability_levels",13,0,1,0,1601}
                }
```

GuiKeyWord kw_115[1] [static]

Initial value:

```
= {
                {"num_probability_levels",13,0,1,0,1597}
                }
```

GuiKeyWord kw_116[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,1559},
                {"freeform",8,0,1,0,1561}
                }
```

GuiKeyWord kw_117[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,1555},
                {"annotated",8,0,1,0,1551},
                {"freeform",8,0,1,0,1553}
                }
```

GuiKeyWord kw_118[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,1587},
                {"series",8,0,1,1,1585}
                }
```

GuiKeyWord kw_119[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,1581},
                {"probabilities",8,0,1,1,1579},
                {"system",8,2,2,0,1583,kw_118}
                }
```

GuiKeyWord kw_120[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,1577,kw_119},
                {"num_response_levels",13,0,1,0,1575}
                }
```

GuiKeyWord kw_121[2] [static]

Initial value:

```
= {
                {"mt19937",8,0,1,1,1569},
                {"rnum2",8,0,1,1,1571}
                }
```

GuiKeyWord kw_122[16] [static]

Initial value:

```
= {
                {"dakota",8,0,2,0,1547},
                {"distribution",8,2,10,0,1589,kw_113},
                {"export_points_file",11,2,4,0,1557,kw_116,0.,0.,0.,0,"{File name for exporting
        approximation-based samples from evaluating the GP} MethodCommands.html#MethodNonDGlobalRel"},
                {"gen_reliability_levels",14,1,12,0,1599,kw_114},
                {"import_points_file",11,3,3,0,1549,kw_117,0.,0.,0.,0,"{File name for points to be imported
        as the basis for the initial GP} MethodCommands.html#MethodNonDGlobalRel"},
                {"model_pointer",11,0,9,0,1667},
                {"probability_levels",14,1,11,0,1595,kw_115},
                {"response_levels",14,2,8,0,1573,kw_120},
                {"rng",8,2,7,0,1567,kw_121},
                {"seed",0x19,0,6,0,1565,0,0.,0.,0.,0,"{Random seed for initial GP construction}
        MethodCommands.html#MethodNonDGlobalRel"},
                {"surfpack",8,0,2,0,1545},
                {"u_gaussian_process",8,0,1,1,1543},
                {"u_kriging",0,0,1,1,1542},
                {"use_derivatives",8,0,5,0,1563,0,0.,0.,0.,0,"{Derivative usage}
        MethodCommands.html#MethodNonDGlobalRel"},
                {"x_gaussian_process",8,0,1,1,1541},
                {"x_kriging",0,0,1,1,1540}
                }
```

GuiKeyWord kw_123[2] [static]

Initial value:

```
= {
                {"master",8,0,1,1,143},
                {"peer",8,0,1,1,145}
                }
```

GuiKeyWord kw_124[1] [static]

Initial value:

```
= {
                {"model_pointer_list",11,0,1,0,107,0,0.,0.,0.,0,"{List of model pointers}
        MethodCommands.html#MethodMetaHybrid"}
                }
```

GuiKeyWord kw_125[2] [static]

Initial value:

```
= {
                {"method_name_list",15,1,1,1,105,kw_124,0.,0.,0.,0,"{List of method names}
        MethodCommands.html#MethodMetaHybrid"},
                {"method_pointer_list",15,0,1,1,109,0,0.,0.,0.,0,"{List of method pointers}
        MethodCommands.html#MethodMetaHybrid"}
                }
```

GuiKeyWord kw_126[1] [static]

Initial value:

```
= {
                {"global_model_pointer",11,0,1,0,91,0,0.,0.,0.,0,"{Pointer to the global model
        specification} MethodCommands.html#MethodMetaHybrid"}
                }
```

GuiKeyWord kw_127[1] [static]

Initial value:

```
= {
                {"local_model_pointer",11,0,1,0,97,0,0.,0.,0.,0,"{Pointer to the local model specification}
        MethodCommands.html#MethodMetaHybrid"}
                }
```

GuiKeyWord kw_128[5] [static]

Initial value:

```
= {
                {"global_method_name",11,1,1,1,89,kw_126,0.,0.,0.,0,"{Name of the global method}
        MethodCommands.html#MethodMetaHybrid"},
                {"global_method_pointer",11,0,1,1,93,0,0.,0.,0.,0,"{Pointer to the global method
        specification} MethodCommands.html#MethodMetaHybrid"},
                {"local_method_name",11,1,2,2,95,kw_127,0.,0.,0.,0,"{Name of the local method}
        MethodCommands.html#MethodMetaHybrid"},
                {"local_method_pointer",11,0,2,2,99,0,0.,0.,0.,0,"{Pointer to the local method
        specification} MethodCommands.html#MethodMetaHybrid"},
                {"local_search_probability",10,0,3,0,101,0,0.,0.,0.,0,"{Probability of executing local
        searches} MethodCommands.html#MethodMetaHybrid"}
                }
```

GuiKeyWord kw_129[1] [static]

Initial value:

```
= {
                {"model_pointer_list",11,0,1,0,83,0,0.,0.,0.,0,"{List of model pointers}
        MethodCommands.html#MethodMetaHybrid"}
                }
```

GuiKeyWord kw_130[2] [static]

Initial value:

```
= {
                {"method_name_list",15,1,1,1,81,kw_129,0.,0.,0.,0,"{List of method names}
        MethodCommands.html#MethodMetaHybrid"},
                {"method_pointer_list",15,0,1,1,85,0,0.,0.,0.,0,"{List of method pointers}
        MethodCommands.html#MethodMetaHybrid"}
                }
```

GuiKeyWord kw_131[8] [static]

Initial value:

```
= {
                {"collaborative",8,2,1,1,103,kw_125,0.,0.,0.,0,"{Collaborative hybrid}
        MethodCommands.html#MethodMetaHybrid"},
                {"coupled",0,5,1,1,86,kw_128},
                {"embedded",8,5,1,1,87,kw_128,0.,0.,0.,0,"{Embedded hybrid}
        MethodCommands.html#MethodMetaHybrid"},
                {"iterator_scheduling",8,2,3,0,141,kw_123,0.,0.,0.,0,"{Message passing configuration for
        scheduling of iterator jobs} MethodCommands.html#MethodMeta"},
                {"iterator_servers",0x19,0,2,0,139,0,0.,0.,0.,0,"{Number of iterator servers}
        MethodCommands.html#MethodMeta"},
                {"processors_per_iterator",0x19,0,4,0,147,0,0.,0.,0.,0,"{Number of processors per iterator
        server} MethodCommands.html#MethodMeta"},
                {"sequential",8,2,1,1,79,kw_130,0.,0.,0.,0,"{Sequential hybrid}
        MethodCommands.html#MethodMetaHybrid"},
                {"uncoupled",0,2,1,1,78,kw_130}
                }
```

GuiKeyWord kw_132[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,1009},
                {"series",8,0,1,1,1007}
                }
```

GuiKeyWord kw_133[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,1003},
                {"probabilities",8,0,1,1,1001},
                {"system",8,2,2,0,1005,kw_132}
                }
```

GuiKeyWord kw_134[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,999,kw_133},
                {"num_response_levels",13,0,1,0,997}
                }
```

GuiKeyWord kw_135[12] [static]

Initial value:

```
= {
                {"adapt_import",8,0,1,1,989},
                {"distribution",8,2,7,0,1189,kw_22,0.,0.,0.,0,"{Distribution type}
        MethodCommands.html#MethodNonD"},
                {"gen_reliability_levels",14,1,9,0,1199,kw_23,0.,0.,0.,0,"{Generalized reliability levels}
        MethodCommands.html#MethodNonD"},
                {"import",8,0,1,1,987},
                {"mm_adapt_import",8,0,1,1,991},
                {"model_pointer",11,0,4,0,1667},
```

```
            {"probability_levels",14,1,8,0,1195,kw_24,0.,0.,0.,0,"{Probability levels}
 MethodCommands.html#MethodNonD"},
            {"refinement_samples",9,0,2,0,993},
            {"response_levels",14,2,3,0,995,kw_134},
            {"rng",8,2,10,0,1203,kw_25,0.,0.,0.,0,"{Random number generator}
 MethodCommands.html#MethodNonDMC"},
            {"samples",9,0,5,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
},
            {"seed",0x19,0,6,0,1433,0,0.,0.,0.,0,"{Refinement seed}
 MethodCommands.html#MethodNonDLocalRel"}
            }
```

GuiKeyWord kw_136[3] [static]

Initial value:

```
= {
            {"active_only",8,0,2,0,1645},
            {"annotated",8,0,1,0,1641},
            {"freeform",8,0,1,0,1643}
            }
```

GuiKeyWord kw_137[3] [static]

Initial value:

```
= {
            {"import_points_file",11,3,1,1,1639,kw_136},
            {"list_of_points",14,0,1,1,1637,0,0.,0.,0.,0,"{List of points to evaluate}
 MethodCommands.html#MethodPSLPS"},
            {"model_pointer",11,0,2,0,1667}
            }
```

GuiKeyWord kw_138[2] [static]

Initial value:

```
= {
            {"complementary",8,0,1,1,1469},
            {"cumulative",8,0,1,1,1467}
            }
```

GuiKeyWord kw_139[1] [static]

Initial value:

```
= {
            {"num_gen_reliability_levels",13,0,1,0,1463}
            }
```

GuiKeyWord kw_140[1] [static]

Initial value:

```
= {
            {"num_probability_levels",13,0,1,0,1459}
            }
```

GuiKeyWord kw_141[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,1455},
                {"series",8,0,1,1,1453}
                }
```

GuiKeyWord kw_142[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,1449},
                {"probabilities",8,0,1,1,1447},
                {"system",8,2,2,0,1451,kw_141}
                }
```

GuiKeyWord kw_143[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,1445,kw_142},
                {"num_response_levels",13,0,1,0,1443}
                }
```

GuiKeyWord kw_144[7] [static]

Initial value:

```
= {
                {"distribution",8,2,5,0,1465,kw_138},
                {"gen_reliability_levels",14,1,4,0,1461,kw_139},
                {"model_pointer",11,0,6,0,1667},
                {"nip",8,0,1,0,1439},
                {"probability_levels",14,1,3,0,1457,kw_140},
                {"response_levels",14,2,2,0,1441,kw_143},
                {"sqp",8,0,1,0,1437}
                }
```

GuiKeyWord kw_145[3] [static]

Initial value:

```
= {
                {"model_pointer",11,0,2,0,1667},
                {"nip",8,0,1,0,1475},
                {"sqp",8,0,1,0,1473}
                }
```

GuiKeyWord kw_146[5] [static]

Initial value:

```
= {
                {"adapt_import",8,0,1,1,1509},
                {"import",8,0,1,1,1507},
                {"mm_adapt_import",8,0,1,1,1511},
                {"refinement_samples",9,0,2,0,1513},
                {"seed",0x19,0,3,0,1515,0,0.,0.,0.,0,"{Random seed}
        MethodCommands.html#MethodNonDBayesCalib"}
                }
```

GuiKeyWord kw_147[4] [static]

Initial value:

```
= {
                {"first_order",8,0,1,1,1501},
                {"probability_refinement",8,5,2,0,1505,kw_146},
                {"sample_refinement",0,5,2,0,1504,kw_146},
                {"second_order",8,0,1,1,1503}
                }
```

GuiKeyWord kw_148[10] [static]

Initial value:

```
= {
                {"integration",8,4,3,0,1499,kw_147,0.,0.,0.,0,"{Integration method}
        MethodCommands.html#MethodNonDLocalRel"},
                {"nip",8,0,2,0,1497},
                {"no_approx",8,0,1,1,1493},
                {"sqp",8,0,2,0,1495},
                {"u_taylor_mean",8,0,1,1,1483},
                {"u_taylor_mpp",8,0,1,1,1487},
                {"u_two_point",8,0,1,1,1491},
                {"x_taylor_mean",8,0,1,1,1481},
                {"x_taylor_mpp",8,0,1,1,1485},
                {"x_two_point",8,0,1,1,1489}
                }
```

GuiKeyWord kw_149[1] [static]

Initial value:

```
= {
                {"num_reliability_levels",13,0,1,0,1537}
                }
```

GuiKeyWord kw_150[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,1533},
                {"series",8,0,1,1,1531}
                }
```

GuiKeyWord kw_151[4] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,1527},
                {"probabilities",8,0,1,1,1523},
                {"reliabilities",8,0,1,1,1525},
                {"system",8,2,2,0,1529,kw_150}
                }
```

GuiKeyWord kw_152[2] [static]

Initial value:

```
= {
                {"compute",8,4,2,0,1521,kw_151},
                {"num_response_levels",13,0,1,0,1519}
                }
```

GuiKeyWord kw_153[7] [static]

Initial value:

```
= {
                {"distribution",8,2,5,0,1589,kw_113},
                {"gen_reliability_levels",14,1,7,0,1599,kw_114},
                {"model_pointer",11,0,4,0,1667},
                {"mpp_search",8,10,1,0,1479,kw_148,0.,0.,0.,0,"{MPP search type}
        MethodCommands.html#MethodNonDLocalRel"},
                {"probability_levels",14,1,6,0,1595,kw_115},
                {"reliability_levels",14,1,3,0,1535,kw_149},
                {"response_levels",14,2,2,0,1517,kw_152}
                }
```

GuiKeyWord kw_154[16] [static]

Initial value:

```
= {
                {"display_all_evaluations",8,0,6,0,361,0,0.,0.,0.,0,"{Display NOMAD evaluations}
        MethodCommands.html#MethodNOMADDC"},
                {"display_format",11,0,4,0,357},
                {"function_precision",10,0,1,0,351,0,0.,0.,0.,0,"{Function Evaluation Precision}
        MethodCommands.html#MethodNOMADDC"},
                {"history_file",11,0,3,0,355,0,0.,0.,0.,0,"{NOMAD History File}
        MethodCommands.html#MethodNOMADDC"},
                {"linear_equality_constraint_matrix",14,0,13,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,15,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,16,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,14,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,8,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,9,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,11,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,12,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,10,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"model_pointer",11,0,7,0,1667},
                {"seed",0x19,0,2,0,353,0,0.,0.,0.,0,"{Random Seed} MethodCommands.html#MethodNOMADDC"},
                {"variable_neighborhood_search",10,0,5,0,359}
                }
```

GuiKeyWord kw_155[2] [static]

Initial value:

```
= {
                {"num_offspring",0x19,0,2,0,463,0,0.,0.,0.,0,"{Number of offspring in random shuffle
        crossover} MethodCommands.html#MethodJEGADC"},
                {"num_parents",0x19,0,1,0,461,0,0.,0.,0.,0,"{Number of parents in random shuffle crossover}
        MethodCommands.html#MethodJEGADC"}
                }
```

GuiKeyWord kw_156[5] [static]

Initial value:

```
= {
                {"crossover_rate",10,0,2,0,465,0,0.,0.,0.,0,"{Crossover rate}
        MethodCommands.html#MethodJEGADC"},
                {"multi_point_binary",9,0,1,1,453,0,0.,0.,0.,0,"{Multi point binary crossover}
        MethodCommands.html#MethodJEGADC"},
                {"multi_point_parameterized_binary",9,0,1,1,455,0,0.,0.,0.,0,"{Multi point parameterized
        binary crossover} MethodCommands.html#MethodJEGADC"},
                {"multi_point_real",9,0,1,1,457,0,0.,0.,0.,0,"{Multi point real crossover}
        MethodCommands.html#MethodJEGADC"},
                {"shuffle_random",8,2,1,1,459,kw_155,0.,0.,0.,0,"{Random shuffle crossover}
        MethodCommands.html#MethodJEGADC"}
                }
```

GuiKeyWord kw_157[3] [static]

Initial value:

```
= {
                {"flat_file",11,0,1,1,449},
                {"simple_random",8,0,1,1,445},
                {"unique_random",8,0,1,1,447}
                }
```

GuiKeyWord kw_158[1] [static]

Initial value:

```
= {
                {"mutation_scale",10,0,1,0,479,0,0.,0.,0.,0,"{Mutation scale}
        MethodCommands.html#MethodJEGADC"}
                }
```

GuiKeyWord kw_159[6] [static]

Initial value:

```
= {
                {"bit_random",8,0,1,1,469},
                {"mutation_rate",10,0,2,0,481,0,0.,0.,0.,0,"{Mutation rate}
        MethodCommands.html#MethodJEGADC"},
                {"offset_cauchy",8,1,1,1,475,kw_158},
                {"offset_normal",8,1,1,1,473,kw_158},
                {"offset_uniform",8,1,1,1,477,kw_158},
                {"replace_uniform",8,0,1,1,471}
                }
```

GuiKeyWord kw_160[3] [static]

Initial value:

```
= {
                {"metric_tracker",8,0,1,1,395,0,0.,0.,0.,0,"{Convergence type}
        MethodCommands.html#MethodJEGAMOGA"},
                {"num_generations",0x29,0,3,0,399,0,0.,0.,0.,0,"{Number generations for metric_tracker
        converger} MethodCommands.html#MethodJEGAMOGA"},
                {"percent_change",10,0,2,0,397,0,0.,0.,0.,0,"{Percent change limit for metric_tracker
        converger} MethodCommands.html#MethodJEGAMOGA"}
                }
```

GuiKeyWord kw_161[2] [static]

Initial value:

```
= {
                {"domination_count",8,0,1,1,369},
                {"layer_rank",8,0,1,1,367}
                }
```

GuiKeyWord kw_162[1] [static]

Initial value:

```
= {
                {"num_designs",0x29,0,1,0,391,0,2.,0.,0.,0,"{Number designs to keep for max_designs nicher}
        MethodCommands.html#MethodJEGAMOGA"}
                }
```

GuiKeyWord kw_163[3] [static]

Initial value:

```
= {
                {"distance",14,0,1,1,387},
                {"max_designs",14,1,1,1,389,kw_162},
                {"radial",14,0,1,1,385}
                }
```

GuiKeyWord kw_164[1] [static]

Initial value:

```
= {
                {"orthogonal_distance",14,0,1,1,403,0,0.,0.,0.,0,"{Post_processor distance}
        MethodCommands.html#MethodJEGAMOGA"}
                }
```

GuiKeyWord kw_165[2] [static]

Initial value:

```
= {
                {"shrinkage_fraction",10,0,1,0,381},
                {"shrinkage_percentage",2,0,1,0,380}
                }
```

GuiKeyWord kw_166[4] [static]

Initial value:

```
= {
                {"below_limit",10,2,1,1,379,kw_165,0.,0.,0.,0,"{Below limit selection}
        MethodCommands.html#MethodJEGADC"},
                {"elitist",8,0,1,1,373},
                {"roulette_wheel",8,0,1,1,375},
                {"unique_roulette_wheel",8,0,1,1,377}
                }
```

GuiKeyWord kw_167[22] [static]

Initial value:

```
= {
                {"convergence_type",8,3,4,0,393,kw_160},
                {"crossover_type",8,5,20,0,451,kw_156,0.,0.,0.,0,"{Crossover type}
        MethodCommands.html#MethodJEGADC"},
                {"fitness_type",8,2,1,0,365,kw_161,0.,0.,0.,0,"{Fitness type}
        MethodCommands.html#MethodJEGAMOGA"},
                {"initialization_type",8,3,19,0,443,kw_157,0.,0.,0.,0,"{Initialization type}
        MethodCommands.html#MethodJEGADC"},
                {"linear_equality_constraint_matrix",14,0,12,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,14,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,15,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,13,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,7,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,8,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,10,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,11,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,9,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"log_file",11,0,17,0,439,0,0.,0.,0.,0,"{Log file} MethodCommands.html#MethodJEGADC"},
                {"model_pointer",11,0,6,0,1667},
                {"mutation_type",8,6,21,0,467,kw_159,0.,0.,0.,0,"{Mutation type}
        MethodCommands.html#MethodJEGADC"},
                {"niching_type",8,3,3,0,383,kw_163,0.,0.,0.,0,"{Niche pressure type}
        MethodCommands.html#MethodJEGAMOGA"},
                {"population_size",0x29,0,16,0,437,0,0.,0.,0.,0,"{Number of population members}
        MethodCommands.html#MethodJEGADC"},
                {"postprocessor_type",8,1,5,0,401,kw_164,0.,0.,0.,0,"{Post_processor type}
        MethodCommands.html#MethodJEGAMOGA"},
                {"print_each_pop",8,0,18,0,441,0,0.,0.,0.,0,"{Population output}
        MethodCommands.html#MethodJEGADC"},
                {"replacement_type",8,4,2,0,371,kw_166,0.,0.,0.,0,"{Replacement type}
        MethodCommands.html#MethodJEGAMOGA"},
                {"seed",0x19,0,22,0,483,0,0.,0.,0.,0,"{Random seed} MethodCommands.html#MethodJEGADC"}
                }
```

GuiKeyWord kw_168[1] [static]

Initial value:

```
= {
                {"model_pointer",11,0,1,0,115,0,0.,0.,0.,0,"{Model pointer} MethodCommands.html#MethodMeta"
      }
                }
```

GuiKeyWord kw_169[1] [static]

Initial value:

```
= {
        {"seed",9,0,1,0,121,0,0.,0.,0.,0,"{Seed for random starting points}
MethodCommands.html#MethodMetaMultiStart"}
        }
```

GuiKeyWord kw_170[7] [static]

Initial value:

```
= {
        {"iterator_scheduling",8,2,5,0,141,kw_123,0.,0.,0.,0,"{Message passing configuration for
scheduling of iterator jobs} MethodCommands.html#MethodMeta"},
        {"iterator_servers",0x19,0,4,0,139,0,0.,0.,0.,0,"{Number of iterator servers}
MethodCommands.html#MethodMeta"},
        {"method_name",11,1,1,1,113,kw_168,0.,0.,0.,0,"{Identification of a sub-method by name (no
separate specification block)} MethodCommands.html#MethodMeta"},
        {"method_pointer",11,0,1,1,117,0,0.,0.,0.,0,"{Identification of a sub-method by pointer to
a separate specification block} MethodCommands.html#MethodMeta"},
        {"processors_per_iterator",0x19,0,6,0,147,0,0.,0.,0.,0,"{Number of processors per iterator
server} MethodCommands.html#MethodMeta"},
        {"random_starts",9,1,2,0,119,kw_169,0.,0.,0.,0,"{Number of random starting points}
MethodCommands.html#MethodMetaMultiStart"},
        {"starting_points",14,0,3,0,123,0,0.,0.,0.,0,"{List of user-specified starting points}
MethodCommands.html#MethodMetaMultiStart"}
        }
```

GuiKeyWord kw_171[2] [static]

Initial value:

```
= {
        {"model_pointer",11,0,2,0,1667},
        {"partitions",13,0,1,1,1655,0,0.,0.,0.,0,"{Partitions per variable}
MethodCommands.html#MethodPSMPS"}
        }
```

GuiKeyWord kw_172[5] [static]

Initial value:

```
= {
        {"min_boxsize_limit",10,0,2,0,671,0,0.,0.,0.,0,"{Min boxsize limit}
MethodCommands.html#MethodNCSUDC"},
        {"model_pointer",11,0,4,0,1667},
        {"solution_accuracy",2,0,1,0,668},
        {"solution_target",10,0,1,0,669,0,0.,0.,0.,0,"{Solution Target }
MethodCommands.html#MethodNCSUDC"},
        {"volume_boxsize_limit",10,0,3,0,673,0,0.,0.,0.,0,"{Volume boxsize limit}
MethodCommands.html#MethodNCSUDC"}
        }
```

GuiKeyWord kw_173[10] [static]

Initial value:

```
= {
                {"absolute_conv_tol",10,0,2,0,647,0,0.,0.,0.,0,"{Absolute function convergence tolerance}
        MethodCommands.html#MethodLSNL2SOL"},
                {"covariance",9,0,8,0,659,0,0.,0.,0.,0,"{Covariance post-processing}
        MethodCommands.html#MethodLSNL2SOL"},
                {"false_conv_tol",10,0,6,0,655,0,0.,0.,0.,0,"{False convergence tolerance}
        MethodCommands.html#MethodLSNL2SOL"},
                {"function_precision",10,0,1,0,645,0,0.,0.,0.,0,"{Relative precision in least squares
        terms} MethodCommands.html#MethodLSNL2SOL"},
                {"initial_trust_radius",10,0,7,0,657,0,0.,0.,0.,0,"{Initial trust region radius}
        MethodCommands.html#MethodLSNL2SOL"},
                {"model_pointer",11,0,10,0,1667},
                {"regression_diagnostics",8,0,9,0,661,0,0.,0.,0.,0,"{Regression diagnostics
        post-processing} MethodCommands.html#MethodLSNL2SOL"},
                {"singular_conv_tol",10,0,4,0,651,0,0.,0.,0.,0,"{Singular convergence tolerance}
        MethodCommands.html#MethodLSNL2SOL"},
                {"singular_radius",10,0,5,0,653,0,0.,0.,0.,0,"{Step limit for sctol}
        MethodCommands.html#MethodLSNL2SOL"},
                {"x_conv_tol",10,0,3,0,649,0,0.,0.,0.,0,"{Convergence tolerance for change in parameter
        vector} MethodCommands.html#MethodLSNL2SOL"}
                }
```

GuiKeyWord kw_174[1] [static]

Initial value:

```
= {
                {"surrogate_order",9,0,1,0,1115}
                }
```

GuiKeyWord kw_175[2] [static]

Initial value:

```
= {
                {"gaussian_process",8,0,1,1,1111},
                {"voronoi_surrogate",8,1,1,1,1113,kw_174}
                }
```

GuiKeyWord kw_176[2] [static]

Initial value:

```
= {
                {"global",8,0,1,1,1107},
                {"local",8,0,1,1,1105}
                }
```

GuiKeyWord kw_177[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,1133},
                {"series",8,0,1,1,1131}
                }
```

GuiKeyWord kw_178[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,1127},
                {"probabilities",8,0,1,1,1125},
                {"system",8,2,2,0,1129,kw_177}
                }
```

GuiKeyWord kw_179[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,1123,kw_178},
                {"num_response_levels",13,0,1,0,1121}
                }
```

GuiKeyWord kw_180[11] [static]

Initial value:

```
= {
                {"distribution",8,2,8,0,1189,kw_22,0.,0.,0.,0,"{Distribution type}
       MethodCommands.html#MethodNonD"},
                {"emulator",8,2,2,0,1109,kw_175},
                {"emulator_samples",9,0,3,0,1117},
                {"gen_reliability_levels",14,1,10,0,1199,kw_23,0.,0.,0.,0,"{Generalized reliability levels}
       MethodCommands.html#MethodNonD"},
                {"lipschitz",8,2,1,0,1103,kw_176},
                {"model_pointer",11,0,5,0,1667},
                {"probability_levels",14,1,9,0,1195,kw_24,0.,0.,0.,0,"{Probability levels}
       MethodCommands.html#MethodNonD"},
                {"response_levels",14,2,4,0,1119,kw_179},
                {"rng",8,2,11,0,1203,kw_25,0.,0.,0.,0,"{Random number generator}
       MethodCommands.html#MethodNonDMC"},
                {"samples",9,0,6,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
      },
                {"seed",0x19,0,7,0,1433,0,0.,0.,0.,0,"{Refinement seed}
       MethodCommands.html#MethodNonDLocalRel"}
                }
```

GuiKeyWord kw_181[1] [static]

Initial value:

```
= {
                {"num_reliability_levels",13,0,1,0,965,0,0.,0.,0.,0,"{Number of reliability levels}
       MethodCommands.html#MethodNonD"}
                }
```

GuiKeyWord kw_182[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,983},
                {"series",8,0,1,1,981}
                }
```

GuiKeyWord kw_183[4] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,977},
                {"probabilities",8,0,1,1,973},
                {"reliabilities",8,0,1,1,975},
                {"system",8,2,2,0,979,kw_182}
                }
```

GuiKeyWord kw_184[2] [static]

Initial value:

```
= {
                {"compute",8,4,2,0,971,kw_183,0.,0.,0.,0,"{Target statistics for response levels}
        MethodCommands.html#MethodNonD"},
                {"num_response_levels",13,0,1,0,969,0,0.,0.,0.,0,"{Number of response levels}
        MethodCommands.html#MethodNonD"}
                }
```

GuiKeyWord kw_185[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,819},
                {"annotated",8,0,1,0,815},
                {"freeform",8,0,1,0,817}
                }
```

GuiKeyWord kw_186[2] [static]

Initial value:

```
= {
                {"advancements",9,0,1,0,753},
                {"soft_convergence_limit",9,0,2,0,755}
                }
```

GuiKeyWord kw_187[3] [static]

Initial value:

```
= {
                {"adapted",8,2,1,1,751,kw_186},
                {"tensor_product",8,0,1,1,747},
                {"total_order",8,0,1,1,749}
                }
```

GuiKeyWord kw_188[1] [static]

Initial value:

```
= {
                {"noise_tolerance",14,0,1,0,777}
                }
```

GuiKeyWord kw_189[1] [static]

Initial value:

```
= {
                {"noise_tolerance",14,0,1,0,781}
                }
```

GuiKeyWord kw_190[2] [static]

Initial value:

```
= {
                {"l2_penalty",10,0,2,0,787,0,0.,0.,0.,0,"{l2_penalty used for elastic net modification of
        LASSO} MethodCommands.html#MethodNonDPCE"},
                {"noise_tolerance",14,0,1,0,785}
                }
```

GuiKeyWord kw_191[2] [static]

Initial value:

```
= {
                {"equality_constrained",8,0,1,0,767},
                {"svd",8,0,1,0,765}
                }
```

GuiKeyWord kw_192[1] [static]

Initial value:

```
= {
                {"noise_tolerance",14,0,1,0,771}
                }
```

GuiKeyWord kw_193[17] [static]

Initial value:

```
= {
                {"basis_pursuit",8,0,2,0,773,0,0.,0.,0.,0,"{L1 minimization via Basis Pursuit (BP)}
        MethodCommands.html#MethodNonDPCE"},
                {"basis_pursuit_denoising",8,1,2,0,775,kw_188,0.,0.,0.,0,"{L1 minimization via Basis
        Pursuit DeNoising (BPDN)} MethodCommands.html#MethodNonDPCE"},
                {"bp",0,0,2,0,772},
                {"bpdn",0,1,2,0,774,kw_188},
                {"cross_validation",8,0,3,0,789,0,0.,0.,0.,0,"{Specify whether to use cross validation}
        MethodCommands.html#MethodNonDPCE"},
                {"lars",0,1,2,0,778,kw_189},
                {"lasso",0,2,2,0,782,kw_190},
                {"least_absolute_shrinkage",8,2,2,0,783,kw_190,0.,0.,0.,0,"{L1 minimization via Least
        Absolute Shrinkage Operator (LASSO)} MethodCommands.html#MethodNonDPCE"},
                {"least_angle_regression",8,1,2,0,779,kw_189,0.,0.,0.,0,"{L1 minimization via Least Angle
        Regression (LARS)} MethodCommands.html#MethodNonDPCE"},
                {"least_squares",8,2,2,0,763,kw_191,0.,0.,0.,0,"{Least squares regression}
        MethodCommands.html#MethodNonDPCE"},
                {"omp",0,1,2,0,768,kw_192},
                {"orthogonal_matching_pursuit",8,1,2,0,769,kw_192,0.,0.,0.,0,"{L1 minimization via
        Orthogonal Matching Pursuit (OMP)} MethodCommands.html#MethodNonDPCE"},
                {"ratio_order",10,0,1,0,761,0,0.,0.,0.,0,"{Order of collocation oversampling relationship}
        MethodCommands.html#MethodNonDPCE"},
                {"reuse_points",8,0,6,0,795},
                {"reuse_samples",0,0,6,0,794},
                {"tensor_grid",8,0,5,0,793},
                {"use_derivatives",8,0,4,0,791}
                }
```

GuiKeyWord kw_194[3] [static]

Initial value:

```
= {
                {"incremental_lhs",8,0,2,0,801,0,0.,0.,0.,0,"{Use incremental LHS for expansion_samples}
        MethodCommands.html#MethodNonDPCE"},
                {"reuse_points",8,0,1,0,799},
                {"reuse_samples",0,0,1,0,798}
                }
```

GuiKeyWord kw_195[6] [static]

Initial value:

```
= {
                {"basis_type",8,3,2,0,745,kw_187},
                {"collocation_points",13,17,3,1,757,kw_193,0.,0.,0.,0,"{Number collocation points to
        estimate coeffs} MethodCommands.html#MethodNonDPCE"},
                {"collocation_ratio",10,17,3,1,759,kw_193,0.,0.,0.,0,"{Collocation point oversampling ratio
        to estimate coeffs} MethodCommands.html#MethodNonDPCE"},
                {"dimension_preference",14,0,1,0,743},
                {"expansion_samples",13,3,3,1,797,kw_194,0.,0.,0.,0,"{Number simulation samples to estimate
        coeffs} MethodCommands.html#MethodNonDPCE"},
                {"import_points_file",11,3,4,0,813,kw_185,0.,0.,0.,0,"{File name for points to be imported
        for forming a PCE (unstructured grid assumed)} MethodCommands.html#MethodNonDPCE"}
                }
```

GuiKeyWord kw_196[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,853},
                {"freeform",8,0,1,0,855}
                }
```

GuiKeyWord kw_197[6] [static]

Initial value:

```
= {
                {"collocation_points",13,0,1,1,805},
                {"cross_validation",8,0,2,0,807},
                {"import_points_file",11,3,5,0,813,kw_185,0.,0.,0.,0,"{File name for points to be imported
        for forming a PCE (unstructured grid assumed)} MethodCommands.html#MethodNonDPCE"},
                {"reuse_points",8,0,4,0,811},
                {"reuse_samples",0,0,4,0,810},
                {"tensor_grid",13,0,3,0,809}
                }
```

GuiKeyWord kw_198[3] [static]

Initial value:

```
= {
                {"decay",8,0,1,1,717},
                {"generalized",8,0,1,1,719},
                {"sobol",8,0,1,1,715}
                }
```

GuiKeyWord kw_199[2] [static]

Initial value:

```
= {
                {"dimension_adaptive",8,3,1,1,713,kw_198},
                {"uniform",8,0,1,1,711}
                }
```

GuiKeyWord kw_200[4] [static]

Initial value:

```
= {
                {"adapt_import",8,0,1,1,845},
                {"import",8,0,1,1,843},
                {"mm_adapt_import",8,0,1,1,847},
                {"refinement_samples",9,0,2,0,849,0,0.,0.,0.,0,"{Refinement samples}
        MethodCommands.html#MethodNonDLocalRel"}
                }
```

GuiKeyWord kw_201[3] [static]

Initial value:

```
= {
                {"dimension_preference",14,0,1,0,733,0,0.,0.,0.,0,"{Dimension preference for anisotropic
        tensor and sparse grids} MethodCommands.html#MethodNonDPCE"},
                {"nested",8,0,2,0,735},
                {"non_nested",8,0,2,0,737}
                }
```

GuiKeyWord kw_202[2] [static]

Initial value:

```
= {
                {"lhs",8,0,1,1,837},
                {"random",8,0,1,1,839}
                }
```

GuiKeyWord kw_203[5] [static]

Initial value:

```
= {
                {"dimension_preference",14,0,2,0,733,0,0.,0.,0.,0,"{Dimension preference for anisotropic
        tensor and sparse grids} MethodCommands.html#MethodNonDPCE"},
                {"nested",8,0,3,0,735},
                {"non_nested",8,0,3,0,737},
                {"restricted",8,0,1,0,729},
                {"unrestricted",8,0,1,0,731}
                }
```

GuiKeyWord kw_204[2] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,2,0,827,0,0.,0.,0.,0,"{VBD tolerance for omitting small indices}
        MethodCommands.html#MethodNonDMC"},
                {"interaction_order",0x19,0,1,0,825,0,0.,0.,0.,0,"{Restriction of order of VBD interations}
        MethodCommands.html#MethodNonDPCE"}
                }
```

GuiKeyWord kw_206[1] [static]

Initial value:

```
= {
                {"previous_samples",9,0,1,1,953,0,0.,0.,0.,0,"{Previous samples for incremental approaches}
        MethodCommands.html#MethodNonDMC"}
                }
```

GuiKeyWord kw_207[4] [static]

Initial value:

```
= {
                {"incremental_lhs",8,1,1,1,949,kw_206},
                {"incremental_random",8,1,1,1,951,kw_206},
                {"lhs",8,0,1,1,947},
                {"random",8,0,1,1,945}
                }
```

GuiKeyWord kw_208[1] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,1,0,957}
                }
```

GuiKeyWord kw_209[13] [static]

Initial value:

```
= {
                {"backfill",8,0,3,0,959},
                {"distribution",8,2,7,0,1189,kw_22,0.,0.,0.,0,"{Distribution type}
        MethodCommands.html#MethodNonD"},
                {"fixed_seed",8,0,13,0,961,0,0.,0.,0.,0,"{Fixed seed flag} MethodCommands.html#MethodNonDMC
       "},
                {"gen_reliability_levels",14,1,9,0,1199,kw_23,0.,0.,0.,0,"{Generalized reliability levels}
        MethodCommands.html#MethodNonD"},
                {"model_pointer",11,0,4,0,1667},
                {"probability_levels",14,1,8,0,1195,kw_24,0.,0.,0.,0,"{Probability levels}
        MethodCommands.html#MethodNonD"},
                {"reliability_levels",14,1,11,0,963,kw_181,0.,0.,0.,0,"{Reliability levels}
        MethodCommands.html#MethodNonD"},
                {"response_levels",14,2,12,0,967,kw_184,0.,0.,0.,0,"{Response levels}
        MethodCommands.html#MethodNonD"},
                {"rng",8,2,10,0,1203,kw_25,0.,0.,0.,0,"{Random number generator}
        MethodCommands.html#MethodNonDMC"},
                {"sample_type",8,4,1,0,943,kw_207},
                {"samples",9,0,5,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
```

```
},
          {"seed",0x19,0,6,0,1433,0,0.,0.,0.,0,"{Refinement seed}
 MethodCommands.html#MethodNonDLocalRel"},
          {"variance_based_decomp",8,1,2,0,955,kw_208}
          }
```

GuiKeyWord kw_210[2] [static]

Initial value:

```
= {
          {"annotated",8,0,1,0,937},
          {"freeform",8,0,1,0,939}
          }
```

GuiKeyWord kw_211[2] [static]

Initial value:

```
= {
          {"generalized",8,0,1,1,879},
          {"sobol",8,0,1,1,877}
          }
```

GuiKeyWord kw_212[3] [static]

Initial value:

```
= {
          {"dimension_adaptive",8,2,1,1,875,kw_211},
          {"local_adaptive",8,0,1,1,881},
          {"uniform",8,0,1,1,873}
          }
```

GuiKeyWord kw_213[2] [static]

Initial value:

```
= {
          {"generalized",8,0,1,1,869},
          {"sobol",8,0,1,1,867}
          }
```

GuiKeyWord kw_214[2] [static]

Initial value:

```
= {
          {"dimension_adaptive",8,2,1,1,865,kw_213},
          {"uniform",8,0,1,1,863}
          }
```

GuiKeyWord kw_215[4] [static]

Initial value:

```
= {
                {"adapt_import",8,0,1,1,929},
                {"import",8,0,1,1,927},
                {"mm_adapt_import",8,0,1,1,931},
                {"refinement_samples",9,0,2,0,933}
                }
```

GuiKeyWord kw_216[2] [static]

Initial value:

```
= {
                {"lhs",8,0,1,1,921},
                {"random",8,0,1,1,923}
                }
```

GuiKeyWord kw_217[4] [static]

Initial value:

```
= {
                {"hierarchical",8,0,2,0,899},
                {"nodal",8,0,2,0,897},
                {"restricted",8,0,1,0,893},
                {"unrestricted",8,0,1,0,895}
                }
```

GuiKeyWord kw_218[2] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,2,0,913,0,0.,0.,0.,0,"{VBD tolerance for omitting small indices}
        MethodCommands.html#MethodNonDSC"},
                {"interaction_order",0x19,0,1,0,911,0,0.,0.,0.,0,"{Restriction of order of VBD interations}
        MethodCommands.html#MethodNonDSC"}
                }
```

GuiKeyWord kw_219[28] [static]

Initial value:

```
= {
                {"askey",8,0,2,0,885},
                {"diagonal_covariance",8,0,8,0,915},
                {"dimension_preference",14,0,4,0,901,0,0.,0.,0.,0,"{Dimension preference for anisotropic
        tensor and sparse grids} MethodCommands.html#MethodNonDSC"},
                {"distribution",8,2,15,0,1189,kw_22,0.,0.,0.,0,"{Distribution type}
        MethodCommands.html#MethodNonD"},
                {"export_points_file",11,2,11,0,935,kw_210,0.,0.,0.,0,"{File name for exporting
        approximation-based samples from evaluating the interpolant} MethodCommands.html#MethodNonDSC"},
                {"fixed_seed",8,0,21,0,961,0,0.,0.,0.,0,"{Fixed seed flag} MethodCommands.html#MethodNonDMC
       "},
                {"full_covariance",8,0,8,0,917},
                {"gen_reliability_levels",14,1,17,0,1199,kw_23,0.,0.,0.,0,"{Generalized reliability levels}
        MethodCommands.html#MethodNonD"},
                {"h_refinement",8,3,1,0,871,kw_212},
                {"model_pointer",11,0,12,0,1667},
```

```
                {"nested",8,0,6,0,905},
                {"non_nested",8,0,6,0,907},
                {"p_refinement",8,2,1,0,861,kw_214},
                {"piecewise",8,0,2,0,883},
                {"probability_levels",14,1,16,0,1195,kw_24,0.,0.,0.,0,"{Probability levels}
        MethodCommands.html#MethodNonD"},
                {"probability_refinement",8,4,10,0,925,kw_215},
                {"quadrature_order",13,0,3,1,889,0,0.,0.,0.,0,"{Quadrature order for collocation points}
        MethodCommands.html#MethodNonDSC"},
                {"reliability_levels",14,1,19,0,963,kw_181,0.,0.,0.,0,"{Reliability levels}
        MethodCommands.html#MethodNonD"},
                {"response_levels",14,2,20,0,967,kw_184,0.,0.,0.,0,"{Response levels}
        MethodCommands.html#MethodNonD"},
                {"rng",8,2,18,0,1203,kw_25,0.,0.,0.,0,"{Random number generator}
        MethodCommands.html#MethodNonDMC"},
                {"sample_refinement",0,4,10,0,924,kw_215},
                {"sample_type",8,2,9,0,919,kw_216},
                {"samples",9,0,13,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC
       "},
                {"seed",0x19,0,14,0,1433,0,0.,0.,0.,0,"{Refinement seed}
        MethodCommands.html#MethodNonDLocalRel"},
                {"sparse_grid_level",13,4,3,1,891,kw_217,0.,0.,0.,0,"{Sparse grid level for collocation
        points} MethodCommands.html#MethodNonDSC"},
                {"use_derivatives",8,0,5,0,903,0,0.,0.,0.,0,"{Derivative enhancement flag}
        MethodCommands.html#MethodNonDSC"},
                {"variance_based_decomp",8,2,7,0,909,kw_218,0.,0.,0.,0,"{Variance-based decomposition
        (VBD)} MethodCommands.html#MethodNonDSC"},
                {"wiener",8,0,2,0,887}
                }
```

GuiKeyWord kw_220[2] `[static]`

Initial value:

```
= {
                {"misc_options",15,0,1,0,665},
                {"model_pointer",11,0,2,0,1667}
                }
```

GuiKeyWord kw_221[13] `[static]`

Initial value:

```
= {
                {"function_precision",10,0,12,0,267,0,0.,0.,0.,0,"{Function precision}
        MethodCommands.html#MethodNPSOLDC"},
                {"linear_equality_constraint_matrix",14,0,7,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,9,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,10,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,8,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,2,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,3,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,5,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,6,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,4,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"linesearch_tolerance",10,0,13,0,269,0,0.,0.,0.,0,"{Line search tolerance}
        MethodCommands.html#MethodNPSOLDC"},
                {"model_pointer",11,0,1,0,1667},
                {"verify_level",9,0,11,0,265,0,0.,0.,0.,0,"{Gradient verification level}
        MethodCommands.html#MethodNPSOLDC"}
                }
```

GuiKeyWord kw_222[12] [static]

Initial value:

```
= {
                {"gradient_tolerance",10,0,12,0,307},
                {"linear_equality_constraint_matrix",14,0,7,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,9,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,10,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,8,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,2,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,3,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,5,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,6,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,4,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"max_step",10,0,11,0,305},
                {"model_pointer",11,0,1,0,1667}
                }
```

GuiKeyWord kw_223[11] [static]

Initial value:

```
= {
                {"linear_equality_constraint_matrix",14,0,8,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,10,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,11,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,9,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,3,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,4,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,6,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,7,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,5,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"model_pointer",11,0,2,0,1667},
                {"search_scheme_size",9,0,1,0,311}
                }
```

GuiKeyWord kw_224[3] [static]

Initial value:

```
= {
                {"argaez_tapia",8,0,1,1,297},
                {"el_bakry",8,0,1,1,295},
                {"van_shanno",8,0,1,1,299}
                }
```

GuiKeyWord kw_225[4] [static]

Initial value:

```
= {
                {"gradient_based_line_search",8,0,1,1,287,0,0.,0.,0.,0,"[CHOOSE line search type]"},
                {"tr_pds",8,0,1,1,291},
                {"trust_region",8,0,1,1,289},
                {"value_based_line_search",8,0,1,1,285}
                }
```

GuiKeyWord kw_226[16] [static]

Initial value:

```
= {
                {"centering_parameter",10,0,4,0,303},
                {"gradient_tolerance",10,0,16,0,307},
                {"linear_equality_constraint_matrix",14,0,11,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,13,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,14,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,12,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,6,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,7,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,9,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,10,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,8,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"max_step",10,0,15,0,305},
                {"merit_function",8,3,2,0,293,kw_224},
                {"model_pointer",11,0,5,0,1667},
                {"search_method",8,4,1,0,283,kw_225},
                {"steplength_to_boundary",10,0,3,0,301}
                }
```

GuiKeyWord kw_227[5] [static]

Initial value:

```
= {
                {"debug",8,0,1,1,53,0,0.,0.,0.,0,"[CHOOSE output level]"},
                {"normal",8,0,1,1,57},
                {"quiet",8,0,1,1,59},
                {"silent",8,0,1,1,61},
                {"verbose",8,0,1,1,55}
                }
```

GuiKeyWord kw_228[2] [static]

Initial value:

```
= {
                {"model_pointer",11,0,1,0,129,0,0.,0.,0.,0,"{Identification of model by pointer}
        MethodCommands.html#MethodMetaMultiStart"},
                {"opt_model_pointer",3,0,1,0,128}
                }
```

GuiKeyWord kw_229[1] [static]

Initial value:

```
= {
                {"seed",9,0,1,0,135,0,0.,0.,0.,0,"{Seed for random weighting sets}
        MethodCommands.html#MethodMetaParetoSet"}
                }
```

GuiKeyWord kw_230[10] [static]

Initial value:

```
= {
                {"iterator_scheduling",8,2,5,0,141,kw_123,0.,0.,0.,0,"{Message passing configuration for
        scheduling of iterator jobs} MethodCommands.html#MethodMeta"},
                {"iterator_servers",0x19,0,4,0,139,0,0.,0.,0.,0,"{Number of iterator servers}
        MethodCommands.html#MethodMeta"},
                {"method_name",11,2,1,1,127,kw_228,0.,0.,0.,0,"{Identification of sub-iterator by name}
        MethodCommands.html#MethodMetaMultiStart"},
                {"method_pointer",11,0,1,1,131,0,0.,0.,0.,0,"{Identification of sub-iterator by pointer}
        MethodCommands.html#MethodMetaMultiStart"},
                {"multi_objective_weight_sets",6,0,3,0,136},
                {"opt_method_name",3,2,1,1,126,kw_228},
                {"opt_method_pointer",3,0,1,1,130},
                {"processors_per_iterator",0x19,0,6,0,147,0,0.,0.,0.,0,"{Number of processors per iterator
        server} MethodCommands.html#MethodMeta"},
                {"random_weight_sets",9,1,2,0,133,kw_229,0.,0.,0.,0,"{Number of random weighting sets}
        MethodCommands.html#MethodMetaParetoSet"},
                {"weight_sets",14,0,3,0,137,0,0.,0.,0.,0,"{List of user-specified weighting sets}
        MethodCommands.html#MethodMetaParetoSet"}
                }
```

GuiKeyWord kw_231[4] [static]

Initial value:

```
= {
                {"model_pointer",11,0,2,0,1667},
                {"partitions",13,0,1,0,1429,0,0.,0.,0.,0,"{Number of partitions}
        MethodCommands.html#MethodPSUADE"},
                {"samples",9,0,3,0,1431,0,0.,0.,0.,0,"{Number of samples} MethodCommands.html#MethodNonDMC"
       },
                {"seed",0x19,0,4,0,1433,0,0.,0.,0.,0,"{Refinement seed}
        MethodCommands.html#MethodNonDLocalRel"}
                }
```

GuiKeyWord kw_232[5] [static]

Initial value:

```
= {
                {"converge_order",8,0,1,1,1661},
                {"converge_qoi",8,0,1,1,1663},
                {"estimate_order",8,0,1,1,1659},
                {"model_pointer",11,0,3,0,1667},
                {"refinement_rate",10,0,2,0,1665,0,0.,0.,0.,0,"{Refinement rate}
        MethodCommands.html#MethodSolnRichardson"}
                }
```

GuiKeyWord kw_233[2] [static]

Initial value:

```
= {
                {"num_generations",0x29,0,2,0,435},
                {"percent_change",10,0,1,0,433}
                }
```

GuiKeyWord kw_234[2] [static]

Initial value:

```
= {
                {"num_generations",0x29,0,2,0,429,0,0.,0.,0.,0,"{Number of generations (for convergence
        test) } MethodCommands.html#MethodJEGASOGA"},
                {"percent_change",10,0,1,0,427,0,0.,0.,0.,0,"{Percent change in fitness}
        MethodCommands.html#MethodJEGASOGA"}
                }
```

GuiKeyWord kw_235[2] [static]

Initial value:

```
= {
                {"average_fitness_tracker",8,2,1,1,431,kw_233},
                {"best_fitness_tracker",8,2,1,1,425,kw_234}
                }
```

GuiKeyWord kw_236[2] [static]

Initial value:

```
= {
                {"constraint_penalty",10,0,2,0,411,0,0.,0.,0.,0,"{Constraint penalty in merit function}
        MethodCommands.html#MethodJEGASOGA"},
                {"merit_function",8,0,1,1,409}
                }
```

GuiKeyWord kw_237[4] [static]

Initial value:

```
= {
                {"elitist",8,0,1,1,415},
                {"favor_feasible",8,0,1,1,417},
                {"roulette_wheel",8,0,1,1,419},
                {"unique_roulette_wheel",8,0,1,1,421}
                }
```

GuiKeyWord kw_238[20] [static]

Initial value:

```
= {
                {"convergence_type",8,2,3,0,423,kw_235,0.,0.,0.,0,"{Convergence type}
        MethodCommands.html#MethodJEGASOGA"},
                {"crossover_type",8,5,18,0,451,kw_156,0.,0.,0.,0,"{Crossover type}
        MethodCommands.html#MethodJEGADC"},
                {"fitness_type",8,2,1,0,407,kw_236,0.,0.,0.,0,"{Fitness type}
        MethodCommands.html#MethodJEGASOGA"},
                {"initialization_type",8,3,17,0,443,kw_157,0.,0.,0.,0,"{Initialization type}
        MethodCommands.html#MethodJEGADC"},
                {"linear_equality_constraint_matrix",14,0,10,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,12,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,13,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,11,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,5,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,6,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,8,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,9,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,7,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"log_file",11,0,15,0,439,0,0.,0.,0.,0,"{Log file} MethodCommands.html#MethodJEGADC"},
                {"model_pointer",11,0,4,0,1667},
                {"mutation_type",8,6,19,0,467,kw_159,0.,0.,0.,0,"{Mutation type}
        MethodCommands.html#MethodJEGADC"},
                {"population_size",0x29,0,14,0,437,0,0.,0.,0.,0,"{Number of population members}
        MethodCommands.html#MethodJEGADC"},
                {"print_each_pop",8,0,16,0,441,0,0.,0.,0.,0,"{Population output}
        MethodCommands.html#MethodJEGADC"},
                {"replacement_type",8,4,2,0,413,kw_237,0.,0.,0.,0,"{Replacement type}
        MethodCommands.html#MethodJEGASOGA"},
                {"seed",0x19,0,20,0,483,0,0.,0.,0.,0,"{Random seed} MethodCommands.html#MethodJEGADC"}
                }
```

GuiKeyWord kw_239[15] [static]

Initial value:

```
= {
                {"function_precision",10,0,13,0,267,0,0.,0.,0.,0,"{Function precision}
        MethodCommands.html#MethodNPSOLDC"},
                {"linear_equality_constraint_matrix",14,0,8,0,495,0,0.,0.,0.,0,"{Linear equality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_equality_scale_types",15,0,10,0,499,0,0.,0.,0.,0,"{Linear equality scaling types}
        MethodCommands.html#MethodMin"},
                {"linear_equality_scales",14,0,11,0,501,0,0.,0.,0.,0,"{Linear equality scales}
        MethodCommands.html#MethodMin"},
                {"linear_equality_targets",14,0,9,0,497,0,0.,0.,0.,0,"{Linear equality targets}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_constraint_matrix",14,0,3,0,485,0,0.,0.,0.,0,"{Linear inequality
        coefficient matrix} MethodCommands.html#MethodMin"},
                {"linear_inequality_lower_bounds",14,0,4,0,487,0,0.,0.,0.,0,"{Linear inequality lower
        bounds} MethodCommands.html#MethodMin"},
                {"linear_inequality_scale_types",15,0,6,0,491,0,0.,0.,0.,0,"{Linear inequality scaling
        types} MethodCommands.html#MethodMin"},
                {"linear_inequality_scales",14,0,7,0,493,0,0.,0.,0.,0,"{Linear inequality scales}
        MethodCommands.html#MethodMin"},
                {"linear_inequality_upper_bounds",14,0,5,0,489,0,0.,0.,0.,0,"{Linear inequality upper
        bounds} MethodCommands.html#MethodMin"},
                {"linesearch_tolerance",10,0,14,0,269,0,0.,0.,0.,0,"{Line search tolerance}
        MethodCommands.html#MethodNPSOLDC"},
```

```
                {"model_pointer",11,0,2,0,1667},
                {"nlssol",8,0,1,1,263},
                {"npsol",8,0,1,1,261},
                {"verify_level",9,0,12,0,265,0,0.,0.,0.,0,"{Gradient verification level}
      MethodCommands.html#MethodNPSOLDC"}
                }
```

GuiKeyWord kw_240[7] [static]

Initial value:

```
= {
                {"approx_method_name",3,0,1,1,214},
                {"approx_method_pointer",3,0,1,1,212},
                {"approx_model_pointer",3,0,2,2,216},
                {"method_name",11,0,1,1,215},
                {"method_pointer",11,0,1,1,213},
                {"model_pointer",11,0,2,2,217},
                {"replace_points",8,0,3,0,219,0,0.,0.,0.,0,"{Replace points used in surrogate construction
      with best points from previous iteration} MethodCommands.html#MethodSBG"}
                }
```

GuiKeyWord kw_241[2] [static]

Initial value:

```
= {
                {"filter",8,0,1,1,205,0,0.,0.,0.,0,"@[CHOOSE acceptance logic]"},
                {"tr_ratio",8,0,1,1,203}
                }
```

GuiKeyWord kw_242[7] [static]

Initial value:

```
= {
                {"augmented_lagrangian_objective",8,0,1,1,181,0,0.,0.,0.,0,"[CHOOSE objective formulation]"
      },
                {"lagrangian_objective",8,0,1,1,183},
                {"linearized_constraints",8,0,2,2,187,0,0.,0.,0.,0,"[CHOOSE constraint formulation]"},
                {"no_constraints",8,0,2,2,189},
                {"original_constraints",8,0,2,2,185,0,0.,0.,0.,0,"@"},
                {"original_primary",8,0,1,1,177,0,0.,0.,0.,0,"@"},
                {"single_objective",8,0,1,1,179}
                }
```

GuiKeyWord kw_243[1] [static]

Initial value:

```
= {
                {"homotopy",8,0,1,1,209}
                }
```

GuiKeyWord kw_244[4] [static]

Initial value:

```
= {
                {"adaptive_penalty_merit",8,0,1,1,195,0,0.,0.,0.,0,"[CHOOSE merit function]"},
                {"augmented_lagrangian_merit",8,0,1,1,199,0,0.,0.,0.,0,"@"},
                {"lagrangian_merit",8,0,1,1,197},
                {"penalty_merit",8,0,1,1,193}
                }
```

GuiKeyWord kw_245[6] [static]

Initial value:

```
= {
                {"contract_threshold",10,0,3,0,167,0,0.,0.,0.,0,"{Shrink trust region if trust region ratio
        is below this value} MethodCommands.html#MethodSBL"},
                {"contraction_factor",10,0,5,0,171,0,0.,0.,0.,0,"{Trust region contraction factor}
        MethodCommands.html#MethodSBL"},
                {"expand_threshold",10,0,4,0,169,0,0.,0.,0.,0,"{Expand trust region if trust region ratio
        is above this value} MethodCommands.html#MethodSBL"},
                {"expansion_factor",10,0,6,0,173,0,0.,0.,0.,0,"{Trust region expansion factor}
        MethodCommands.html#MethodSBL"},
                {"initial_size",10,0,1,0,163,0,0.,0.,0.,0,"{Trust region initial size (relative to bounds)}
        MethodCommands.html#MethodSBL"},
                {"minimum_size",10,0,2,0,165,0,0.,0.,0.,0,"{Trust region minimum size}
        MethodCommands.html#MethodSBL"}
                }
```

GuiKeyWord kw_246[13] [static]

Initial value:

```
= {
                {"acceptance_logic",8,2,8,0,201,kw_241,0.,0.,0.,0,"{SBL iterate acceptance logic}
        MethodCommands.html#MethodSBL"},
                {"approx_method_name",3,0,1,1,152},
                {"approx_method_pointer",3,0,1,1,150},
                {"approx_model_pointer",3,0,2,2,154},
                {"approx_subproblem",8,7,6,0,175,kw_242,0.,0.,0.,0,"{Approximate subproblem formulation}
        MethodCommands.html#MethodSBL"},
                {"constraint_relax",8,1,9,0,207,kw_243,0.,0.,0.,0,"{SBL constraint relaxation method for
        infeasible iterates} MethodCommands.html#MethodSBL"},
                {"merit_function",8,4,7,0,191,kw_244,0.,0.,0.,0,"{SBL merit function}
        MethodCommands.html#MethodSBL"},
                {"method_name",11,0,1,1,153,0,0.,0.,0.,0,"{Identification of minimizer by name}
        MethodCommands.html#MethodMetaParetoSet"},
                {"method_pointer",11,0,1,1,151,0,0.,0.,0.,0,"{Identification of minimizer by pointer}
        MethodCommands.html#MethodMetaParetoSet"},
                {"model_pointer",11,0,2,2,155,0,0.,0.,0.,0,"{Identification of model by pointer}
        MethodCommands.html#MethodMetaParetoSet"},
                {"soft_convergence_limit",9,0,3,0,157,0,0.,0.,0.,0,"{Soft convergence limit for SBL
        iterations} MethodCommands.html#MethodSBL"},
                {"trust_region",8,6,5,0,161,kw_245,0.,0.,0.,0,"{Trust region group specification}
        MethodCommands.html#MethodSBL"},
                {"truth_surrogate_bypass",8,0,4,0,159,0,0.,0.,0.,0,"{Flag for bypassing lower level
        surrogates in truth verifications} MethodCommands.html#MethodSBL"}
                }
```

GuiKeyWord kw_247[4] [static]

Initial value:

```
= {
                {"final_point",14,0,1,1,1629,0,0.,0.,0.,0,"[CHOOSE final pt or increment]{Termination point
        of vector} MethodCommands.html#MethodPSVPS"},
                {"model_pointer",11,0,3,0,1667},
                {"num_steps",9,0,2,2,1633,0,0.,0.,0.,0,"{Number of steps along vector}
        MethodCommands.html#MethodPSVPS"},
                {"step_vector",14,0,1,1,1631,0,0.,0.,0.,0,"{Step vector} MethodCommands.html#MethodPSVPS"}
                }
```

GuiKeyWord kw_249[1] [static]

Initial value:

```
= {
                {"optional_interface_responses_pointer",11,0,1,0,1877,0,0.,0.,0.,0,"{Responses pointer for
        nested model optional interfaces} ModelCommands.html#ModelNested"}
                }
```

GuiKeyWord kw_250[2] [static]

Initial value:

```
= {
                {"master",8,0,1,1,1885},
                {"peer",8,0,1,1,1887}
                }
```

GuiKeyWord kw_251[7] [static]

Initial value:

```
= {
                {"iterator_scheduling",8,2,2,0,1883,kw_250},
                {"iterator_servers",0x19,0,1,0,1881},
                {"primary_response_mapping",14,0,6,0,1895,0,0.,0.,0.,0,"{Primary response mappings for
        nested models} ModelCommands.html#ModelNested"},
                {"primary_variable_mapping",15,0,4,0,1891,0,0.,0.,0.,0,"{Primary variable mappings for
        nested models} ModelCommands.html#ModelNested"},
                {"processors_per_iterator",0x19,0,3,0,1889},
                {"secondary_response_mapping",14,0,7,0,1897,0,0.,0.,0.,0,"{Secondary response mappings for
        nested models} ModelCommands.html#ModelNested"},
                {"secondary_variable_mapping",15,0,5,0,1893,0,0.,0.,0.,0,"{Secondary variable mappings for
        nested models} ModelCommands.html#ModelNested"}
                }
```

GuiKeyWord kw_252[2] [static]

Initial value:

```
= {
                {"optional_interface_pointer",11,1,1,0,1875,kw_249,0.,0.,0.,0,"{Optional interface set
        pointer} ModelCommands.html#ModelNested"},
                {"sub_method_pointer",11,7,2,1,1879,kw_251,0.,0.,0.,0,"{Sub-method pointer for nested
        models} ModelCommands.html#ModelNested"}
                }
```

GuiKeyWord kw_253[1] [static]

Initial value:

```
= {
                {"interface_pointer",11,0,1,0,1681,0,0.,0.,0.,0,"{Interface set pointer}
        ModelCommands.html#ModelSingle"}
                }
```

GuiKeyWord kw_254[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,1841},
                {"annotated",8,0,1,0,1837},
                {"freeform",8,0,1,0,1839}
                }
```

GuiKeyWord kw_255[6] [static]

Initial value:

```
= {
                {"additive",8,0,2,2,1819,0,0.,0.,0.,0,"[CHOOSE correction type]"},
                {"combined",8,0,2,2,1823},
                {"first_order",8,0,1,1,1815,0,0.,0.,0.,0,"[CHOOSE correction order]"},
                {"multiplicative",8,0,2,2,1821},
                {"second_order",8,0,1,1,1817},
                {"zeroth_order",8,0,1,1,1813}
                }
```

GuiKeyWord kw_256[2] [static]

Initial value:

```
= {
                {"folds",9,0,1,0,1829,0,0.,0.,0.,0,"{Number cross validation folds}
        ModelCommands.html#ModelSurrG"},
                {"percent",10,0,1,0,1831,0,0.,0.,0.,0,"{Percent points per CV fold}
        ModelCommands.html#ModelSurrG"}
                }
```

GuiKeyWord kw_257[2] [static]

Initial value:

```
= {
                {"cross_validation",8,2,1,0,1827,kw_256,0.,0.,0.,0,"{Perform cross validation}
        ModelCommands.html#ModelSurrG"},
                {"press",8,0,2,0,1833,0,0.,0.,0.,0,"{Perform PRESS cross validation}
        ModelCommands.html#ModelSurrG"}
                }
```

GuiKeyWord kw_258[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,1805},
                {"freeform",8,0,1,0,1807}
                }
```

GuiKeyWord kw_259[3] [static]

Initial value:

```
= {
                {"constant",8,0,1,1,1697},
                {"linear",8,0,1,1,1699},
                {"reduced_quadratic",8,0,1,1,1701}
                }
```

GuiKeyWord kw_260[2] [static]

Initial value:

```
= {
                {"point_selection",8,0,1,0,1693,0,0.,0.,0.,0,"{GP point selection}
       ModelCommands.html#ModelSurrG"},
                {"trend",8,3,2,0,1695,kw_259,0.,0.,0.,0,"{GP trend function} ModelCommands.html#ModelSurrG"
      }
                }
```

GuiKeyWord kw_261[4] [static]

Initial value:

```
= {
                {"constant",8,0,1,1,1707},
                {"linear",8,0,1,1,1709},
                {"quadratic",8,0,1,1,1713},
                {"reduced_quadratic",8,0,1,1,1711}
                }
```

GuiKeyWord kw_262[7] [static]

Initial value:

```
= {
                {"correlation_lengths",14,0,5,0,1723,0,0.,0.,0.,0,"{Surfpack GP correlation lengths}
       ModelCommands.html#ModelSurrG"},
                {"export_model_file",11,0,6,0,1725},
                {"find_nugget",9,0,4,0,1721,0,0.,0.,0.,0,"{Surfpack finds the optimal nugget }
       ModelCommands.html#ModelSurrG"},
                {"max_trials",0x19,0,3,0,1717,0,0.,0.,0.,0,"{Surfpack GP maximum trials}
       ModelCommands.html#ModelSurrG"},
                {"nugget",0x1a,0,4,0,1719,0,0.,0.,0.,0,"{Surfpack user-specified nugget }
       ModelCommands.html#ModelSurrG"},
                {"optimization_method",11,0,2,0,1715,0,0.,0.,0.,0,"{Surfpack GP optimization method}
       ModelCommands.html#ModelSurrG"},
                {"trend",8,4,1,0,1705,kw_261,0.,0.,0.,0,"{Surfpack GP trend function}
       ModelCommands.html#ModelSurrG"}
                }
```

GuiKeyWord kw_263[2] [static]

Initial value:

```
= {
                {"dakota",8,2,1,1,1691,kw_260},
                {"surfpack",8,7,1,1,1703,kw_262}
                }
```

GuiKeyWord kw_264[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,1801},
                {"annotated",8,0,1,0,1797,0,0.,0.,0.,0,"{Challenge file in annotated format}
        ModelCommands.html#ModelSurrG"},
                {"freeform",8,0,1,0,1799,0,0.,0.,0.,0,"{Challenge file in freeform format}
        ModelCommands.html#ModelSurrG"}
                }
```

GuiKeyWord kw_265[2] [static]

Initial value:

```
= {
                {"cubic",8,0,1,1,1735},
                {"linear",8,0,1,1,1733}
                }
```

GuiKeyWord kw_266[3] [static]

Initial value:

```
= {
                {"export_model_file",11,0,3,0,1737},
                {"interpolation",8,2,2,0,1731,kw_265,0.,0.,0.,0,"{MARS interpolation}
        ModelCommands.html#ModelSurrG"},
                {"max_bases",9,0,1,0,1729,0,0.,0.,0.,0,"{MARS maximum bases} ModelCommands.html#ModelSurrG"
       }
                }
```

GuiKeyWord kw_267[3] [static]

Initial value:

```
= {
                {"export_model_file",11,0,3,0,1745},
                {"poly_order",9,0,1,0,1741,0,0.,0.,0.,0,"{MLS polynomial order}
        ModelCommands.html#ModelSurrG"},
                {"weight_function",9,0,2,0,1743,0,0.,0.,0.,0,"{MLS weight function}
        ModelCommands.html#ModelSurrG"}
                }
```

GuiKeyWord kw_268[5] [static]

Initial value:

```
= {
                {"export_model_file",11,0,4,0,1755},
                {"max_nodes",9,0,1,0,1749},
                {"nodes",1,0,1,0,1748},
                {"random_weight",9,0,3,0,1753,0,0.,0.,0.,0,"{ANN random weight}
        ModelCommands.html#ModelSurrG"},
                {"range",10,0,2,0,1751,0,0.,0.,0.,0,"{ANN range} ModelCommands.html#ModelSurrG"}
                }
```

GuiKeyWord kw_269[4] [static]

Initial value:

```
= {
                {"cubic",8,0,1,1,1775,0,0.,0.,0.,0,"[CHOOSE polynomial order]"},
                {"export_model_file",11,0,2,0,1777},
                {"linear",8,0,1,1,1771},
                {"quadratic",8,0,1,1,1773}
                }
```

GuiKeyWord kw_270[5] [static]

Initial value:

```
= {
                {"bases",9,0,1,0,1759,0,0.,0.,0.,0,"{RBF number of bases} ModelCommands.html#ModelSurrG"},
                {"export_model_file",11,0,5,0,1767},
                {"max_pts",9,0,2,0,1761,0,0.,0.,0.,0,"{RBF maximum points} ModelCommands.html#ModelSurrG"},
                {"max_subsets",9,0,4,0,1765},
                {"min_partition",9,0,3,0,1763,0,0.,0.,0.,0,"{RBF minimum partitions}
        ModelCommands.html#ModelSurrG"}
                }
```

GuiKeyWord kw_271[3] [static]

Initial value:

```
= {
                {"all",8,0,1,1,1789},
                {"none",8,0,1,1,1793},
                {"region",8,0,1,1,1791}
                }
```

GuiKeyWord kw_272[21] [static]

Initial value:

```
= {
                {"challenge_points_file",11,3,10,0,1835,kw_254,0.,0.,0.,0,"{Challenge file for surrogate
        metrics} ModelCommands.html#ModelSurrG"},
                {"correction",8,6,8,0,1811,kw_255,0.,0.,0.,0,"{Surrogate correction approach}
        ModelCommands.html#ModelSurrG"},
                {"dace_method_pointer",11,0,3,0,1785,0,0.,0.,0.,0,"{Design of experiments method pointer}
        ModelCommands.html#ModelSurrG"},
                {"diagnostics",7,2,9,0,1824,kw_257},
                {"export_points_file",11,2,6,0,1803,kw_258,0.,0.,0.,0,"{File export of global
        approximation-based sample results} ModelCommands.html#ModelSurrG"},
                {"gaussian_process",8,2,1,1,1689,kw_263,0.,0.,0.,0,"[CHOOSE surrogate type]{Dakota Gaussian
        process} ModelCommands.html#ModelSurrG"},
                {"import_points_file",11,3,5,0,1795,kw_264,0.,0.,0.,0,"{File import of samples for global
        approximation builds} ModelCommands.html#ModelSurrG"},
                {"kriging",0,2,1,1,1688,kw_263},
                {"mars",8,3,1,1,1727,kw_266,0.,0.,0.,0,"{Multivariate adaptive regression splines}
        ModelCommands.html#ModelSurrG"},
                {"metrics",15,2,9,0,1825,kw_257,0.,0.,0.,0,"{Compute surrogate diagnostics}
        ModelCommands.html#ModelSurrG"},
                {"minimum_points",8,0,2,0,1781},
                {"moving_least_squares",8,3,1,1,1739,kw_267,0.,0.,0.,0,"{Moving least squares}
        ModelCommands.html#ModelSurrG"},
                {"neural_network",8,5,1,1,1747,kw_268,0.,0.,0.,0,"{Artificial neural network}
        ModelCommands.html#ModelSurrG"},
                {"polynomial",8,4,1,1,1769,kw_269,0.,0.,0.,0,"{Polynomial} ModelCommands.html#ModelSurrG"},
                {"radial_basis",8,5,1,1,1757,kw_270},
```

```
                {"recommended_points",8,0,2,0,1783},
                {"reuse_points",8,3,4,0,1787,kw_271},
                {"reuse_samples",0,3,4,0,1786,kw_271},
                {"samples_file",3,3,5,0,1794,kw_264},
                {"total_points",9,0,2,0,1779},
                {"use_derivatives",8,0,7,0,1809,0,0.,0.,0.,0,"{Surfpack GP gradient enhancement}
        ModelCommands.html#ModelSurrG"}
                }
```

GuiKeyWord kw_273[6] [static]

Initial value:

```
= {
                {"additive",8,0,2,2,1867,0,0.,0.,0.,0,"[CHOOSE correction type]"},
                {"combined",8,0,2,2,1871},
                {"first_order",8,0,1,1,1863,0,0.,0.,0.,0,"[CHOOSE correction order]"},
                {"multiplicative",8,0,2,2,1869},
                {"second_order",8,0,1,1,1865},
                {"zeroth_order",8,0,1,1,1861}
                }
```

GuiKeyWord kw_274[3] [static]

Initial value:

```
= {
                {"correction",8,6,3,3,1859,kw_273,0.,0.,0.,0,"{Surrogate correction approach}
        ModelCommands.html#ModelSurrH"},
                {"high_fidelity_model_pointer",11,0,2,2,1857,0,0.,0.,0.,0,"{Pointer to the high fidelity
        model specification} ModelCommands.html#ModelSurrH"},
                {"low_fidelity_model_pointer",11,0,1,1,1855,0,0.,0.,0.,0,"{Pointer to the low fidelity
        model specification} ModelCommands.html#ModelSurrH"}
                }
```

GuiKeyWord kw_275[2] [static]

Initial value:

```
= {
                {"actual_model_pointer",11,0,2,2,1851,0,0.,0.,0.,0,"{Pointer to the truth model
        specification} ModelCommands.html#ModelSurrMP"},
                {"taylor_series",8,0,1,1,1849,0,0.,0.,0.,0,"{Taylor series local approximation }
        ModelCommands.html#ModelSurrL"}
                }
```

GuiKeyWord kw_276[2] [static]

Initial value:

```
= {
                {"actual_model_pointer",11,0,2,2,1851,0,0.,0.,0.,0,"{Pointer to the truth model
        specification} ModelCommands.html#ModelSurrMP"},
                {"tana",8,0,1,1,1845,0,0.,0.,0.,0,"{Two-point adaptive nonlinear approximation }
        ModelCommands.html#ModelSurrMP"}
                }
```

GuiKeyWord kw_277[5] [static]

Initial value:

```
= {
                {"global",8,21,2,1,1687,kw_272,0.,0.,0.,0,"[CHOOSE surrogate category]{Global
        approximations } ModelCommands.html#ModelSurrG"},
                {"hierarchical",8,3,2,1,1853,kw_274,0.,0.,0.,0,"{Hierarchical approximation }
        ModelCommands.html#ModelSurrH"},
                {"id_surrogates",13,0,1,0,1685,0,0.,0.,0.,0,"{Surrogate response ids}
        ModelCommands.html#ModelSurrogate"},
                {"local",8,2,2,1,1847,kw_275,0.,0.,0.,0,"{Local approximation}
        ModelCommands.html#ModelSurrL"},
                {"multipoint",8,2,2,1,1843,kw_276,0.,0.,0.,0,"{Multipoint approximation}
        ModelCommands.html#ModelSurrMP"}
                }
```

GuiKeyWord kw_278[7] [static]

Initial value:

```
= {
                {"hierarchical_tagging",8,0,4,0,1677,0,0.,0.,0.,0,"{Hierarchical evaluation tags}
        ModelCommands.html#ModelIndControl"},
                {"id_model",11,0,1,0,1671,0,0.,0.,0.,0,"{Model set identifier}
        ModelCommands.html#ModelIndControl"},
                {"nested",8,2,5,1,1873,kw_252,0.,0.,0.,0,"[CHOOSE model type]"},
                {"responses_pointer",11,0,3,0,1675,0,0.,0.,0.,0,"{Responses set pointer}
        ModelCommands.html#ModelIndControl"},
                {"single",8,1,5,1,1679,kw_253,0.,0.,0.,0,"@"},
                {"surrogate",8,5,5,1,1683,kw_277},
                {"variables_pointer",11,0,2,0,1673,0,0.,0.,0.,0,"{Variables set pointer}
        ModelCommands.html#ModelIndControl"}
                }
```

GuiKeyWord kw_279[6] [static]

Initial value:

```
= {
                {"annotated",8,0,2,0,2513,0,0.,0.,0.,0,"{Data file in annotated format}
        RespCommands.html#RespFnLS"},
                {"freeform",8,0,2,0,2515,0,0.,0.,0.,0,"{Data file in freeform format}
        RespCommands.html#RespFnLS"},
                {"num_config_variables",0x29,0,3,0,2517,0,0.,0.,0.,0,"{Configuration variable columns in
        file} RespCommands.html#RespFnLS"},
                {"num_experiments",0x29,0,1,0,2511,0,0.,0.,0.,0,"{Experiments in file}
        RespCommands.html#RespFnLS"},
                {"num_std_deviations",0x29,0,5,0,2521,0,0.,0.,0.,0,"{Standard deviation columns in file}
        RespCommands.html#RespFnLS"},
                {"sigma_type",0x80f,0,4,0,2519,0,0.,0.,0.,0,0,0,"calibration_terms"}
                }
```

GuiKeyWord kw_280[9] [static]

Initial value:

```
= {
                {"annotated",8,0,2,0,2555},
                {"config_data_file",11,0,6,0,2565},
                {"field_coordinate_data_file",11,0,7,0,2567},
                {"field_data_file",11,0,5,0,2563},
                {"freeform",8,0,2,0,2557},
                {"num_config_variables",0x29,0,3,0,2559},
                {"num_experiments",0x29,0,1,0,2553},
                {"sigma_data_file",11,0,8,0,2569},
                {"sigma_type",0x80f,0,4,0,2561,0,0.,0.,0.,0,0,0,"field_calibration_terms"}
                }
```

GuiKeyWord kw_281[5] [static]

Initial value:

```
= {
                {"coordinate_data_file",11,0,3,0,2549},
                {"coordinate_list",14,0,3,0,2547},
                {"field_data",8,9,4,0,2551,kw_280},
                {"lengths",13,0,1,1,2543},
                {"num_coordinates_per_field",13,0,2,0,2545}
                }
```

GuiKeyWord kw_282[6] [static]

Initial value:

```
= {
                {"nonlinear_equality_scale_types",0x807,0,2,0,2536,0,0.,0.,0.,0,0,0,"
      nonlinear_equality_constraints"},
                {"nonlinear_equality_scales",0x806,0,3,0,2538,0,0.,0.,0.,0,0,0,"
      nonlinear_equality_constraints"},
                {"nonlinear_equality_targets",6,0,1,0,2534,0,0.,0.,0.,0,0,0,"nonlinear_equality_constraints
      "},
                {"scale_types",0x80f,0,2,0,2537,0,0.,0.,0.,0,0,0,"nonlinear_equality_constraints"},
                {"scales",0x80e,0,3,0,2539,0,0.,0.,0.,0,0,0,"nonlinear_equality_constraints"},
                {"targets",14,0,1,0,2535,0,0.,0.,0.,0,"{Nonlinear equality targets}
       RespCommands.html#RespFnLS",0,"nonlinear_equality_constraints"}
                }
```

GuiKeyWord kw_283[8] [static]

Initial value:

```
= {
                {"lower_bounds",14,0,1,0,2525,0,0.,0.,0.,0,"{Nonlinear inequality lower bounds}
       RespCommands.html#RespFnLS",0,"nonlinear_inequality_constraints"},
                {"nonlinear_inequality_lower_bounds",6,0,1,0,2524,0,0.,0.,0.,0,0,0,"
      nonlinear_inequality_constraints"},
                {"nonlinear_inequality_scale_types",0x807,0,3,0,2528,0,0.,0.,0.,0,0,0,"
      nonlinear_inequality_constraints"},
                {"nonlinear_inequality_scales",0x806,0,4,0,2530,0,0.,0.,0.,0,0,0,"
      nonlinear_inequality_constraints"},
                {"nonlinear_inequality_upper_bounds",6,0,2,0,2526,0,0.,0.,0.,0,0,0,"
      nonlinear_inequality_constraints"},
                {"scale_types",0x80f,0,3,0,2529,0,0.,0.,0.,0,0,0,"nonlinear_inequality_constraints"},
                {"scales",0x80e,0,4,0,2531,0,0.,0.,0.,0,0,0,"nonlinear_inequality_constraints"},
                {"upper_bounds",14,0,2,0,2527,0,0.,0.,0.,0,"{Nonlinear inequality upper bounds}
       RespCommands.html#RespFnLS",0,"nonlinear_inequality_constraints"}
                }
```

GuiKeyWord kw_284[19] [static]

Initial value:

```
= {
                {"calibration_data_file",11,6,5,0,2509,kw_279,0.,0.,0.,0,"{Calibration data file name}
       RespCommands.html#RespFnLS"},
                {"calibration_term_scale_types",0x807,0,1,0,2500,0,0.,0.,0.,0,0,0,"calibration_terms"},
                {"calibration_term_scales",0x806,0,2,0,2502,0,0.,0.,0.,0,0,0,"calibration_terms"},
                {"calibration_weights",6,0,3,0,2504,0,0.,0.,0.,0,0,0,"calibration_terms"},
                {"field_calibration_terms",0x29,5,8,0,2541,kw_281},
                {"least_squares_data_file",3,6,5,0,2508,kw_279},
                {"least_squares_term_scale_types",0x807,0,1,0,2500,0,0.,0.,0.,0,0,0,"calibration_terms"},
                {"least_squares_term_scales",0x806,0,2,0,2502,0,0.,0.,0.,0,0,0,"calibration_terms"},
```

```
                {"least_squares_weights",6,0,3,0,2504,0,0.,0.,0.,0,0,0,"calibration_terms"},
                {"nonlinear_equality_constraints",0x29,6,7,0,2533,kw_282,0.,0.,0.,0,"{Number of nonlinear
        equality constraints} RespCommands.html#RespFnLS"},
                {"nonlinear_inequality_constraints",0x29,8,6,0,2523,kw_283,0.,0.,0.,0,"{Number of nonlinear
        inequality constraints} RespCommands.html#RespFnLS"},
                {"num_field_calibration_terms",0x21,5,8,0,2540,kw_281},
                {"num_nonlinear_equality_constraints",0x21,6,7,0,2532,kw_282},
                {"num_nonlinear_inequality_constraints",0x21,8,6,0,2522,kw_283},
                {"num_scalar_calibration_terms",0x21,0,4,0,2506},
                {"primary_scale_types",0x80f,0,1,0,2501,0,0.,0.,0.,0,"{Calibration scaling types}
        RespCommands.html#RespFnLS",0,"calibration_terms"},
                {"primary_scales",0x80e,0,2,0,2503,0,0.,0.,0.,0,"{Calibration scales}
        RespCommands.html#RespFnLS",0,"calibration_terms"},
                {"scalar_calibration_terms",0x29,0,4,0,2507},
                {"weights",14,0,3,0,2505,0,0.,0.,0.,0,"{Calibration term weights}
        RespCommands.html#RespFnLS",0,"calibration_terms"}
                }
```

GuiKeyWord kw_285[4] `[static]`

Initial value:

```
= {
                {"absolute",8,0,2,0,2605},
                {"bounds",8,0,2,0,2607},
                {"ignore_bounds",8,0,1,0,2601,0,0.,0.,0.,0,"{Ignore variable bounds}
        RespCommands.html#RespGradMixed"},
                {"relative",8,0,2,0,2603}
                }
```

GuiKeyWord kw_286[10] `[static]`

Initial value:

```
= {
                {"central",8,0,6,0,2615,0,0.,0.,0.,0,"[CHOOSE difference interval]"},
                {"dakota",8,4,4,0,2599,kw_285,0.,0.,0.,0,"@[CHOOSE gradient source]{Interval scaling type}
        RespCommands.html#RespGradNum"},
                {"fd_gradient_step_size",6,0,7,0,2616},
                {"fd_step_size",14,0,7,0,2617,0,0.,0.,0.,0,"{Finite difference step size}
        RespCommands.html#RespGradMixed"},
                {"forward",8,0,6,0,2613,0,0.,0.,0.,0,"@"},
                {"id_analytic_gradients",13,0,2,2,2593,0,0.,0.,0.,0,"{Analytic derivatives function list}
        RespCommands.html#RespGradMixed"},
                {"id_numerical_gradients",13,0,1,1,2591,0,0.,0.,0.,0,"{Numerical derivatives function list}
        RespCommands.html#RespGradMixed"},
                {"interval_type",8,0,5,0,2611,0,0.,0.,0.,0,"{Interval type} RespCommands.html#RespGradNum"}
      ,
                {"method_source",8,0,3,0,2597,0,0.,0.,0.,0,"{Method source} RespCommands.html#RespGradNum"}
      ,
                {"vendor",8,0,4,0,2609}
                }
```

GuiKeyWord kw_287[2] `[static]`

Initial value:

```
= {
                {"fd_hessian_step_size",6,0,1,0,2648},
                {"fd_step_size",14,0,1,0,2649,0,0.,0.,0.,0,"{Finite difference step size}
        RespCommands.html#RespHessMixed"}
                }
```

GuiKeyWord kw_288[1] [static]

Initial value:

```
= {
                {"damped",8,0,1,0,2665,0,0.,0.,0.,0,"{Numerical safeguarding of BFGS update}
        RespCommands.html#RespHessMixed"}
                }
```

GuiKeyWord kw_289[2] [static]

Initial value:

```
= {
                {"bfgs",8,1,1,1,2663,kw_288,0.,0.,0.,0,"[CHOOSE Hessian approx.]"},
                {"sr1",8,0,1,1,2667}
                }
```

GuiKeyWord kw_290[8] [static]

Initial value:

```
= {
                {"absolute",8,0,2,0,2653},
                {"bounds",8,0,2,0,2655},
                {"central",8,0,3,0,2659,0,0.,0.,0.,0,"[CHOOSE difference interval]"},
                {"forward",8,0,3,0,2657,0,0.,0.,0.,0,"@"},
                {"id_analytic_hessians",13,0,5,0,2669,0,0.,0.,0.,0,"{Analytic Hessians function list}
        RespCommands.html#RespHessMixed"},
                {"id_numerical_hessians",13,2,1,0,2647,kw_287,0.,0.,0.,0,"{Numerical Hessians function
        list} RespCommands.html#RespHessMixed"},
                {"id_quasi_hessians",13,2,4,0,2661,kw_289,0.,0.,0.,0,"{Quasi Hessians function list}
        RespCommands.html#RespHessMixed"},
                {"relative",8,0,2,0,2651}
                }
```

GuiKeyWord kw_291[4] [static]

Initial value:

```
= {
                {"coordinate_data_file",11,0,3,0,2497},
                {"coordinate_list",14,0,3,0,2495},
                {"lengths",13,0,1,1,2491},
                {"num_coordinates_per_field",13,0,2,0,2493}
                }
```

GuiKeyWord kw_292[6] [static]

Initial value:

```
= {
                {"nonlinear_equality_scale_types",0x807,0,2,0,2482,0,0.,0.,0.,0,0,0,"
       nonlinear_equality_constraints"},
                {"nonlinear_equality_scales",0x806,0,3,0,2484,0,0.,0.,0.,0,0,0,"
       nonlinear_equality_constraints"},
                {"nonlinear_equality_targets",6,0,1,0,2480,0,0.,0.,0.,0,0,0,"nonlinear_equality_constraints
       "},
                {"scale_types",0x80f,0,2,0,2483,0,0.,0.,0.,0,"{Nonlinear scaling types (for inequalities or
        equalities)} RespCommands.html#RespFnLS",0,"nonlinear_equality_constraints"},
                {"scales",0x80e,0,3,0,2485,0,0.,0.,0.,0,"{Nonlinear scales (for inequalities or
        equalities)} RespCommands.html#RespFnLS",0,"nonlinear_equality_constraints"},
                {"targets",14,0,1,0,2481,0,0.,0.,0.,0,"{Nonlinear equality constraint targets}
        RespCommands.html#RespFnOpt",0,"nonlinear_equality_constraints"}
                }
```

GuiKeyWord kw_293[8] [static]

Initial value:

```
= {
                {"lower_bounds",14,0,1,0,2471,0,0.,0.,0.,0,"{Nonlinear inequality constraint lower bounds}
       RespCommands.html#RespFnOpt",0,"nonlinear_inequality_constraints"},
                {"nonlinear_inequality_lower_bounds",6,0,1,0,2470,0,0.,0.,0.,0,0,0,"
      nonlinear_inequality_constraints"},
                {"nonlinear_inequality_scale_types",0x807,0,3,0,2474,0,0.,0.,0.,0,0,0,"
      nonlinear_inequality_constraints"},
                {"nonlinear_inequality_scales",0x806,0,4,0,2476,0,0.,0.,0.,0,0,0,"
      nonlinear_inequality_constraints"},
                {"nonlinear_inequality_upper_bounds",6,0,2,0,2472,0,0.,0.,0.,0,0,0,"
      nonlinear_inequality_constraints"},
                {"scale_types",0x80f,0,3,0,2475,0,0.,0.,0.,0,"{Nonlinear constraint scaling types (for
       inequalities or equalities)} RespCommands.html#RespFnOpt",0,"nonlinear_inequality_constraints"},
                {"scales",0x80e,0,4,0,2477,0,0.,0.,0.,0,"{Nonlinear constraint scales (for inequalities or
       equalities)} RespCommands.html#RespFnOpt",0,"nonlinear_inequality_constraints"},
                {"upper_bounds",14,0,2,0,2473,0,0.,0.,0.,0,"{Nonlinear inequality constraint upper bounds}
       RespCommands.html#RespFnOpt",0,"nonlinear_inequality_constraints"}
                }
```

GuiKeyWord kw_294[15] [static]

Initial value:

```
= {
                {"field_objectives",0x29,4,8,0,2489,kw_291},
                {"multi_objective_weights",6,0,4,0,2466,0,0.,0.,0.,0,0,0,"objective_functions"},
                {"nonlinear_equality_constraints",0x29,6,6,0,2479,kw_292,0.,0.,0.,0,"{Number of nonlinear
       equality constraints} RespCommands.html#RespFnOpt"},
                {"nonlinear_inequality_constraints",0x29,8,5,0,2469,kw_293,0.,0.,0.,0,"{Number of nonlinear
       inequality constraints} RespCommands.html#RespFnOpt"},
                {"num_field_objectives",0x21,4,8,0,2488,kw_291},
                {"num_nonlinear_equality_constraints",0x21,6,6,0,2478,kw_292},
                {"num_nonlinear_inequality_constraints",0x21,8,5,0,2468,kw_293},
                {"num_scalar_objectives",0x21,0,7,0,2486},
                {"objective_function_scale_types",0x807,0,2,0,2462,0,0.,0.,0.,0,0,0,"objective_functions"},
                {"objective_function_scales",0x806,0,3,0,2464,0,0.,0.,0.,0,0,0,"objective_functions"},
                {"primary_scale_types",0x80f,0,2,0,2463,0,0.,0.,0.,0,"{Objective function scaling types}
       RespCommands.html#RespFnOpt",0,"objective_functions"},
                {"primary_scales",0x80e,0,3,0,2465,0,0.,0.,0.,0,"{Objective function scales}
       RespCommands.html#RespFnOpt",0,"objective_functions"},
                {"scalar_objectives",0x29,0,7,0,2487},
                {"sense",0x80f,0,1,0,2461,0,0.,0.,0.,0,"{Optimization sense} RespCommands.html#RespFnOpt",0
      ,"objective_functions"},
                {"weights",14,0,4,0,2467,0,0.,0.,0.,0,"{Multi-objective weightings}
       RespCommands.html#RespFnOpt",0,"objective_functions"}
                }
```

GuiKeyWord kw_295[4] [static]

Initial value:

```
= {
                {"coordinate_data_file",11,0,3,0,2583},
                {"coordinate_list",14,0,3,0,2581},
                {"lengths",13,0,1,1,2577},
                {"num_coordinates_per_field",13,0,2,0,2579}
                }
```

GuiKeyWord kw_296[4] [static]

Initial value:

```
= {
                {"field_responses",0x29,4,2,0,2575,kw_295},
                {"num_field_responses",0x21,4,2,0,2574,kw_295},
                {"num_scalar_responses",0x21,0,1,0,2572},
                {"scalar_responses",0x29,0,1,0,2573}
                }
```

GuiKeyWord kw_297[8] [static]

Initial value:

```
= {
                {"central",8,0,6,0,2615,0,0.,0.,0.,0,"[CHOOSE difference interval]"},
                {"dakota",8,4,4,0,2599,kw_285,0.,0.,0.,0,"@[CHOOSE gradient source]{Interval scaling type}
        RespCommands.html#RespGradNum"},
                {"fd_gradient_step_size",6,0,7,0,2616},
                {"fd_step_size",14,0,7,0,2617,0,0.,0.,0.,0,"{Finite difference step size}
        RespCommands.html#RespGradMixed"},
                {"forward",8,0,6,0,2613,0,0.,0.,0.,0,"@"},
                {"interval_type",8,0,5,0,2611,0,0.,0.,0.,0,"{Interval type} RespCommands.html#RespGradNum"}
      ,
                {"method_source",8,0,3,0,2597,0,0.,0.,0.,0,"{Method source} RespCommands.html#RespGradNum"}
      ,
                {"vendor",8,0,4,0,2609}
                }
```

GuiKeyWord kw_298[7] [static]

Initial value:

```
= {
                {"absolute",8,0,2,0,2627},
                {"bounds",8,0,2,0,2629},
                {"central",8,0,3,0,2633,0,0.,0.,0.,0,"[CHOOSE difference interval]"},
                {"fd_hessian_step_size",6,0,1,0,2622},
                {"fd_step_size",14,0,1,0,2623,0,0.,0.,0.,0,"{Finite difference step size}
        RespCommands.html#RespHessNum"},
                {"forward",8,0,3,0,2631,0,0.,0.,0.,0,"@"},
                {"relative",8,0,2,0,2625}
                }
```

GuiKeyWord kw_299[1] [static]

Initial value:

```
= {
                {"damped",8,0,1,0,2639,0,0.,0.,0.,0,"{Numerical safeguarding of BFGS update}
        RespCommands.html#RespHessQuasi"}
                }
```

GuiKeyWord kw_300[2] [static]

Initial value:

```
= {
                {"bfgs",8,1,1,1,2637,kw_299,0.,0.,0.,0,"[CHOOSE Hessian approx.]"},
                {"sr1",8,0,1,1,2641}
                }
```

GuiKeyWord kw_301[19] [static]

Initial value:

```
= {
                {"analytic_gradients",8,0,4,2,2587,0,0.,0.,0.,0,"[CHOOSE gradient type]"},
                {"analytic_hessians",8,0,5,3,2643,0,0.,0.,0.,0,"[CHOOSE Hessian type]"},
                {"calibration_terms",0x29,19,3,1,2499,kw_284,0.,0.,0.,0,"{{Calibration (Least squares)}
      Number of calibration terms} RespCommands.html#RespFnLS"},
                {"descriptors",15,0,2,0,2457,0,0.,0.,0.,0,"{Response labels} RespCommands.html#RespLabels"}
     ,
                {"id_responses",11,0,1,0,2455,0,0.,0.,0.,0,"{Responses set identifier}
      RespCommands.html#RespSetId"},
                {"least_squares_terms",0x21,19,3,1,2498,kw_284},
                {"mixed_gradients",8,10,4,2,2589,kw_286,0.,0.,0.,0,"{Mixed gradients}
      RespCommands.html#RespGradMixed"},
                {"mixed_hessians",8,8,5,3,2645,kw_290,0.,0.,0.,0,"{Mixed Hessians}
      RespCommands.html#RespHessMixed"},
                {"no_gradients",8,0,4,2,2585,0,0.,0.,0.,0,"@"},
                {"no_hessians",8,0,5,3,2619,0,0.,0.,0.,0,"@"},
                {"num_least_squares_terms",0x21,19,3,1,2498,kw_284},
                {"num_objective_functions",0x21,15,3,1,2458,kw_294},
                {"num_response_functions",0x21,4,3,1,2570,kw_296},
                {"numerical_gradients",8,8,4,2,2595,kw_297,0.,0.,0.,0,"{Numerical gradients}
      RespCommands.html#RespGradNum"},
                {"numerical_hessians",8,7,5,3,2621,kw_298,0.,0.,0.,0,"{Numerical Hessians}
      RespCommands.html#RespHessNum"},
                {"objective_functions",0x29,15,3,1,2459,kw_294,0.,0.,0.,0,"{{Optimization} Number of
      objective functions} RespCommands.html#RespFnOpt"},
                {"quasi_hessians",8,2,5,3,2635,kw_300,0.,0.,0.,0,"{Quasi Hessians}
      RespCommands.html#RespHessQuasi"},
                {"response_descriptors",7,0,2,0,2456},
                {"response_functions",0x29,4,3,1,2571,kw_296,0.,0.,0.,0,"{{Generic responses} Number of
      response functions} RespCommands.html#RespFnGen"}
                }
```

GuiKeyWord kw_302[6] [static]

Initial value:

```
= {
                {"aleatory",8,0,1,1,1911},
                {"all",8,0,1,1,1905},
                {"design",8,0,1,1,1907},
                {"epistemic",8,0,1,1,1913},
                {"state",8,0,1,1,1915},
                {"uncertain",8,0,1,1,1909}
                }
```

GuiKeyWord kw_303[11] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,2057,0,0.,0.,0.,0,"{beta uncertain alphas} VarCommands.html#VarCAUV_Beta
     ",0,"beta_uncertain"},
                {"betas",14,0,2,2,2059,0,0.,0.,0.,0,"{beta uncertain betas} VarCommands.html#VarCAUV_Beta",
     0,"beta_uncertain"},
                {"buv_alphas",6,0,1,1,2056,0,0.,0.,0.,0,0,0,"beta_uncertain"},
                {"buv_betas",6,0,2,2,2058,0,0.,0.,0.,0,0,0,"beta_uncertain"},
                {"buv_descriptors",7,0,6,0,2066,0,0.,0.,0.,0,0,0,"beta_uncertain"},
                {"buv_lower_bounds",6,0,3,3,2060,0,0.,0.,0.,0,0,0,"beta_uncertain"},
                {"buv_upper_bounds",6,0,4,4,2062,0,0.,0.,0.,0,0,0,"beta_uncertain"},
                {"descriptors",15,0,6,0,2067,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCAUV_Gamma",0,
     "beta_uncertain"},
                {"initial_point",14,0,5,0,2065,0,0.,0.,0.,0,0,0,"beta_uncertain"},
                {"lower_bounds",14,0,3,3,2061,0,0.,0.,0.,0,"{Distribution lower bounds}
```

```
VarCommands.html#VarCAUV_Beta",0,"beta_uncertain"},
        {"upper_bounds",14,0,4,4,2063,0,0.,0.,0.,0,"{Distribution upper bounds}
VarCommands.html#VarCAUV_Beta",0,"beta_uncertain"}
        }
```

GuiKeyWord kw_304[5] [static]

Initial value:

```
= {
        {"descriptors",15,0,4,0,2139,0,0.,0.,0.,0,"{Descriptors}
VarCommands.html#VarDAUV_Negative_Binomial",0,"binomial_uncertain"},
        {"initial_point",13,0,3,0,2137,0,0.,0.,0.,0,0,0,"binomial_uncertain"},
        {"num_trials",13,0,2,2,2135,0,0.,0.,0.,0,"{binomial uncertain num_trials}
VarCommands.html#VarDAUV_Binomial",0,"binomial_uncertain"},
        {"prob_per_trial",6,0,1,1,2132,0,0.,0.,0.,0,0,0,"binomial_uncertain"},
        {"probability_per_trial",14,0,1,1,2133,0,0.,0.,0.,0,0,0,"binomial_uncertain"}
        }
```

GuiKeyWord kw_305[12] [static]

Initial value:

```
= {
        {"cdv_descriptors",7,0,6,0,1932,0,0.,0.,0.,0,0,0,"continuous_design"},
        {"cdv_initial_point",6,0,1,0,1922,0,0.,0.,0.,0,0,0,"continuous_design"},
        {"cdv_lower_bounds",6,0,2,0,1924,0,0.,0.,0.,0,0,0,"continuous_design"},
        {"cdv_scale_types",0x807,0,4,0,1928,0,0.,0.,0.,0,0,0,"continuous_design"},
        {"cdv_scales",0x806,0,5,0,1930,0,0.,0.,0.,0,0,0,"continuous_design"},
        {"cdv_upper_bounds",6,0,3,0,1926,0,0.,0.,0.,0,0,0,"continuous_design"},
        {"descriptors",15,0,6,0,1933,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCDV",0,"
continuous_design"},
        {"initial_point",14,0,1,0,1923,0,0.,0.,0.,0,"{Initial point} VarCommands.html#VarCDV",0,"
continuous_design"},
        {"lower_bounds",14,0,2,0,1925,0,0.,0.,0.,0,"{Lower bounds} VarCommands.html#VarCDV",0,"
continuous_design"},
        {"scale_types",0x80f,0,4,0,1929,0,0.,0.,0.,0,"{Scaling types} VarCommands.html#VarCDV",0,"
continuous_design"},
        {"scales",0x80e,0,5,0,1931,0,0.,0.,0.,0,"{Scales} VarCommands.html#VarCDV",0,"
continuous_design"},
        {"upper_bounds",14,0,3,0,1927,0,0.,0.,0.,0,"{Upper bounds} VarCommands.html#VarCDV",0,"
continuous_design"}
        }
```

GuiKeyWord kw_306[10] [static]

Initial value:

```
= {
        {"descriptors",15,0,6,0,2223,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDUSRV",0,"
continuous_interval_uncertain"},
        {"initial_point",14,0,5,0,2221,0,0.,0.,0.,0,0,0,"continuous_interval_uncertain"},
        {"interval_probabilities",14,0,2,0,2215,0,0.,0.,0.,0,"{basic probability assignments per
 continuous interval} VarCommands.html#VarCEUV_Interval"},
        {"interval_probs",6,0,2,0,2214},
        {"iuv_descriptors",7,0,6,0,2222,0,0.,0.,0.,0,0,0,"continuous_interval_uncertain"},
        {"iuv_interval_probs",6,0,2,0,2214},
        {"iuv_num_intervals",5,0,1,0,2212,0,0.,0.,0.,0,0,0,"continuous_interval_uncertain"},
        {"lower_bounds",14,0,3,1,2217,0,0.,0.,0.,0,"{lower bounds of continuous intervals}
VarCommands.html#VarCEUV_Interval"},
        {"num_intervals",13,0,1,0,2213,0,0.,0.,0.,0,"{number of intervals defined for each
 continuous interval variable} VarCommands.html#VarCEUV_Interval",0,"continuous_interval_uncertain"},
        {"upper_bounds",14,0,4,2,2219,0,0.,0.,0.,0,"{upper bounds of continuous intervals}
VarCommands.html#VarCEUV_Interval"}
        }
```

GuiKeyWord kw_307[8] [static]

Initial value:

```
= {
                {"csv_descriptors",7,0,4,0,2288,0,0.,0.,0.,0,0,0,"continuous_state"},
                {"csv_initial_state",6,0,1,0,2282,0,0.,0.,0.,0,0,0,"continuous_state"},
                {"csv_lower_bounds",6,0,2,0,2284,0,0.,0.,0.,0,0,0,"continuous_state"},
                {"csv_upper_bounds",6,0,3,0,2286,0,0.,0.,0.,0,0,0,"continuous_state"},
                {"descriptors",15,0,4,0,2289,0,0.,0.,0.,0,0,0,"continuous_state"},
                {"initial_state",14,0,1,0,2283,0,0.,0.,0.,0,"{Initial states} VarCommands.html#VarCSV",0,"
      continuous_state"},
                {"lower_bounds",14,0,2,0,2285,0,0.,0.,0.,0,"{Lower bounds} VarCommands.html#VarCSV",0,"
      continuous_state"},
                {"upper_bounds",14,0,3,0,2287,0,0.,0.,0.,0,"{Upper bounds} VarCommands.html#VarCSV",0,"
      continuous_state"}
                }
```

GuiKeyWord kw_308[8] [static]

Initial value:

```
= {
                {"ddv_descriptors",7,0,4,0,1942,0,0.,0.,0.,0,0,0,"discrete_design_range"},
                {"ddv_initial_point",5,0,1,0,1936,0,0.,0.,0.,0,0,0,"discrete_design_range"},
                {"ddv_lower_bounds",5,0,2,0,1938,0,0.,0.,0.,0,0,0,"discrete_design_range"},
                {"ddv_upper_bounds",5,0,3,0,1940,0,0.,0.,0.,0,0,0,"discrete_design_range"},
                {"descriptors",15,0,4,0,1943,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDDRIV",0,"
      discrete_design_range"},
                {"initial_point",13,0,1,0,1937,0,0.,0.,0.,0,"{Initial point} VarCommands.html#VarDDRIV",0,"
      discrete_design_range"},
                {"lower_bounds",13,0,2,0,1939,0,0.,0.,0.,0,"{Lower bounds} VarCommands.html#VarDDRIV",0,"
      discrete_design_range"},
                {"upper_bounds",13,0,3,0,1941,0,0.,0.,0.,0,"{Upper bounds} VarCommands.html#VarDDRIV",0,"
      discrete_design_range"}
                }
```

GuiKeyWord kw_309[7] [static]

Initial value:

```
= {
                {"categorical",15,0,3,0,1953,0,0.,0.,0.,0,0,0,"integer"},
                {"descriptors",15,0,5,0,1957,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDDSIV",0,"
      integer"},
                {"elements",13,0,2,1,1951},
                {"elements_per_variable",0x80d,0,1,0,1949,0,0.,0.,0.,0,0,0,"integer"},
                {"initial_point",13,0,4,0,1955,0,0.,0.,0.,0,"{Initial point} VarCommands.html#VarDDSIV",0,"
      integer"},
                {"num_set_values",0x805,0,1,0,1948,0,0.,0.,0.,0,0,0,"integer"},
                {"set_values",5,0,2,1,1950}
                }
```

GuiKeyWord kw_310[7] [static]

Initial value:

```
= {
                {"categorical",15,0,3,0,1975,0,0.,0.,0.,0,0,0,"integer"},
                {"descriptors",15,0,5,0,1979,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCAUV_Normal",0
      ,"real"},
                {"elements",14,0,2,1,1973},
                {"elements_per_variable",0x80d,0,1,0,1971,0,0.,0.,0.,0,0,0,"real"},
                {"initial_point",14,0,4,0,1977,0,0.,0.,0.,0,0,0,"real"},
                {"num_set_values",0x805,0,1,0,1970,0,0.,0.,0.,0,0,0,"real"},
                {"set_values",6,0,2,1,1972}
                }
```

GuiKeyWord kw_311[6] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,1967,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDDSRV",0,"
      string"},
                {"elements",15,0,2,1,1963},
                {"elements_per_variable",0x80d,0,1,0,1961,0,0.,0.,0.,0,0,0,"string"},
                {"initial_point",15,0,3,0,1965,0,0.,0.,0.,0,"{Initial point} VarCommands.html#VarDDSRV",0,"
      string"},
                {"num_set_values",0x805,0,1,0,1960,0,0.,0.,0.,0,0,0,"string"},
                {"set_values",7,0,2,1,1962}
                }
```

GuiKeyWord kw_312[3] [static]

Initial value:

```
= {
                {"integer",0x19,7,1,0,1947,kw_309},
                {"real",0x19,7,3,0,1969,kw_310},
                {"string",0x19,6,2,0,1959,kw_311}
                }
```

GuiKeyWord kw_313[9] [static]

Initial value:

```
= {
                {"descriptors",15,0,6,0,2237,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCSV",0,"
      discrete_interval_uncertain"},
                {"initial_point",13,0,5,0,2235,0,0.,0.,0.,0,0,0,"discrete_interval_uncertain"},
                {"interval_probabilities",14,0,2,0,2229,0,0.,0.,0.,0,"{Basic probability assignments per
       interval} VarCommands.html#VarDIUV"},
                {"interval_probs",6,0,2,0,2228},
                {"lower_bounds",13,0,3,1,2231,0,0.,0.,0.,0,"{Lower bounds} VarCommands.html#VarDIUV"},
                {"num_intervals",13,0,1,0,2227,0,0.,0.,0.,0,"{Number of intervals defined for each interval
       variable} VarCommands.html#VarDIUV",0,"discrete_interval_uncertain"},
                {"range_probabilities",6,0,2,0,2228},
                {"range_probs",6,0,2,0,2228},
                {"upper_bounds",13,0,4,2,2233,0,0.,0.,0.,0,"{Upper bounds} VarCommands.html#VarDIUV"}
                }
```

GuiKeyWord kw_314[8] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,2299,0,0.,0.,0.,0,0,0,"discrete_state_range"},
                {"dsv_descriptors",7,0,4,0,2298,0,0.,0.,0.,0,0,0,"discrete_state_range"},
                {"dsv_initial_state",5,0,1,0,2292,0,0.,0.,0.,0,0,0,"discrete_state_range"},
                {"dsv_lower_bounds",5,0,2,0,2294,0,0.,0.,0.,0,0,0,"discrete_state_range"},
                {"dsv_upper_bounds",5,0,3,0,2296,0,0.,0.,0.,0,0,0,"discrete_state_range"},
                {"initial_state",13,0,1,0,2293,0,0.,0.,0.,0,"{Initial states} VarCommands.html#VarDSRIV",0,
      "discrete_state_range"},
                {"lower_bounds",13,0,2,0,2295,0,0.,0.,0.,0,"{Lower bounds} VarCommands.html#VarDSRIV",0,"
      discrete_state_range"},
                {"upper_bounds",13,0,3,0,2297,0,0.,0.,0.,0,"{Upper bounds} VarCommands.html#VarDSRIV",0,"
      discrete_state_range"}
                }
```

GuiKeyWord kw_315[7] [static]

Initial value:

```
= {
                {"categorical",15,0,3,0,2309,0,0.,0.,0.,0,0,0,"integer"},
                {"descriptors",15,0,5,0,2313,0,0.,0.,0.,0,0,0,"integer"},
                {"elements",13,0,2,1,2307},
                {"elements_per_variable",0x80d,0,1,0,2305,0,0.,0.,0.,0,0,0,"integer"},
                {"initial_state",13,0,4,0,2311,0,0.,0.,0.,0,"{Initial state} VarCommands.html#VarDSSIV",0,"
      integer"},
                {"num_set_values",0x805,0,1,0,2304,0,0.,0.,0.,0,0,0,"integer"},
                {"set_values",5,0,2,1,2306}
                }
```

GuiKeyWord kw_316[7] [static]

Initial value:

```
= {
                {"categorical",15,0,3,0,2331,0,0.,0.,0.,0,0,0,"integer"},
                {"descriptors",15,0,5,0,2335,0,0.,0.,0.,0,0,0,"real"},
                {"elements",14,0,2,1,2329},
                {"elements_per_variable",0x80d,0,1,0,2327,0,0.,0.,0.,0,0,0,"real"},
                {"initial_state",14,0,4,0,2333,0,0.,0.,0.,0,0,0,"real"},
                {"num_set_values",0x805,0,1,0,2326,0,0.,0.,0.,0,0,0,"real"},
                {"set_values",6,0,2,1,2328}
                }
```

GuiKeyWord kw_317[6] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,2323,0,0.,0.,0.,0,0,0,"string"},
                {"elements",15,0,2,1,2319},
                {"elements_per_variable",0x80d,0,1,0,2317,0,0.,0.,0.,0,0,0,"string"},
                {"initial_state",15,0,3,0,2321,0,0.,0.,0.,0,"{Initial state} VarCommands.html#VarDSSRV",0,"
      string"},
                {"num_set_values",0x805,0,1,0,2316,0,0.,0.,0.,0,0,0,"string"},
                {"set_values",7,0,2,1,2318}
                }
```

GuiKeyWord kw_318[3] [static]

Initial value:

```
= {
                {"integer",0x19,7,1,0,2303,kw_315},
                {"real",0x19,7,3,0,2325,kw_316},
                {"string",0x19,6,2,0,2315,kw_317}
                }
```

GuiKeyWord kw_319[9] [static]

Initial value:

```
= {
                {"categorical",15,0,4,0,2249,0,0.,0.,0.,0,0,0,"integer"},
                {"descriptors",15,0,6,0,2253,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDSRIV",0,"
      integer"},
                {"elements",13,0,2,1,2245},
                {"elements_per_variable",13,0,1,0,2243,0,0.,0.,0.,0,0,0,"integer"},
                {"initial_point",13,0,5,0,2251,0,0.,0.,0.,0,0,0,"integer"},
                {"num_set_values",5,0,1,0,2242,0,0.,0.,0.,0,0,0,"integer"},
                {"set_probabilities",14,0,3,0,2247,0,0.,0.,0.,0,"{Probabilities for each set member}
       VarCommands.html#VarDUSIV"},
                {"set_probs",6,0,3,0,2246},
                {"set_values",5,0,2,1,2244}
                }
```

GuiKeyWord kw_320[9] [static]

Initial value:

```
= {
                {"categorical",15,0,4,0,2275,0,0.,0.,0.,0,0,0,"real"},
                {"descriptors",15,0,6,0,2279,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDSSRV",0,"real
      "},
                {"elements",14,0,2,1,2271},
                {"elements_per_variable",13,0,1,0,2269,0,0.,0.,0.,0,0,0,"real"},
                {"initial_point",14,0,5,0,2277,0,0.,0.,0.,0,0,0,"real"},
                {"num_set_values",5,0,1,0,2268,0,0.,0.,0.,0,0,0,"real"},
                {"set_probabilities",14,0,3,0,2273},
                {"set_probs",6,0,3,0,2272},
                {"set_values",6,0,2,1,2270}
                }
```

GuiKeyWord kw_321[8] [static]

Initial value:

```
= {
                {"descriptors",15,0,5,0,2265,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDSSIV",0,"
      string"},
                {"elements",15,0,2,1,2259},
                {"elements_per_variable",13,0,1,0,2257,0,0.,0.,0.,0,0,0,"string"},
                {"initial_point",15,0,4,0,2263,0,0.,0.,0.,0,0,0,"string"},
                {"num_set_values",5,0,1,0,2256,0,0.,0.,0.,0,0,0,"string"},
                {"set_probabilities",14,0,3,0,2261,0,0.,0.,0.,0,"{Probabilities for each set member}
       VarCommands.html#VarDUSRV"},
                {"set_probs",6,0,3,0,2260},
                {"set_values",7,0,2,1,2258}
                }
```

GuiKeyWord kw_322[3] [static]

Initial value:

```
= {
                {"integer",0x19,9,1,0,2241,kw_319},
                {"real",0x19,9,3,0,2267,kw_320},
                {"string",0x19,8,2,0,2255,kw_321}
                }
```

GuiKeyWord kw_323[5] [static]

Initial value:

```
= {
                {"betas",14,0,1,1,2049,0,0.,0.,0.,0,"{exponential uncertain betas}
        VarCommands.html#VarCAUV_Exponential",0,"exponential_uncertain"},
                {"descriptors",15,0,3,0,2053,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCAUV_Beta",0,"
       exponential_uncertain"},
                {"euv_betas",6,0,1,1,2048,0,0.,0.,0.,0,0,0,"exponential_uncertain"},
                {"euv_descriptors",7,0,3,0,2052,0,0.,0.,0.,0,0,0,"exponential_uncertain"},
                {"initial_point",14,0,2,0,2051,0,0.,0.,0.,0,0,0,"exponential_uncertain"}
                }
```

GuiKeyWord kw_324[7] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,2091,0,0.,0.,0.,0,"{frechet uncertain alphas}
        VarCommands.html#VarCAUV_Frechet",0,"frechet_uncertain"},
                {"betas",14,0,2,2,2093,0,0.,0.,0.,0,"{frechet uncertain betas}
        VarCommands.html#VarCAUV_Frechet",0,"frechet_uncertain"},
                {"descriptors",15,0,4,0,2097,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCAUV_Weibull",
       0,"frechet_uncertain"},
                {"fuv_alphas",6,0,1,1,2090,0,0.,0.,0.,0,0,0,"frechet_uncertain"},
                {"fuv_betas",6,0,2,2,2092,0,0.,0.,0.,0,0,0,"frechet_uncertain"},
                {"fuv_descriptors",7,0,4,0,2096,0,0.,0.,0.,0,0,0,"frechet_uncertain"},
                {"initial_point",14,0,3,0,2095,0,0.,0.,0.,0,0,0,"frechet_uncertain"}
                }
```

GuiKeyWord kw_325[7] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,2071,0,0.,0.,0.,0,"{gamma uncertain alphas}
        VarCommands.html#VarCAUV_Gamma",0,"gamma_uncertain"},
                {"betas",14,0,2,2,2073,0,0.,0.,0.,0,"{gamma uncertain betas} VarCommands.html#VarCAUV_Gamma
       ",0,"gamma_uncertain"},
                {"descriptors",15,0,4,0,2077,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCAUV_Gumbel",0
       ,"gamma_uncertain"},
                {"gauv_alphas",6,0,1,1,2070,0,0.,0.,0.,0,0,0,"gamma_uncertain"},
                {"gauv_betas",6,0,2,2,2072,0,0.,0.,0.,0,0,0,"gamma_uncertain"},
                {"gauv_descriptors",7,0,4,0,2076,0,0.,0.,0.,0,0,0,"gamma_uncertain"},
                {"initial_point",14,0,3,0,2075,0,0.,0.,0.,0,0,0,"gamma_uncertain"}
                }
```

GuiKeyWord kw_326[4] [static]

Initial value:

```
= {
                {"descriptors",15,0,3,0,2157,0,0.,0.,0.,0,"{Descriptors}
        VarCommands.html#VarDAUV_Hypergeometric",0,"geometric_uncertain"},
                {"initial_point",13,0,2,0,2155,0,0.,0.,0.,0,0,0,"geometric_uncertain"},
                {"prob_per_trial",6,0,1,1,2152,0,0.,0.,0.,0,0,0,"geometric_uncertain"},
                {"probability_per_trial",14,0,1,1,2153,0,0.,0.,0.,0,0,0,"geometric_uncertain"}
                }
```

GuiKeyWord kw_327[7] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,2081,0,0.,0.,0.,0,"{gumbel uncertain alphas}
        VarCommands.html#VarCAUV_Gumbel",0,"gumbel_uncertain"},
                {"betas",14,0,2,2,2083,0,0.,0.,0.,0,"{gumbel uncertain betas}
        VarCommands.html#VarCAUV_Gumbel",0,"gumbel_uncertain"},
                {"descriptors",15,0,4,0,2087,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCAUV_Frechet",
       0,"gumbel_uncertain"},
                {"guuv_alphas",6,0,1,1,2080,0,0.,0.,0.,0,0,0,"gumbel_uncertain"},
                {"guuv_betas",6,0,2,2,2082,0,0.,0.,0.,0,0,0,"gumbel_uncertain"},
                {"guuv_descriptors",7,0,4,0,2086,0,0.,0.,0.,0,0,0,"gumbel_uncertain"},
                {"initial_point",14,0,3,0,2085,0,0.,0.,0.,0,0,0,"gumbel_uncertain"}
                }
```

GuiKeyWord kw_328[11] [static]

Initial value:

```
= {
                {"abscissas",14,0,2,1,2113,0,0.,0.,0.,0,"{sets of abscissas for bin-based histogram
        variables} VarCommands.html#VarCAUV_Bin_Histogram"},
                {"counts",14,0,3,2,2117,0,0.,0.,0.,0,"{sets of counts for bin-based histogram variables}
        VarCommands.html#VarCAUV_Bin_Histogram"},
                {"descriptors",15,0,5,0,2121,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDAUV_Poisson",
       0,"histogram_bin_uncertain"},
                {"huv_bin_abscissas",6,0,2,1,2112},
                {"huv_bin_counts",6,0,3,2,2116},
                {"huv_bin_descriptors",7,0,5,0,2120,0,0.,0.,0.,0,0,0,"histogram_bin_uncertain"},
                {"huv_bin_ordinates",6,0,3,2,2114},
                {"initial_point",14,0,4,0,2119,0,0.,0.,0.,0,0,0,"histogram_bin_uncertain"},
                {"num_pairs",5,0,1,0,2110,0,0.,0.,0.,0,0,0,"histogram_bin_uncertain"},
                {"ordinates",14,0,3,2,2115,0,0.,0.,0.,0,"{sets of ordinates for bin-based histogram
        variables} VarCommands.html#VarCAUV_Bin_Histogram"},
                {"pairs_per_variable",13,0,1,0,2111,0,0.,0.,0.,0,0,0,"histogram_bin_uncertain"}
                }
```

GuiKeyWord kw_329[6] [static]

Initial value:

```
= {
                {"abscissas",13,0,2,1,2177,0,0.,0.,0.,0,"{sets of abscissas for point-based histogram
        variables} VarCommands.html#VarDAUV_Point_Histogram"},
                {"counts",14,0,3,2,2179,0,0.,0.,0.,0,"{sets of counts for point-based histogram variables}
        VarCommands.html#VarDAUV_Point_Histogram"},
                {"descriptors",15,0,5,0,2183,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCEUV_Interval"
       ,0,"integer"},
                {"initial_point",13,0,4,0,2181,0,0.,0.,0.,0,0,0,"integer"},
                {"num_pairs",5,0,1,0,2174,0,0.,0.,0.,0,0,0,"integer"},
                {"pairs_per_variable",13,0,1,0,2175,0,0.,0.,0.,0,0,0,"integer"}
                }
```

GuiKeyWord kw_330[6] [static]

Initial value:

```
= {
                {"abscissas",14,0,2,1,2201},
                {"counts",14,0,3,2,2203},
                {"descriptors",15,0,5,0,2207,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDUSIV",0,"real
       "},
                {"initial_point",14,0,4,0,2205,0,0.,0.,0.,0,0,0,"real"},
                {"num_pairs",5,0,1,0,2198,0,0.,0.,0.,0,0,0,"real"},
                {"pairs_per_variable",13,0,1,0,2199,0,0.,0.,0.,0,0,0,"real"}
                }
```

GuiKeyWord kw_331[6] [static]

Initial value:

```
= {
                {"abscissas",15,0,2,1,2189},
                {"counts",14,0,3,2,2191},
                {"descriptors",15,0,5,0,2195,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDIUV",0,"
      string"},
                {"initial_point",15,0,4,0,2193,0,0.,0.,0.,0,0,0,"string"},
                {"num_pairs",5,0,1,0,2186,0,0.,0.,0.,0,0,0,"string"},
                {"pairs_per_variable",13,0,1,0,2187,0,0.,0.,0.,0,0,0,"string"}
                }
```

GuiKeyWord kw_332[3] [static]

Initial value:

```
= {
                {"integer",0x19,6,1,0,2173,kw_329},
                {"real",0x19,6,3,0,2197,kw_330},
                {"string",0x19,6,2,0,2185,kw_331}
                }
```

GuiKeyWord kw_333[5] [static]

Initial value:

```
= {
                {"descriptors",15,0,5,0,2169,0,0.,0.,0.,0,"{Descriptors}
      VarCommands.html#VarDAUV_Point_Histogram",0,"hypergeometric_uncertain"},
                {"initial_point",13,0,4,0,2167,0,0.,0.,0.,0,0,0,"hypergeometric_uncertain"},
                {"num_drawn",13,0,3,3,2165,0,0.,0.,0.,0,"{hypergeometric uncertain num_drawn }
      VarCommands.html#VarDAUV_Hypergeometric",0,"hypergeometric_uncertain"},
                {"selected_population",13,0,2,2,2163,0,0.,0.,0.,0,"{hypergeometric uncertain
      selected_population} VarCommands.html#VarDAUV_Hypergeometric",0,"hypergeometric_uncertain"},
                {"total_population",13,0,1,1,2161,0,0.,0.,0.,0,"{hypergeometric uncertain total_population}
      VarCommands.html#VarDAUV_Hypergeometric",0,"hypergeometric_uncertain"}
                }
```

GuiKeyWord kw_334[2] [static]

Initial value:

```
= {
                {"lnuv_zetas",6,0,1,1,1998,0,0.,0.,0.,0,0,0,"lognormal_uncertain"},
                {"zetas",14,0,1,1,1999,0,0.,0.,0.,0,"{lognormal uncertain zetas}
      VarCommands.html#VarCAUV_Lognormal",0,"lognormal_uncertain"}
                }
```

GuiKeyWord kw_335[4] [static]

Initial value:

```
= {
                {"error_factors",14,0,1,1,2005,0,0.,0.,0.,0,"[CHOOSE variance spec.]{lognormal uncertain
      error factors} VarCommands.html#VarCAUV_Lognormal",0,"lognormal_uncertain"},
                {"lnuv_error_factors",6,0,1,1,2004,0,0.,0.,0.,0,0,0,"lognormal_uncertain"},
                {"lnuv_std_deviations",6,0,1,1,2002,0,0.,0.,0.,0,0,0,"lognormal_uncertain"},
                {"std_deviations",14,0,1,1,2003,0,0.,0.,0.,0,"@{lognormal uncertain standard deviations}
      VarCommands.html#VarCAUV_Lognormal",0,"lognormal_uncertain"}
                }
```

GuiKeyWord kw_336[11] [static]

Initial value:

```
= {
                {"descriptors",15,0,5,0,2013,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCAUV_Uniform",
      0,"lognormal_uncertain"},
                {"initial_point",14,0,4,0,2011,0,0.,0.,0.,0,0,0,"lognormal_uncertain"},
                {"lambdas",14,2,1,1,1997,kw_334,0.,0.,0.,0,"[CHOOSE characterization]{lognormal uncertain
       lambdas} VarCommands.html#VarCAUV_Lognormal",0,"lognormal_uncertain"},
                {"lnuv_descriptors",7,0,5,0,2012,0,0.,0.,0.,0,0,0,"lognormal_uncertain"},
                {"lnuv_lambdas",6,2,1,1,1996,kw_334,0.,0.,0.,0,0,0,"lognormal_uncertain"},
                {"lnuv_lower_bounds",6,0,2,0,2006,0,0.,0.,0.,0,0,0,"lognormal_uncertain"},
                {"lnuv_means",6,4,1,1,2000,kw_335,0.,0.,0.,0,0,0,"lognormal_uncertain"},
                {"lnuv_upper_bounds",6,0,3,0,2008,0,0.,0.,0.,0,0,0,"lognormal_uncertain"},
                {"lower_bounds",14,0,2,0,2007,0,0.,0.,0.,0,"{Distribution lower bounds}
       VarCommands.html#VarCAUV_Lognormal",0,"lognormal_uncertain"},
                {"means",14,4,1,1,2001,kw_335,0.,0.,0.,0,"@{lognormal uncertain means}
       VarCommands.html#VarCAUV_Lognormal",0,"lognormal_uncertain"},
                {"upper_bounds",14,0,3,0,2009,0,0.,0.,0.,0,"{Distribution upper bounds}
       VarCommands.html#VarCAUV_Lognormal",0,"lognormal_uncertain"}
                }
```

GuiKeyWord kw_337[7] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,2033,0,0.,0.,0.,0,"{Descriptors}
       VarCommands.html#VarCAUV_Triangular",0,"loguniform_uncertain"},
                {"initial_point",14,0,3,0,2031,0,0.,0.,0.,0,0,0,"loguniform_uncertain"},
                {"lower_bounds",14,0,1,1,2027,0,0.,0.,0.,0,"{Distribution lower bounds}
       VarCommands.html#VarCAUV_Loguniform",0,"loguniform_uncertain"},
                {"luuv_descriptors",7,0,4,0,2032,0,0.,0.,0.,0,0,0,"loguniform_uncertain"},
                {"luuv_lower_bounds",6,0,1,1,2026,0,0.,0.,0.,0,0,0,"loguniform_uncertain"},
                {"luuv_upper_bounds",6,0,2,2,2028,0,0.,0.,0.,0,0,0,"loguniform_uncertain"},
                {"upper_bounds",14,0,2,2,2029,0,0.,0.,0.,0,"{Distribution upper bounds}
       VarCommands.html#VarCAUV_Loguniform",0,"loguniform_uncertain"}
                }
```

GuiKeyWord kw_338[5] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,2149,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDAUV_Geometric
      ",0,"negative_binomial_uncertain"},
                {"initial_point",13,0,3,0,2147,0,0.,0.,0.,0,0,0,"negative_binomial_uncertain"},
                {"num_trials",13,0,2,2,2145,0,0.,0.,0.,0,"{negative binomial uncertain success num_trials}
       VarCommands.html#VarDAUV_Negative_Binomial",0,"negative_binomial_uncertain"},
                {"prob_per_trial",6,0,1,1,2142,0,0.,0.,0.,0,0,0,"negative_binomial_uncertain"},
                {"probability_per_trial",14,0,1,1,2143,0,0.,0.,0.,0,0,0,"negative_binomial_uncertain"}
                }
```

GuiKeyWord kw_339[11] [static]

Initial value:

```
= {
                {"descriptors",15,0,6,0,1993,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarCAUV_Lognormal
      ",0,"normal_uncertain"},
                {"initial_point",14,0,5,0,1991,0,0.,0.,0.,0,0,0,"normal_uncertain"},
                {"lower_bounds",14,0,3,0,1987,0,0.,0.,0.,0,"{Distribution lower bounds}
       VarCommands.html#VarCAUV_Normal",0,"normal_uncertain"},
```

```
                {"means",14,0,1,1,1983,0,0.,0.,0.,0,"{normal uncertain means}
        VarCommands.html#VarCAUV_Normal",0,"normal_uncertain"},
                {"nuv_descriptors",7,0,6,0,1992,0,0.,0.,0.,0,0,0,"normal_uncertain"},
                {"nuv_lower_bounds",6,0,3,0,1986,0,0.,0.,0.,0,0,0,"normal_uncertain"},
                {"nuv_means",6,0,1,1,1982,0,0.,0.,0.,0,0,0,"normal_uncertain"},
                {"nuv_std_deviations",6,0,2,2,1984,0,0.,0.,0.,0,0,0,"normal_uncertain"},
                {"nuv_upper_bounds",6,0,4,0,1988,0,0.,0.,0.,0,0,0,"normal_uncertain"},
                {"std_deviations",14,0,2,2,1985,0,0.,0.,0.,0,"{normal uncertain standard deviations}
        VarCommands.html#VarCAUV_Normal",0,"normal_uncertain"},
                {"upper_bounds",14,0,4,0,1989,0,0.,0.,0.,0,"{Distribution upper bounds}
        VarCommands.html#VarCAUV_Normal",0,"normal_uncertain"}
                }
```

GuiKeyWord kw_340[3] [static]

Initial value:

```
= {
                {"descriptors",15,0,3,0,2129,0,0.,0.,0.,0,"{Descriptors} VarCommands.html#VarDAUV_Binomial"
       ,0,"poisson_uncertain"},
                {"initial_point",13,0,2,0,2127,0,0.,0.,0.,0,0,0,"poisson_uncertain"},
                {"lambdas",14,0,1,1,2125,0,0.,0.,0.,0,"{poisson uncertain lambdas}
        VarCommands.html#VarDAUV_Poisson",0,"poisson_uncertain"}
                }
```

GuiKeyWord kw_341[9] [static]

Initial value:

```
= {
                {"descriptors",15,0,5,0,2045,0,0.,0.,0.,0,"{Descriptors}
        VarCommands.html#VarCAUV_Exponential",0,"triangular_uncertain"},
                {"initial_point",14,0,4,0,2043,0,0.,0.,0.,0,0,0,"triangular_uncertain"},
                {"lower_bounds",14,0,2,2,2039,0,0.,0.,0.,0,"{Distribution lower bounds}
        VarCommands.html#VarCAUV_Triangular",0,"triangular_uncertain"},
                {"modes",14,0,1,1,2037,0,0.,0.,0.,0,"{triangular uncertain modes}
        VarCommands.html#VarCAUV_Triangular",0,"triangular_uncertain"},
                {"tuv_descriptors",7,0,5,0,2044,0,0.,0.,0.,0,0,0,"triangular_uncertain"},
                {"tuv_lower_bounds",6,0,2,2,2038,0,0.,0.,0.,0,0,0,"triangular_uncertain"},
                {"tuv_modes",6,0,1,1,2036,0,0.,0.,0.,0,0,0,"triangular_uncertain"},
                {"tuv_upper_bounds",6,0,3,3,2040,0,0.,0.,0.,0,0,0,"triangular_uncertain"},
                {"upper_bounds",14,0,3,3,2041,0,0.,0.,0.,0,"{Distribution upper bounds}
        VarCommands.html#VarCAUV_Triangular",0,"triangular_uncertain"}
                }
```

GuiKeyWord kw_342[7] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,2023,0,0.,0.,0.,0,"{Descriptors}
        VarCommands.html#VarCAUV_Loguniform",0,"uniform_uncertain"},
                {"initial_point",14,0,3,0,2021,0,0.,0.,0.,0,0,0,"uniform_uncertain"},
                {"lower_bounds",14,0,1,1,2017,0,0.,0.,0.,0,"{Distribution lower bounds}
        VarCommands.html#VarCAUV_Uniform",0,"uniform_uncertain"},
                {"upper_bounds",14,0,2,2,2019,0,0.,0.,0.,0,"{Distribution upper bounds}
        VarCommands.html#VarCAUV_Uniform",0,"uniform_uncertain"},
                {"uuv_descriptors",7,0,4,0,2022,0,0.,0.,0.,0,0,0,"uniform_uncertain"},
                {"uuv_lower_bounds",6,0,1,1,2016,0,0.,0.,0.,0,0,0,"uniform_uncertain"},
                {"uuv_upper_bounds",6,0,2,2,2018,0,0.,0.,0.,0,0,0,"uniform_uncertain"}
                }
```

GuiKeyWord kw_343[7] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,2101,0,0.,0.,0.,0,"{weibull uncertain alphas}
        VarCommands.html#VarCAUV_Weibull",0,"weibull_uncertain"},
                {"betas",14,0,2,2,2103,0,0.,0.,0.,0,"{weibull uncertain betas}
        VarCommands.html#VarCAUV_Weibull",0,"weibull_uncertain"},
                {"descriptors",15,0,4,0,2107,0,0.,0.,0.,0,"{Descriptors}
        VarCommands.html#VarCAUV_Bin_Histogram",0,"weibull_uncertain"},
                {"initial_point",14,0,3,0,2105,0,0.,0.,0.,0,0,0,"weibull_uncertain"},
                {"wuv_alphas",6,0,1,1,2100,0,0.,0.,0.,0,0,0,"weibull_uncertain"},
                {"wuv_betas",6,0,2,2,2102,0,0.,0.,0.,0,0,0,"weibull_uncertain"},
                {"wuv_descriptors",7,0,4,0,2106,0,0.,0.,0.,0,0,0,"weibull_uncertain"}
                }
```

GuiKeyWord kw_345[6] [static]

Initial value:

```
= {
                {"environment",0x108,15,1,1,1,kw_7,0.,0.,0.,0,"{Environment} The environment specifies the
        top level technique which will govern the management of iterators and models in the solution of the problem
        of interest. EnvCommands.html"},
                {"interface",0x308,9,5,5,2337,kw_21,0.,0.,0.,0,"{Interface} An interface specifies how
        function evaluations will be performed in order to map a set of parameters into a set of responses.
        InterfCommands.html"},
                {"method",0x308,90,2,2,47,kw_248,0.,0.,0.,0,"{Method} A method specifies the name and
        controls of an iterative procedure, e.g., a sensitivity analysis, uncertainty quantification, or optimization
        method. MethodCommands.html"},
                {"model",8,7,3,3,1669,kw_278,0.,0.,0.,0,"{Model} A model consists of a model type and maps
        specified variables through an interface to generate responses. ModelCommands.html"},
                {"responses",0x308,19,6,6,2453,kw_301,0.,0.,0.,0,"{Responses} A responses object specifies
        the data that can be returned to DAKOTA through the interface after the completion of a function evaluation.
        RespCommands.html"},
                {"variables",0x308,34,4,4,1899,kw_344,0.,0.,0.,0,"{Variables} A variables object specifies
        the parameter set to be iterated by a particular method. VarCommands.html"}
                }
```

KeyWord kw_1[2] [static]

Initial value:

```
= {
                {"input",11,0,1,0,0,0.,0.,0,N_stm(str,postRunInput)},
                {"output",11,0,2,0,0,0.,0.,0,N_stm(str,postRunOutput)}
                }
```

1335 distinct keywords (plus 207 aliases)

KeyWord kw_2[2] [static]

Initial value:

```
= {
                {"input",11,0,1,0,0,0.,0.,0,N_stm(str,preRunInput)},
                {"output",11,0,2,0,0,0.,0.,0,N_stm(str,preRunOutput)}
                }
```

KeyWord kw_3[1] [static]

Initial value:

```
= {
                {"stop_restart",0x29,0,1,0,0,0.,0.,0,N_stm(int,stopRestart)}
                }
```

KeyWord kw_4[1] [static]

Initial value:

```
= {
                {"results_output_file",11,0,1,0,0,0.,0.,0,N_stm(str,resultsOutputFile)}
                }
```

KeyWord kw_5[2] [static]

Initial value:

```
= {
                {"input",11,0,1,0,0,0.,0.,0,N_stm(str,runInput)},
                {"output",11,0,2,0,0,0.,0.,0,N_stm(str,runOutput)}
                }
```

KeyWord kw_6[2] [static]

Initial value:

```
= {
                {"tabular_data_file",11,0,1,0,0,0.,0.,0,N_stm(str,tabularDataFile)},
                {"tabular_graphics_file",3,0,1,0,0,0.,0.,-1,N_stm(str,tabularDataFile)}
                }
```

KeyWord kw_7[15] [static]

Initial value:

```
= {
                {"check",8,0,1,0,0,0.,0.,0,N_stm(true,checkFlag)},
                {"error_file",11,0,3,0,0,0.,0.,0,N_stm(str,errorFile)},
                {"graphics",8,0,9,0,0,0.,0.,0,N_stm(true,graphicsFlag)},
                {"method_pointer",3,0,13,0,0,0.,0.,10,N_stm(str,topMethodPointer)},
                {"output_file",11,0,2,0,0,0.,0.,0,N_stm(str,outputFile)},
                {"output_precision",0x29,0,11,0,0,0.,0.,0,N_stm(int,outputPrecision)},
                {"post_run",8,2,8,0,kw_1,0.,0.,0,N_stm(true,postRunFlag)},
                {"pre_run",8,2,6,0,kw_2,0.,0.,0,N_stm(true,preRunFlag)},
                {"read_restart",11,1,4,0,kw_3,0.,0.,0,N_stm(str,readRestart)},
                {"results_output",8,1,12,0,kw_4,0.,0.,0,N_stm(true,resultsOutputFlag)},
                {"run",8,2,7,0,kw_5,0.,0.,0,N_stm(true,runFlag)},
                {"tabular_data",8,2,10,0,kw_6,0.,0.,0,N_stm(true,tabularDataFlag)},
                {"tabular_graphics_data",0,2,10,0,kw_6,0.,0.,-1,N_stm(true,tabularDataFlag)},
                {"top_method_pointer",11,0,13,0,0,0.,0.,0,N_stm(str,topMethodPointer)},
                {"write_restart",11,0,5,0,0,0.,0.,0,N_stm(str,writeRestart)}
                }
```

KeyWord kw_8[1] [static]

Initial value:

```
= {
                {"cache_tolerance",10,0,1,0,0,0.,0.,0,N_ifm(Real,nearbyEvalCacheTol)}
                }
```

KeyWord kw_9[4] [static]

Initial value:

```
= {
                {"active_set_vector",8,0,1,0,0,0.,0.,0,N_ifm(false,activeSetVectorFlag)},
                {"evaluation_cache",8,0,2,0,0,0.,0.,0,N_ifm(false,evalCacheFlag)},
                {"restart_file",8,0,4,0,0,0.,0.,0,N_ifm(false,restartFileFlag)},
                {"strict_cache_equality",8,1,3,0,kw_8,0.,0.,0,N_ifm(true,nearbyEvalCacheFlag)}
                }
```

KeyWord kw_10[1] [static]

Initial value:

```
= {
                {"processors_per_analysis",0x19,0,1,0,0,0.,0.,0,N_ifm(pint,procsPerAnalysis)}
                }
```

KeyWord kw_11[4] [static]

Initial value:

```
= {
                {"abort",8,0,1,1,0,0.,0.,0,N_ifm(lit,failAction_abort)},
                {"continuation",8,0,1,1,0,0.,0.,0,N_ifm(lit,failAction_continuation)},
                {"recover",14,0,1,1,0,0.,0.,0,N_ifm(Rlit,TYPE_DATA_failAction_recover)},
                {"retry",9,0,1,1,0,0.,0.,0,N_ifm(ilit,TYPE_DATA_failAction_retry)}
                }
```

KeyWord kw_12[1] [static]

Initial value:

```
= {
                {"numpy",8,0,1,0,0,0.,0.,0,N_ifm(true,numpyFlag)}
                }
```

KeyWord kw_13[8] [static]

Initial value:

```
= {
                {"copy_files",15,0,5,0,0,0.,0.,0,N_ifm(strL,copyFiles)},
                {"dir_save",0,0,3,0,0,0.,0.,2,N_ifm(true,dirSave)},
                {"dir_tag",0,0,2,0,0,0.,0.,2,N_ifm(true,dirTag)},
                {"directory_save",8,0,3,0,0,0.,0.,0,N_ifm(true,dirSave)},
                {"directory_tag",8,0,2,0,0,0.,0.,0,N_ifm(true,dirTag)},
                {"link_files",15,0,4,0,0,0.,0.,0,N_ifm(strL,linkFiles)},
                {"named",11,0,1,0,0,0.,0.,0,N_ifm(str,workDir)},
                {"replace",8,0,6,0,0,0.,0.,0,N_ifm(true,templateReplace)}
                }
```

KeyWord kw_14[9] [static]

Initial value:

```
= {
                {"allow_existing_results",8,0,3,0,0,0.,0.,0,N_ifm(true,allowExistingResultsFlag)},
                {"aprepro",8,0,5,0,0,0.,0.,0,N_ifm(true,apreproFlag)},
                {"dprepro",0,0,5,0,0,0.,0.,-1,N_ifm(true,apreproFlag)},
                {"file_save",8,0,7,0,0,0.,0.,0,N_ifm(true,fileSaveFlag)},
                {"file_tag",8,0,6,0,0,0.,0.,0,N_ifm(true,fileTagFlag)},
                {"parameters_file",11,0,1,0,0,0.,0.,0,N_ifm(str,parametersFile)},
                {"results_file",11,0,2,0,0,0.,0.,0,N_ifm(str,resultsFile)},
                {"verbatim",8,0,4,0,0,0.,0.,0,N_ifm(true,verbatimFlag)},
                {"work_directory",8,8,8,0,kw_13,0.,0.,0,N_ifm(true,useWorkdir)}
                }
```

KeyWord kw_15[12] [static]

Initial value:

```
= {
                {"analysis_components",15,0,1,0,0,0.,0.,0,N_ifm(str2D,analysisComponents)},
                {"deactivate",8,4,6,0,kw_9},
                {"direct",8,1,4,1,kw_10,0.,0.,0,N_ifm(type,interfaceType_TEST_INTERFACE)},
                {"failure_capture",8,4,5,0,kw_11},
                {"fork",8,9,4,1,kw_14,0.,0.,0,N_ifm(type,interfaceType_FORK_INTERFACE)},
                {"grid",8,0,4,1,0,0.,0.,0,N_ifm(type,interfaceType_GRID_INTERFACE)},
                {"input_filter",11,0,2,0,0,0.,0.,0,N_ifm(str,inputFilter)},
                {"matlab",8,0,4,1,0,0.,0.,0,N_ifm(type,interfaceType_MATLAB_INTERFACE)},
                {"output_filter",11,0,3,0,0,0.,0.,0,N_ifm(str,outputFilter)},
                {"python",8,1,4,1,kw_12,0.,0.,0,N_ifm(type,interfaceType_PYTHON_INTERFACE)},
                {"scilab",8,0,4,1,0,0.,0.,0,N_ifm(type,interfaceType_SCILAB_INTERFACE)},
                {"system",8,9,4,1,kw_14,0.,0.,0,N_ifm(type,interfaceType_SYSTEM_INTERFACE)}
                }
```

KeyWord kw_16[2] [static]

Initial value:

```
= {
                {"master",8,0,1,1,0,0.,0.,0,N_ifm(type,analysisScheduling_MASTER_SCHEDULING)},
                {"peer",8,0,1,1,0,0.,0.,0,N_ifm(type,analysisScheduling_PEER_SCHEDULING)}
                }
```

KeyWord kw_17[2] [static]

Initial value:

```
= {
                {"dynamic",8,0,1,1,0,0.,0.,0,N_ifm(type,asynchLocalEvalScheduling_DYNAMIC_SCHEDULING)},
                {"static",8,0,1,1,0,0.,0.,0,N_ifm(type,asynchLocalEvalScheduling_STATIC_SCHEDULING)}
                }
```

KeyWord kw_18[3] [static]

Initial value:

```
= {
                {"analysis_concurrency",0x19,0,3,0,0,0.,0.,0,N_ifm(pint,asynchLocalAnalysisConcurrency)},
                {"evaluation_concurrency",0x19,0,1,0,0,0.,0.,0,N_ifm(pint,asynchLocalEvalConcurrency)},
                {"local_evaluation_scheduling",8,2,2,0,kw_17}
                }
```

KeyWord kw_19[2] [static]

Initial value:

```
= {
                {"dynamic",8,0,1,1,0,0.,0.,0,N_ifm(type,evalScheduling_PEER_DYNAMIC_SCHEDULING)},
                {"static",8,0,1,1,0,0.,0.,0,N_ifm(type,evalScheduling_PEER_STATIC_SCHEDULING)}
                }
```

KeyWord kw_20[2] [static]

Initial value:

```
= {
                {"master",8,0,1,1,0,0.,0.,0,N_ifm(type,evalScheduling_MASTER_SCHEDULING)},
                {"peer",8,2,1,1,kw_19}
                }
```

KeyWord kw_21[9] [static]

Initial value:

```
= {
                {"algebraic_mappings",11,0,2,0,0,0.,0.,0,N_ifm(str,algebraicMappings)},
                {"analysis_drivers",15,12,3,0,kw_15,0.,0.,0,N_ifm(strL,analysisDrivers)},
                {"analysis_scheduling",8,2,9,0,kw_16},
                {"analysis_servers",0x19,0,8,0,0,0.,0.,0,N_ifm(pint,analysisServers)},
                {"asynchronous",8,3,4,0,kw_18,0.,0.,0,N_ifm(type,
     interfaceSynchronization_ASYNCHRONOUS_INTERFACE)},
                {"evaluation_scheduling",8,2,6,0,kw_20},
                {"evaluation_servers",0x19,0,5,0,0,0.,0.,0,N_ifm(pint,evalServers)},
                {"id_interface",11,0,1,0,0,0.,0.,0,N_ifm(str,idInterface)},
                {"processors_per_evaluation",0x19,0,7,0,0,0.,0.,0,N_ifm(pint,procsPerEval)}
                }
```

KeyWord kw_22[1] [static]

Initial value:

```
= {
                {"model_pointer",11,0,1,0,0,0.,0.,0,N_mdm(str,modelPointer)}
                }
```

KeyWord kw_23[2] [static]

Initial value:

```
= {
                {"samples",9,0,1,0,0,0.,0.,0,N_mdm(int,numSamples)},
                {"seed",0x19,0,2,0,0,0.,0.,0,N_mdm(pint,randomSeed)}
                }
```

KeyWord kw_24[2] [static]

Initial value:

```
= {
                {"complementary",8,0,1,1,0,0.,0.,0,N_mdm(type,distributionType_COMPLEMENTARY)},
                {"cumulative",8,0,1,1,0,0.,0.,0,N_mdm(type,distributionType_CUMULATIVE)}
                }
```

KeyWord kw_25[1] [static]

Initial value:

```
= {
                {"num_gen_reliability_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,genReliabilityLevels)}
                }
```

KeyWord kw_26[1] [static]

Initial value:

```
= {
                {"num_probability_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,probabilityLevels)}
                }
```

KeyWord kw_27[2] [static]

Initial value:

```
= {
                {"mt19937",8,0,1,1,0,0.,0.,0,N_mdm(lit,rngName_mt19937)},
                {"rnum2",8,0,1,1,0,0.,0.,0,N_mdm(lit,rngName_rnum2)}
                }
```

KeyWord kw_28[4] [static]

Initial value:

```
= {
                {"distribution",8,2,1,0,kw_24},
                {"gen_reliability_levels",14,1,3,0,kw_25,0.,0.,0,N_mdm(resplevs,genReliabilityLevels)},
                {"probability_levels",14,1,2,0,kw_26,0.,0.,0,N_mdm(resplevs01,probabilityLevels)},
                {"rng",8,2,4,0,kw_27}
                }
```

KeyWord kw_29[4] [static]

Initial value:

```
= {
                {"constant_liar",8,0,1,1,0,0.,0.,0,N_mdm(lit,batchSelectionType_constant_liar)},
                {"distance_penalty",8,0,1,1,0,0.,0.,0,N_mdm(lit,batchSelectionType_distance_penalty)},
                {"naive",8,0,1,1,0,0.,0.,0,N_mdm(lit,batchSelectionType_naive)},
                {"topology",8,0,1,1,0,0.,0.,0,N_mdm(lit,batchSelectionType_topology)}
                }
```

KeyWord kw_30[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_31[3] [static]

Initial value:

```
= {
                {"distance",8,0,1,1,0,0.,0.,0,N_mdm(lit,fitnessMetricType_distance)},
                {"gradient",8,0,1,1,0,0.,0.,0,N_mdm(lit,fitnessMetricType_gradient)},
                {"predicted_variance",8,0,1,1,0,0.,0.,0,N_mdm(lit,fitnessMetricType_predicted_variance)}
                }
```

KeyWord kw_32[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_33[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
                {"series",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
                }
```

KeyWord kw_34[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
                {"probabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
                {"system",8,2,2,0,kw_33}
                }
```

KeyWord kw_35[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,kw_34},
                {"num_response_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,responseLevels)}
                }
```

KeyWord kw_36[11] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {0,0,4,0,0,kw_28},
                {"batch_selection",8,4,3,0,kw_29},
                {"batch_size",9,0,4,0,0,0.,0.,0,N_mdm(int,batchSize)},
                {"emulator_samples",9,0,1,0,0,0.,0.,0,N_mdm(int,emulatorSamples)},
                {"export_points_file",11,2,6,0,kw_30,0.,0.,0,N_mdm(str,approxExportFile)},
                {"fitness_metric",8,3,2,0,kw_31},
                {"import_points_file",11,3,5,0,kw_32,0.,0.,0,N_mdm(str,approxImportFile)},
                {"misc_options",15,0,8,0,0,0.,0.,0,N_mdm(strL,miscOptions)},
                {"response_levels",14,2,7,0,kw_35,0.,0.,0,N_mdm(resplevs,responseLevels)}
                }
```

KeyWord kw_37[9] [static]

Initial value:

```
= {
                {"linear_equality_constraint_matrix",14,0,6,0,0,0.,0.,0,N_mdm(RealDL,
      linearEqConstraintCoeffs)},
                {"linear_equality_scale_types",15,0,8,0,0,0.,0.,0,N_mdm(strL,linearEqScaleTypes)},
                {"linear_equality_scales",14,0,9,0,0,0.,0.,0,N_mdm(RealDL,linearEqScales)},
                {"linear_equality_targets",14,0,7,0,0,0.,0.,0,N_mdm(RealDL,linearEqTargets)},
                {"linear_inequality_constraint_matrix",14,0,1,0,0,0.,0.,0,N_mdm(RealDL,
      linearIneqConstraintCoeffs)},
                {"linear_inequality_lower_bounds",14,0,2,0,0,0.,0.,0,N_mdm(RealDL,linearIneqLowerBnds)},
                {"linear_inequality_scale_types",15,0,4,0,0,0.,0.,0,N_mdm(strL,linearIneqScaleTypes)},
                {"linear_inequality_scales",14,0,5,0,0,0.,0.,0,N_mdm(RealDL,linearIneqScales)},
                {"linear_inequality_upper_bounds",14,0,3,0,0,0.,0.,0,N_mdm(RealDL,linearIneqUpperBnds)}
                }
```

KeyWord kw_38[7] [static]

Initial value:

```
= {
                {"merit1",8,0,1,1,0,0.,0.,0,N_mdm(lit,meritFunction_merit1)},
                {"merit1_smooth",8,0,1,1,0,0.,0.,0,N_mdm(lit,meritFunction_merit1_smooth)},
                {"merit2",8,0,1,1,0,0.,0.,0,N_mdm(lit,meritFunction_merit2)},
                {"merit2_smooth",8,0,1,1,0,0.,0.,0,N_mdm(lit,meritFunction_merit2_smooth)},
                {"merit2_squared",8,0,1,1,0,0.,0.,0,N_mdm(lit,meritFunction_merit2_squared)},
                {"merit_max",8,0,1,1,0,0.,0.,0,N_mdm(lit,meritFunction_merit_max)},
                {"merit_max_smooth",8,0,1,1,0,0.,0.,0,N_mdm(lit,meritFunction_merit_max_smooth)}
                }
```

KeyWord kw_39[2] [static]

Initial value:

```
= {
                {"blocking",8,0,1,1,0,0.,0.,0,N_mdm(lit,evalSynchronize_blocking)},
                {"nonblocking",8,0,1,1,0,0.,0.,0,N_mdm(lit,evalSynchronize_nonblocking)}
                }
```

KeyWord kw_40[11] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {"constraint_penalty",10,0,7,0,0,0.,0.,0,N_mdm(Real,constrPenalty)},
                {"contraction_factor",10,0,2,0,0,0.,0.,0,N_mdm(Real,contractStepLength)},
                {"initial_delta",10,0,1,0,0,0.,0.,0,N_mdm(Real,initStepLength)},
                {"merit_function",8,7,6,0,kw_38},
                {"smoothing_factor",10,0,8,0,0,0.,0.,0,N_mdm(Real,smoothFactor)},
                {"solution_accuracy",2,0,4,0,0,0.,0.,1,N_mdm(Real,solnTarget)},
                {"solution_target",10,0,4,0,0,0.,0.,0,N_mdm(Real,solnTarget)},
                {"synchronization",8,2,5,0,kw_39},
                {"threshold_delta",10,0,3,0,0,0.,0.,0,N_mdm(Real,threshStepLength)}
                }
```

KeyWord kw_41[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_42[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_43[5] [static]

Initial value:

```
= {
                {"dakota",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_GP_EMULATOR)},
                {"emulator_samples",9,0,2,0,0,0.,0.,0,N_mdm(int,emulatorSamples)},
                {"export_points_file",11,2,4,0,kw_41,0.,0.,0,N_mdm(str,approxExportFile)},
                {"import_points_file",11,3,3,0,kw_42,0.,0.,0,N_mdm(str,approxImportFile)},
                {"surfpack",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_KRIGING_EMULATOR)}
                }
```

KeyWord kw_44[1] [static]

Initial value:

```
= {
                {"sparse_grid_level",13,0,1,0,0,0.,0.,0,N_mdm(usharray,sparseGridLevel)}
                }
```

KeyWord kw_45[1] [static]

Initial value:

```
= {
                {"sparse_grid_level",13,0,1,0,0,0.,0.,0,N_mdm(usharray,sparseGridLevel)}
                }
```

KeyWord kw_46[4] [static]

Initial value:

```
= {
                {"gaussian_process",8,5,1,1,kw_43},
                {"kriging",0,5,1,1,kw_43,0.,0.,-1},
                {"pce",8,1,1,1,kw_44,0.,0.,0,N_mdm(type,emulatorType_PCE_EMULATOR)},
                {"sc",8,1,1,1,kw_45,0.,0.,0,N_mdm(type,emulatorType_SC_EMULATOR)}
                }
```

KeyWord kw_47[6] [static]

Initial value:

```
= {
                {"chains",0x29,0,1,0,0,3.,0.,0,N_mdm(int,numChains)},
                {"crossover_chain_pairs",0x29,0,3,0,0,0.,0.,0,N_mdm(int,crossoverChainPairs)},
                {"emulator",8,4,6,0,kw_46},
                {"gr_threshold",0x1a,0,4,0,0,0.,0.,0,N_mdm(Real,grThreshold)},
                {"jump_step",0x29,0,5,0,0,0.,0.,0,N_mdm(int,jumpStep)},
                {"num_cr",0x29,0,2,0,0,1.,0.,0,N_mdm(int,numCR)}
                }
```

KeyWord kw_48[2] [static]

Initial value:

```
= {
                {"adaptive",8,0,1,1,0,0.,0.,0,N_mdm(lit,metropolisType_adaptive)},
                {"hastings",8,0,1,1,0,0.,0.,0,N_mdm(lit,metropolisType_hastings)}
                }
```

KeyWord kw_49[2] [static]

Initial value:

```
= {
                {"delayed",8,0,1,1,0,0.,0.,0,N_mdm(lit,rejectionType_delayed)},
                {"standard",8,0,1,1,0,0.,0.,0,N_mdm(lit,rejectionType_standard)}
                }
```

KeyWord kw_50[2] [static]

Initial value:

```
= {
                {"metropolis",8,2,2,0,kw_48},
                {"rejection",8,2,1,0,kw_49}
                }
```

KeyWord kw_51[2] [static]

Initial value:

```
= {
                {"dram",8,2,1,1,kw_50,0.,0.,0,N_mdm(lit,mcmcType_dram)},
                {"multilevel",8,0,1,1,0,0.,0.,0,N_mdm(lit,mcmcType_multilevel)}
                }
```

KeyWord kw_52[2] [static]

Initial value:

```
= {
                {"mt19937",8,0,1,1,0,0.,0.,0,N_mdm(lit,rngName_mt19937)},
                {"rnum2",8,0,1,1,0,0.,0.,0,N_mdm(lit,rngName_rnum2)}
                }
```

KeyWord kw_53[3] [static]

Initial value:

```
= {
                {"mcmc_type",8,2,1,0,kw_51},
                {"proposal_covariance_scale",14,0,3,0,0,0.,0.,0,N_mdm(RealDL,proposalCovScale)},
                {"rng",8,2,2,0,kw_52}
                }
```

KeyWord kw_54[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_55[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_56[4] [static]

Initial value:

```
= {
                {0,0,3,0,0,kw_53},
                {"emulator_samples",9,0,1,1,0,0.,0.,0,N_mdm(int,emulatorSamples)},
                {"export_points_file",11,2,3,0,kw_54,0.,0.,0,N_mdm(str,approxExportFile)},
                {"import_points_file",11,3,2,0,kw_55,0.,0.,0,N_mdm(str,approxImportFile)}
                }
```

KeyWord kw_57[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_58[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_59[5] [static]

Initial value:

```
= {
                {"dakota",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_GP_EMULATOR)},
                {"emulator_samples",9,0,2,0,0,0.,0.,0,N_mdm(int,emulatorSamples)},
                {"export_points_file",11,2,4,0,kw_57,0.,0.,0,N_mdm(str,approxExportFile)},
                {"import_points_file",11,3,3,0,kw_58,0.,0.,0,N_mdm(str,approxImportFile)},
                {"surfpack",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_KRIGING_EMULATOR)}
                }
```

KeyWord kw_60[1] [static]

Initial value:

```
= {
                {"sparse_grid_level",13,0,1,0,0,0.,0.,0,N_mdm(usharray,sparseGridLevel)}
                }
```

KeyWord kw_61[1] [static]

Initial value:

```
= {
                {"sparse_grid_level",13,0,1,0,0,0.,0.,0,N_mdm(usharray,sparseGridLevel)}
                }
```

KeyWord kw_62[4] [static]

Initial value:

```
= {
                {"gaussian_process",8,5,1,1,kw_59},
                {"kriging",0,5,1,1,kw_59,0.,0.,-1},
                {"pce",8,1,1,1,kw_60,0.,0.,0,N_mdm(type,emulatorType_PCE_EMULATOR)},
                {"sc",8,1,1,1,kw_61,0.,0.,0,N_mdm(type,emulatorType_SC_EMULATOR)}
                }
```

KeyWord kw_63[2] [static]

Initial value:

```
= {
                {0,0,3,0,0,kw_53},
                {"emulator",8,4,1,0,kw_62}
                }
```

KeyWord kw_64[8] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {"calibrate_sigma",8,0,4,0,0,0.,0.,0,N_mdm(true,calibrateSigmaFlag)},
                {"dream",8,6,1,1,kw_47,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_DREAM)},
                {"gpmsa",8,3,1,1,kw_56,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_GPMSA)},
                {"likelihood_scale",10,0,3,0,0,0.,0.,0,N_mdm(Real,likelihoodScale)},
                {"queso",8,1,1,1,kw_63,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_QUESO)},
                {"use_derivatives",8,0,2,0,0,0.,0.,0,N_mdm(true,methodUseDerivsFlag)}
                }
```

KeyWord kw_65[4] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"deltas_per_variable",5,0,2,2,0,0.,0.,2,N_mdm(ivec,stepsPerVariable)},
                {"step_vector",14,0,1,1,0,0.,0.,0,N_mdm(RealDL,stepVector)},
                {"steps_per_variable",13,0,2,2,0,0.,0.,0,N_mdm(ivec,stepsPerVariable)}
                }
```

KeyWord kw_66[5] [static]

Initial value:

```
= {
                {"misc_options",15,0,4,0,0,0.,0.,0,N_mdm(strL,miscOptions)},
                {"seed",0x19,0,2,0,0,0.,0.,0,N_mdm(pint,randomSeed)},
                {"show_misc_options",8,0,3,0,0,0.,0.,0,N_mdm(true,showMiscOptions)},
                {"solution_accuracy",2,0,1,0,0,0.,0.,1,N_mdm(Real,solnTarget)},
                {"solution_target",10,0,1,0,0,0.,0.,0,N_mdm(Real,solnTarget)}
                }
```

KeyWord kw_67[3] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,5,0,0,kw_66},
                {"beta_solver_name",11,0,1,1,0,0.,0.,0,N_mdm(str,betaSolverName)}
                }
```

KeyWord kw_68[2] [static]

Initial value:

```
= {
                {"initial_delta",10,0,1,0,0,0.,0.,0,N_mdm(Real,initDelta)},
                {"threshold_delta",10,0,2,0,0,0.,0.,0,N_mdm(Real,threshDelta)}
                }
```

KeyWord kw_69[4] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,5,0,0,kw_66},
                {0,0,2,0,0,kw_68},
                {""}
                }
```

KeyWord kw_70[2] [static]

Initial value:

```
= {
                {"all_dimensions",8,0,1,1,0,0.,0.,0,N_mdm(lit,boxDivision_all_dimensions)},
                {"major_dimension",8,0,1,1,0,0.,0.,0,N_mdm(lit,boxDivision_major_dimension)}
                }
```

KeyWord kw_71[8] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,5,0,0,kw_66},
                {"constraint_penalty",10,0,6,0,0,0.,0.,0,N_mdm(Real,constraintPenalty)},
                {"division",8,2,1,0,kw_70},
                {"global_balance_parameter",10,0,2,0,0,0.,0.,0,N_mdm(Real,globalBalanceParam)},
                {"local_balance_parameter",10,0,3,0,0,0.,0.,0,N_mdm(Real,localBalanceParam)},
                {"max_boxsize_limit",10,0,4,0,0,0.,0.,0,N_mdm(Real,maxBoxSize)},
                {"min_boxsize_limit",10,0,5,0,0,0.,0.,0,N_mdm(Real,minBoxSize)}
                }
```

KeyWord kw_72[3] [static]

Initial value:

```
= {
                {"blend",8,0,1,1,0,0.,0.,0,N_mdm(lit,crossoverType_blend)},
                {"two_point",8,0,1,1,0,0.,0.,0,N_mdm(lit,crossoverType_two_point)},
                {"uniform",8,0,1,1,0,0.,0.,0,N_mdm(lit,crossoverType_uniform)}
                }
```

KeyWord kw_73[2] [static]

Initial value:

```
= {
                {"linear_rank",8,0,1,1,0,0.,0.,0,N_mdm(lit,fitnessType_linear_rank)},
                {"merit_function",8,0,1,1,0,0.,0.,0,N_mdm(lit,fitnessType_proportional)}
                }
```

KeyWord kw_74[3] [static]

Initial value:

```
= {
                {"flat_file",11,0,1,1,0,0.,0.,0,N_mdm(slit2,TYPE_DATA_initializationType_flat_file)},
                {"simple_random",8,0,1,1,0,0.,0.,0,N_mdm(lit,initializationType_random)},
                {"unique_random",8,0,1,1,0,0.,0.,0,N_mdm(lit,initializationType_unique_random)}
                }
```

KeyWord kw_75[2] [static]

Initial value:

```
= {
                {"mutation_range",9,0,2,0,0,0.,0.,0,N_mdm(int,mutationRange)},
                {"mutation_scale",10,0,1,0,0,0.,0.,0,N_mdm(Real,mutationScale)}
                }
```

KeyWord kw_76[5] [static]

Initial value:

```
= {
                {"non_adaptive",8,0,2,0,0,0.,0.,0,N_mdm(false,mutationAdaptive)},
                {"offset_cauchy",8,2,1,1,kw_75,0.,0.,0,N_mdm(lit,mutationType_offset_cauchy)},
                {"offset_normal",8,2,1,1,kw_75,0.,0.,0,N_mdm(lit,mutationType_offset_normal)},
                {"offset_uniform",8,2,1,1,kw_75,0.,0.,0,N_mdm(lit,mutationType_offset_uniform)},
                {"replace_uniform",8,0,1,1,0,0.,0.,0,N_mdm(lit,mutationType_replace_uniform)}
                }
```

KeyWord kw_77[4] [static]

Initial value:

```
= {
                {"chc",9,0,1,1,0,0.,0.,0,N_mdm(ilit2,TYPE_DATA_replacementType_chc)},
                {"elitist",9,0,1,1,0,0.,0.,0,N_mdm(ilit2,TYPE_DATA_replacementType_elitist)},
                {"new_solutions_generated",9,0,2,0,0,0.,0.,0,N_mdm(int,newSolnsGenerated)},
                {"random",9,0,1,1,0,0.,0.,0,N_mdm(ilit2,TYPE_DATA_replacementType_random)}
                }
```

KeyWord kw_78[11] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,5,0,0,kw_66},
                {"constraint_penalty",10,0,9,0,0,0.,0.,0,N_mdm(Real,constraintPenalty)},
                {"crossover_rate",10,0,5,0,0,0.,0.,0,N_mdm(Real,crossoverRate)},
                {"crossover_type",8,3,6,0,kw_72},
                {"fitness_type",8,2,3,0,kw_73},
                {"initialization_type",8,3,2,0,kw_74},
                {"mutation_rate",10,0,7,0,0,0.,0.,0,N_mdm(Real,mutationRate)},
                {"mutation_type",8,5,8,0,kw_76},
                {"population_size",0x19,0,1,0,0,0.,0.,0,N_mdm(pint,populationSize)},
                {"replacement_type",8,4,4,0,kw_77}
                }
```

KeyWord kw_79[2] [static]

Initial value:

```
= {
                {"constraint_penalty",10,0,2,0,0,0.,0.,0,N_mdm(Real,constraintPenalty)},
                {"contraction_factor",10,0,1,0,0,0.,0.,0,N_mdm(Real,contractFactor)}
                }
```

KeyWord kw_80[3] [static]

Initial value:

```
= {
                {"adaptive_pattern",8,0,1,1,0,0.,0.,0,N_mdm(lit,exploratoryMoves_adaptive)},
                {"basic_pattern",8,0,1,1,0,0.,0.,0,N_mdm(lit,exploratoryMoves_simple)},
                {"multi_step",8,0,1,1,0,0.,0.,0,N_mdm(lit,exploratoryMoves_multi_step)}
                }
```

KeyWord kw_81[2] [static]

Initial value:

```
= {
                {"coordinate",8,0,1,1,0,0.,0.,0,N_mdm(lit,patternBasis_coordinate)},
                {"simplex",8,0,1,1,0,0.,0.,0,N_mdm(lit,patternBasis_simplex)}
                }
```

KeyWord kw_82[2] [static]

Initial value:

```
= {
                {"blocking",8,0,1,1,0,0.,0.,0,N_mdm(lit,evalSynchronize_blocking)},
                {"nonblocking",8,0,1,1,0,0.,0.,0,N_mdm(lit,evalSynchronize_nonblocking)}
                }
```

KeyWord kw_83[12] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,5,0,0,kw_66},
                {0,0,2,0,0,kw_68},
                {0,0,2,0,0,kw_79},
                {"constant_penalty",8,0,1,0,0,0.,0.,0,N_mdm(true,constantPenalty)},
                {"expand_after_success",9,0,3,0,0,0.,0.,0,N_mdm(int,expandAfterSuccess)},
                {"exploratory_moves",8,3,7,0,kw_80},
                {"no_expansion",8,0,2,0,0,0.,0.,0,N_mdm(false,expansionFlag)},
                {"pattern_basis",8,2,4,0,kw_81},
                {"stochastic",8,0,5,0,0,0.,0.,0,N_mdm(true,randomizeOrderFlag)},
                {"synchronization",8,2,8,0,kw_82},
                {"total_pattern_size",9,0,6,0,0,0.,0.,0,N_mdm(int,totalPatternSize)}
                }
```

KeyWord kw_84[8] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,5,0,0,kw_66},
                {0,0,2,0,0,kw_68},
                {0,0,2,0,0,kw_79},
                {"constant_penalty",8,0,4,0,0,0.,0.,0,N_mdm(true,constantPenalty)},
                {"contract_after_failure",9,0,1,0,0,0.,0.,0,N_mdm(int,contractAfterFail)},
                {"expand_after_success",9,0,3,0,0,0.,0.,0,N_mdm(int,expandAfterSuccess)},
                {"no_expansion",8,0,2,0,0,0.,0.,0,N_mdm(false,expansionFlag)}
                }
```

KeyWord kw_85[4] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {"frcg",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_CONMIN_FRCG)},
                {"mfd",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_CONMIN_MFD)}
                }
```

KeyWord kw_86[3] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {""}
                }
```

KeyWord kw_87[1] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,1,0,0,0.,0.,0,N_mdm(Real,vbdDropTolerance)}
                }
```

KeyWord kw_88[14] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {"box_behnken",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_BOX_BEHNKEN)},
                {"central_composite",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_CENTRAL_COMPOSITE)},
                {"fixed_seed",8,0,5,0,0,0.,0.,0,N_mdm(true,fixedSeedFlag)},
                {"grid",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_GRID)},
                {"lhs",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_LHS)},
                {"main_effects",8,0,2,0,0,0.,0.,0,N_mdm(true,mainEffectsFlag)},
                {"oa_lhs",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_OA_LHS)},
                {"oas",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_OAS)},
                {"quality_metrics",8,0,3,0,0,0.,0.,0,N_mdm(true,volQualityFlag)},
                {"random",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_RANDOM)},
                {"symbols",9,0,6,0,0,0.,0.,0,N_mdm(int,numSymbols)},
                {"variance_based_decomp",8,1,4,0,kw_87,0.,0.,0,N_mdm(true,vbdFlag)}
                }
```

KeyWord kw_89[7] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {"bfgs",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_DOT_BFGS)},
                {"frcg",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_DOT_FRCG)},
                {"mmfd",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_DOT_MMFD)},
                {"slp",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_DOT_SLP)},
                {"sqp",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_DOT_SQP)}
                }
```

KeyWord kw_90[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_91[2] [static]

Initial value:

```
= {
                {"dakota",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_GP_EMULATOR)},
                {"surfpack",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_KRIGING_EMULATOR)}
                }
```

KeyWord kw_92[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_93[7] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"export_points_file",11,2,4,0,kw_90,0.,0.,0,N_mdm(str,approxExportFile)},
                {"gaussian_process",8,2,1,0,kw_91},
                {"import_points_file",11,3,3,0,kw_92,0.,0.,0,N_mdm(str,approxImportFile)},
                {"kriging",0,2,1,0,kw_91,0.,0.,-2},
                {"seed",0x19,0,5,0,0,0.,0.,0,N_mdm(pint,randomSeed)},
                {"use_derivatives",8,0,2,0,0,0.,0.,0,N_mdm(true,methodUseDerivsFlag)}
                }
```

KeyWord kw_94[5] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {0,0,4,0,0,kw_28},
                {"batch_size",9,0,2,0,0,0.,0.,0,N_mdm(int,batchSize)},
                {"emulator_samples",9,0,1,0,0,0.,0.,0,N_mdm(int,emulatorSamples)}
                }
```

KeyWord kw_95[3] [static]

Initial value:

```
= {
                {"grid",8,0,1,1,0,0.,0.,0,N_mdm(lit,trialType_grid)},
                {"halton",8,0,1,1,0,0.,0.,0,N_mdm(lit,trialType_halton)},
                {"random",8,0,1,1,0,0.,0.,0,N_mdm(lit,trialType_random)}
                }
```

KeyWord kw_96[1] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,1,0,0,0.,0.,0,N_mdm(Real,vbdDropTolerance)}
                }
```

KeyWord kw_97[8] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {"fixed_seed",8,0,4,0,0,0.,0.,0,N_mdm(true,fixedSeedFlag)},
                {"latinize",8,0,1,0,0,0.,0.,0,N_mdm(true,latinizeFlag)},
                {"num_trials",9,0,6,0,0,0.,0.,0,N_mdm(int,numTrials)},
                {"quality_metrics",8,0,2,0,0,0.,0.,0,N_mdm(true,volQualityFlag)},
                {"trial_type",8,3,5,0,kw_95},
                {"variance_based_decomp",8,1,3,0,kw_96,0.,0.,0,N_mdm(true,vbdFlag)}
                }
```

KeyWord kw_98[1] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,1,0,0,0.,0.,0,N_mdm(Real,vbdDropTolerance)}
                }
```

KeyWord kw_99[11] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"fixed_sequence",8,0,6,0,0,0.,0.,0,N_mdm(true,fixedSequenceFlag)},
                {"halton",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_FSU_HALTON)},
                {"hammersley",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_FSU_HAMMERSLEY)},
                {"latinize",8,0,2,0,0,0.,0.,0,N_mdm(true,latinizeFlag)},
                {"prime_base",13,0,9,0,0,0.,0.,0,N_mdm(ivec,primeBase)},
                {"quality_metrics",8,0,3,0,0,0.,0.,0,N_mdm(true,volQualityFlag)},
                {"samples",9,0,5,0,0,0.,0.,0,N_mdm(int,numSamples)},
                {"sequence_leap",13,0,8,0,0,0.,0.,0,N_mdm(ivec,sequenceLeap)},
                {"sequence_start",13,0,7,0,0,0.,0.,0,N_mdm(ivec,sequenceStart)},
                {"variance_based_decomp",8,1,4,0,kw_98,0.,0.,0,N_mdm(true,vbdFlag)}
                }
```

KeyWord kw_100[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_101[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_102[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
                {"series",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
                }
```

KeyWord kw_103[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
                {"probabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
                {"system",8,2,2,0,kw_102}
                }
```

KeyWord kw_104[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,kw_103},
                {"num_response_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,responseLevels)}
                }
```

KeyWord kw_105[7] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {0,0,4,0,0,kw_28},
                {"emulator_samples",9,0,1,0,0,0.,0.,0,N_mdm(int,emulatorSamples)},
                {"export_points_file",11,2,3,0,kw_100,0.,0.,0,N_mdm(str,approxExportFile)},
                {"import_points_file",11,3,2,0,kw_101,0.,0.,0,N_mdm(str,approxImportFile)},
                {"response_levels",14,2,4,0,kw_104,0.,0.,0,N_mdm(resplevs,responseLevels)}
                }
```

KeyWord kw_106[2] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"seed",0x19,0,1,0,0,0.,0.,0,N_mdm(pint,randomSeed)}
                }
```

KeyWord kw_107[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
                {"series",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
                }
```

KeyWord kw_108[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
                {"probabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
                {"system",8,2,2,0,kw_107}
                }
```

KeyWord kw_109[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,kw_108},
                {"num_response_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,responseLevels)}
                }
```

KeyWord kw_110[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_111[2] [static]

Initial value:

```
= {
                {"dakota",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_GP_EMULATOR)},
                {"surfpack",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_KRIGING_EMULATOR)}
                }
```

KeyWord kw_112[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_113[5] [static]

Initial value:

```
= {
                {"export_points_file",11,2,4,0,kw_110,0.,0.,0,N_mdm(str,approxExportFile)},
                {"gaussian_process",8,2,1,0,kw_111},
                {"import_points_file",11,3,3,0,kw_112,0.,0.,0,N_mdm(str,approxImportFile)},
                {"kriging",0,2,1,0,kw_111,0.,0.,-2},
                {"use_derivatives",8,0,2,0,0,0.,0.,0,N_mdm(true,methodUseDerivsFlag)}
                }
```

KeyWord kw_114[8] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {0,0,4,0,0,kw_28},
                {"ea",8,0,1,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_EA)},
                {"ego",8,5,1,0,kw_113,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_EGO)},
                {"lhs",8,0,1,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_LHS)},
                {"response_levels",14,2,2,0,kw_109,0.,0.,0,N_mdm(resplevs,responseLevels)},
                {"sbo",8,5,1,0,kw_113,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_SBO)}
                }
```

KeyWord kw_115[2] [static]

Initial value:

```
= {
                {"mt19937",8,0,1,1,0,0.,0.,0,N_mdm(lit,rngName_mt19937)},
                {"rnum2",8,0,1,1,0,0.,0.,0,N_mdm(lit,rngName_rnum2)}
                }
```

KeyWord kw_116[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_117[2] [static]

Initial value:

```
= {
                {"dakota",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_GP_EMULATOR)},
                {"surfpack",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_KRIGING_EMULATOR)}
                }
```

KeyWord kw_118[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_119[5] [static]

Initial value:

```
= {
                {"export_points_file",11,2,4,0,kw_116,0.,0.,0,N_mdm(str,approxExportFile)},
                {"gaussian_process",8,2,1,0,kw_117},
                {"import_points_file",11,3,3,0,kw_118,0.,0.,0,N_mdm(str,approxImportFile)},
                {"kriging",0,2,1,0,kw_117,0.,0.,-2},
                {"use_derivatives",8,0,2,0,0,0.,0.,0,N_mdm(true,methodUseDerivsFlag)}
                }
```

KeyWord kw_120[7] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {"ea",8,0,1,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_EA)},
                {"ego",8,5,1,0,kw_119,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_EGO)},
                {"lhs",8,0,1,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_LHS)},
                {"rng",8,2,2,0,kw_115},
                {"sbo",8,5,1,0,kw_119,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_SBO)}
                }
```

KeyWord kw_121[2] [static]

Initial value:

```
= {
                {"complementary",8,0,1,1,0,0.,0.,0,N_mdm(type,distributionType_COMPLEMENTARY)},
                {"cumulative",8,0,1,1,0,0.,0.,0,N_mdm(type,distributionType_CUMULATIVE)}
                }
```

KeyWord kw_122[1] [static]

Initial value:

```
= {
                {"num_gen_reliability_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,genReliabilityLevels)}
                }
```

KeyWord kw_123[1] [static]

Initial value:

```
= {
                {"num_probability_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,probabilityLevels)}
                }
```

KeyWord kw_124[3] [static]

Initial value:

```
= {
                {"distribution",8,2,1,0,kw_121},
                {"gen_reliability_levels",14,1,3,0,kw_122,0.,0.,0,N_mdm(resplevs,genReliabilityLevels)},
                {"probability_levels",14,1,2,0,kw_123,0.,0.,0,N_mdm(resplevs01,probabilityLevels)}
                }
```

KeyWord kw_125[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_126[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_127[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
                {"series",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
                }
```

KeyWord kw_128[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
                {"probabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
                {"system",8,2,2,0,kw_127}
                }
```

KeyWord kw_129[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,kw_128},
                {"num_response_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,responseLevels)}
                }
```

KeyWord kw_130[2] [static]

Initial value:

```
= {
                {"mt19937",8,0,1,1,0,0.,0.,0,N_mdm(lit,rngName_mt19937)},
                {"rnum2",8,0,1,1,0,0.,0.,0,N_mdm(lit,rngName_rnum2)}
                }
```

KeyWord kw_131[14] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,3,0,0,kw_124},
                {"dakota",8,0,2,0,0,0.,0.,0,N_mdm(type,emulatorType_GP_EMULATOR)},
                {"export_points_file",11,2,4,0,kw_125,0.,0.,0,N_mdm(str,approxExportFile)},
                {"import_points_file",11,3,3,0,kw_126,0.,0.,0,N_mdm(str,approxImportFile)},
                {"response_levels",14,2,8,0,kw_129,0.,0.,0,N_mdm(resplevs,responseLevels)},
                {"rng",8,2,7,0,kw_130},
                {"seed",0x19,0,6,0,0,0.,0.,0,N_mdm(pint,randomSeed)},
                {"surfpack",8,0,2,0,0,0.,0.,0,N_mdm(type,emulatorType_KRIGING_EMULATOR)},
                {"u_gaussian_process",8,0,1,1,0,0.,0.,0,N_mdm(utype,reliabilitySearchType_EGRA_U)},
                {"u_kriging",0,0,1,1,0,0.,0.,-1,N_mdm(utype,reliabilitySearchType_EGRA_U)},
                {"use_derivatives",8,0,5,0,0,0.,0.,0,N_mdm(true,methodUseDerivsFlag)},
                {"x_gaussian_process",8,0,1,1,0,0.,0.,0,N_mdm(utype,reliabilitySearchType_EGRA_X)},
                {"x_kriging",0,0,1,1,0,0.,0.,-1,N_mdm(utype,reliabilitySearchType_EGRA_X)}
                }
```

KeyWord kw_132[2] [static]

Initial value:

```
= {
                {"master",8,0,1,1,0,0.,0.,0,N_mdm(type,iteratorScheduling_MASTER_SCHEDULING)},
                {"peer",8,0,1,1,0,0.,0.,0,N_mdm(type,iteratorScheduling_PEER_SCHEDULING)}
                }
```

KeyWord kw_133[3] [static]

Initial value:

```
= {
                {"iterator_scheduling",8,2,2,0,kw_132},
                {"iterator_servers",0x19,0,1,0,0,0.,0.,0,N_mdm(pint,iteratorServers)},
                {"processors_per_iterator",0x19,0,3,0,0,0.,0.,0,N_mdm(pint,procsPerIterator)}
                }
```

KeyWord kw_134[1] [static]

Initial value:

```
= {
                {"model_pointer_list",11,0,1,0,0,0.,0.,0,N_mdm(strL,hybridModelPointers)}
                }
```

KeyWord kw_135[2] [static]

Initial value:

```
= {
                {"method_name_list",15,1,1,1,kw_134,0.,0.,0,N_mdm(strL,hybridMethodNames)},
                {"method_pointer_list",15,0,1,1,0,0.,0.,0,N_mdm(strL,hybridMethodPointers)}
                }
```

KeyWord kw_136[1] [static]

Initial value:

```
= {
                {"global_model_pointer",11,0,1,0,0,0.,0.,0,N_mdm(str,hybridGlobalModelPointer)}
                }
```

KeyWord kw_137[1] [static]

Initial value:

```
= {
                {"local_model_pointer",11,0,1,0,0,0.,0.,0,N_mdm(str,hybridLocalModelPointer)}
                }
```

KeyWord kw_138[5] [static]

Initial value:

```
= {
                {"global_method_name",11,1,1,1,kw_136,0.,0.,0,N_mdm(str,hybridGlobalMethodName)},
                {"global_method_pointer",11,0,1,1,0,0.,0.,0,N_mdm(str,hybridGlobalMethodPointer)},
                {"local_method_name",11,1,2,2,kw_137,0.,0.,0,N_mdm(str,hybridLocalMethodName)},
                {"local_method_pointer",11,0,2,2,0,0.,0.,0,N_mdm(str,hybridLocalMethodPointer)},
                {"local_search_probability",10,0,3,0,0,0.,0.,0,N_mdm(Real,hybridLSProb)}
                }
```

KeyWord kw_139[1] [static]

Initial value:

```
= {
                {"model_pointer_list",11,0,1,0,0,0.,0.,0,N_mdm(strL,hybridModelPointers)}
                }
```

KeyWord kw_140[2] [static]

Initial value:

```
= {
                {"method_name_list",15,1,1,1,kw_139,0.,0.,0,N_mdm(strL,hybridMethodNames)},
                {"method_pointer_list",15,0,1,1,0,0.,0.,0,N_mdm(strL,hybridMethodPointers)}
                }
```

KeyWord kw_141[6] [static]

Initial value:

```
= {
                {0,0,3,0,0,kw_133},
                {"collaborative",8,2,1,1,kw_135,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_COLLABORATIVE)},
                {"coupled",0,5,1,1,kw_138,0.,0.,1,N_mdm(utype,subMethod_SUBMETHOD_EMBEDDED)},
                {"embedded",8,5,1,1,kw_138,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_EMBEDDED)},
                {"sequential",8,2,1,1,kw_140,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_SEQUENTIAL)},
                {"uncoupled",0,2,1,1,kw_140,0.,0.,-1,N_mdm(utype,subMethod_SUBMETHOD_SEQUENTIAL)}
                }
```

KeyWord kw_142[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
                {"series",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
                }
```

KeyWord kw_143[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
                {"probabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
                {"system",8,2,2,0,kw_142}
                }
```

KeyWord kw_144[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,kw_143},
                {"num_response_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,responseLevels)}
                }
```

KeyWord kw_145[8] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {0,0,4,0,0,kw_28},
                {"adapt_import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_AIS)},
                {"import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_IS)},
                {"mm_adapt_import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_MMAIS)},
                {"refinement_samples",9,0,2,0,0,0.,0.,0,N_mdm(int,refineSamples)},
                {"response_levels",14,2,3,0,kw_144,0.,0.,0,N_mdm(resplevs,responseLevels)}
                }
```

KeyWord kw_146[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,pstudyFileActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,pstudyFileAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,pstudyFileAnnotated)}
                }
```

KeyWord kw_147[3] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"import_points_file",11,3,1,1,kw_146,0.,0.,0,N_mdm(str,pstudyFilename)},
                {"list_of_points",14,0,1,1,0,0.,0.,0,N_mdm(RealDL,listOfPoints)}
                }
```

KeyWord kw_148[2] [static]

Initial value:

```
= {
                {"complementary",8,0,1,1,0,0.,0.,0,N_mdm(type,distributionType_COMPLEMENTARY)},
                {"cumulative",8,0,1,1,0,0.,0.,0,N_mdm(type,distributionType_CUMULATIVE)}
                }
```

KeyWord kw_149[1] [static]

Initial value:

```
= {
                {"num_gen_reliability_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,genReliabilityLevels)}
                }
```

KeyWord kw_150[1] [static]

Initial value:

```
= {
                {"num_probability_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,probabilityLevels)}
                }
```

KeyWord kw_151[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
                {"series",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
                }
```

KeyWord kw_152[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
                {"probabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
                {"system",8,2,2,0,kw_151}
                }
```

KeyWord kw_153[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,kw_152},
                {"num_response_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,responseLevels)}
                }
```

KeyWord kw_154[7] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"distribution",8,2,5,0,kw_148},
                {"gen_reliability_levels",14,1,4,0,kw_149,0.,0.,0,N_mdm(resplevs,genReliabilityLevels)},
                {"nip",8,0,1,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_NIP)},
                {"probability_levels",14,1,3,0,kw_150,0.,0.,0,N_mdm(resplevs01,probabilityLevels)},
                {"response_levels",14,2,2,0,kw_153,0.,0.,0,N_mdm(resplevs,responseLevels)},
                {"sqp",8,0,1,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_SQP)}
                }
```

KeyWord kw_155[3] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"nip",8,0,1,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_NIP)},
                {"sqp",8,0,1,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_SQP)}
                }
```

KeyWord kw_156[5] [static]

Initial value:

```
= {
                {"adapt_import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_AIS)},
                {"import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_IS)},
                {"mm_adapt_import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_MMAIS)},
                {"refinement_samples",9,0,2,0,0,0.,0.,0,N_mdm(int,refineSamples)},
                {"seed",0x19,0,3,0,0,0.,0.,0,N_mdm(pint,randomSeed)}
                }
```

KeyWord kw_157[4] [static]

Initial value:

```
= {
                {"first_order",8,0,1,1,0,0.,0.,0,N_mdm(lit,reliabilityIntegration_first_order)},
                {"probability_refinement",8,5,2,0,kw_156},
                {"sample_refinement",0,5,2,0,kw_156,0.,0.,-1},
                {"second_order",8,0,1,1,0,0.,0.,0,N_mdm(lit,reliabilityIntegration_second_order)}
                }
```

KeyWord kw_158[10] [static]

Initial value:

```
= {
                {"integration",8,4,3,0,kw_157},
                {"nip",8,0,2,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_NIP)},
                {"no_approx",8,0,1,1,0,0.,0.,0,N_mdm(utype,reliabilitySearchType_NO_APPROX)},
                {"sqp",8,0,2,0,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_SQP)},
                {"u_taylor_mean",8,0,1,1,0,0.,0.,0,N_mdm(utype,reliabilitySearchType_AMV_U)},
                {"u_taylor_mpp",8,0,1,1,0,0.,0.,0,N_mdm(utype,reliabilitySearchType_AMV_PLUS_U)},
                {"u_two_point",8,0,1,1,0,0.,0.,0,N_mdm(utype,reliabilitySearchType_TANA_U)},
                {"x_taylor_mean",8,0,1,1,0,0.,0.,0,N_mdm(utype,reliabilitySearchType_AMV_X)},
                {"x_taylor_mpp",8,0,1,1,0,0.,0.,0,N_mdm(utype,reliabilitySearchType_AMV_PLUS_X)},
                {"x_two_point",8,0,1,1,0,0.,0.,0,N_mdm(utype,reliabilitySearchType_TANA_X)}
                }
```

KeyWord kw_159[1] [static]

Initial value:

```
= {
                {"num_reliability_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,reliabilityLevels)}
                }
```

KeyWord kw_160[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
                {"series",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
                }
```

KeyWord kw_161[4] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
                {"probabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
                {"reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_RELIABILITIES)},
                {"system",8,2,2,0,kw_160}
                }
```

KeyWord kw_162[2] [static]

Initial value:

```
= {
                {"compute",8,4,2,0,kw_161},
                {"num_response_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,responseLevels)}
                }
```

KeyWord kw_163[5] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,3,0,0,kw_124},
                {"mpp_search",8,10,1,0,kw_158},
                {"reliability_levels",14,1,3,0,kw_159,0.,0.,0,N_mdm(resplevs,reliabilityLevels)},
                {"response_levels",14,2,2,0,kw_162,0.,0.,0,N_mdm(resplevs,responseLevels)}
                }
```

KeyWord kw_164[8] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {"display_all_evaluations",8,0,6,0,0,0.,0.,0,N_mdm(true,showAllEval)},
                {"display_format",11,0,4,0,0,0.,0.,0,N_mdm(str,displayFormat)},
                {"function_precision",10,0,1,0,0,0.,0.,0,N_mdm(Real,functionPrecision)},
                {"history_file",11,0,3,0,0,0.,0.,0,N_mdm(str,historyFile)},
                {"seed",0x19,0,2,0,0,0.,0.,0,N_mdm(pint,randomSeed)},
                {"variable_neighborhood_search",10,0,5,0,0,0.,0.,0,N_mdm(Real,vns)}
                }
```

KeyWord kw_165[2] [static]

Initial value:

```
= {
                {"num_offspring",0x19,0,2,0,0,0.,0.,0,N_mdm(pintz,numOffspring)},
                {"num_parents",0x19,0,1,0,0,0.,0.,0,N_mdm(pintz,numParents)}
                }
```

KeyWord kw_166[5] [static]

Initial value:

```
= {
                {"crossover_rate",10,0,2,0,0,0.,0.,0,N_mdm(litz,TYPE_DATA_crossoverType_null_crossover)},
                {"multi_point_binary",9,0,1,1,0,0.,0.,0,N_mdm(ilit2p,
    TYPE_DATA_crossoverType_multi_point_binary)},
                {"multi_point_parameterized_binary",9,0,1,1,0,0.,0.,0,N_mdm(ilit2p,
    TYPE_DATA_crossoverType_multi_point_parameterized_binary)},
                {"multi_point_real",9,0,1,1,0,0.,0.,0,N_mdm(ilit2p,TYPE_DATA_crossoverType_multi_point_real
    )},
                {"shuffle_random",8,2,1,1,kw_165,0.,0.,0,N_mdm(litc,TYPE_DATA_crossoverType_shuffle_random)
    }
                }
```

KeyWord kw_167[3] [static]

Initial value:

```
= {
                {"flat_file",11,0,1,1,0,0.,0.,0,N_mdm(slit2,TYPE_DATA_initializationType_flat_file)},
                {"simple_random",8,0,1,1,0,0.,0.,0,N_mdm(lit,initializationType_random)},
                {"unique_random",8,0,1,1,0,0.,0.,0,N_mdm(lit,initializationType_unique_random)}
                }
```

KeyWord kw_168[1] [static]

Initial value:

```
= {
                {"mutation_scale",10,0,1,0,0,0.,0.,0,N_mdm(Real01,mutationScale)}
                }
```

KeyWord kw_169[6] [static]

Initial value:

```
= {
                {"bit_random",8,0,1,1,0,0.,0.,0,N_mdm(lit,mutationType_bit_random)},
                {"mutation_rate",10,0,2,0,0,0.,0.,0,N_mdm(litz,TYPE_DATA_mutationType_null_mutation)},
                {"offset_cauchy",8,1,1,1,kw_168,0.,0.,0,N_mdm(litc,TYPE_DATA_mutationType_offset_cauchy)},
                {"offset_normal",8,1,1,1,kw_168,0.,0.,0,N_mdm(litc,TYPE_DATA_mutationType_offset_normal)},
                {"offset_uniform",8,1,1,1,kw_168,0.,0.,0,N_mdm(litc,TYPE_DATA_mutationType_offset_uniform)}
    ,
                {"replace_uniform",8,0,1,1,0,0.,0.,0,N_mdm(lit,mutationType_replace_uniform)}
                }
```

KeyWord kw_170[7] [static]

Initial value:

```
= {
                {"crossover_type",8,5,5,0,kw_166},
                {"initialization_type",8,3,4,0,kw_167},
                {"log_file",11,0,2,0,0,0.,0.,0,N_mdm(str,logFile)},
                {"mutation_type",8,6,6,0,kw_169},
                {"population_size",0x29,0,1,0,0,0.,0.,0,N_mdm(nnint,populationSize)},
                {"print_each_pop",8,0,3,0,0,0.,0.,0,N_mdm(true,printPopFlag)},
                {"seed",0x19,0,7,0,0,0.,0.,0,N_mdm(pint,randomSeed)}
                }
```

KeyWord kw_171[3] [static]

Initial value:

```
= {
                {"metric_tracker",8,0,1,1,0,0.,0.,0,N_mdm(lit,convergenceType_metric_tracker)},
                {"num_generations",0x29,0,3,0,0,0.,0.,0,N_mdm(sizet,numGenerations)},
                {"percent_change",10,0,2,0,0,0.,0.,0,N_mdm(Realz,convergenceTolerance)}
                }
```

KeyWord kw_172[2] [static]

Initial value:

```
= {
                {"domination_count",8,0,1,1,0,0.,0.,0,N_mdm(lit,fitnessType_domination_count)},
                {"layer_rank",8,0,1,1,0,0.,0.,0,N_mdm(lit,fitnessType_layer_rank)}
                }
```

KeyWord kw_173[1] [static]

Initial value:

```
= {
                {"num_designs",0x29,0,1,0,0,2.,0.,0,N_mdm(pintz,numDesigns)}
                }
```

KeyWord kw_174[3] [static]

Initial value:

```
= {
                {"distance",14,0,1,1,0,0.,0.,0,N_mdm(RealLlit,TYPE_DATA_nichingType_distance)},
                {"max_designs",14,1,1,1,kw_173,0.,0.,0,N_mdm(RealLlit,TYPE_DATA_nichingType_max_designs)},
                {"radial",14,0,1,1,0,0.,0.,0,N_mdm(RealLlit,TYPE_DATA_nichingType_radial)}
                }
```

KeyWord kw_175[1] [static]

Initial value:

```
= {
                {"orthogonal_distance",14,0,1,1,0,0.,0.,0,N_mdm(RealLlit,
      TYPE_DATA_postProcessorType_distance_postprocessor)}
                }
```

KeyWord kw_176[2] [static]

Initial value:

```
= {
                {"shrinkage_fraction",10,0,1,0,0,0.,0.,0,N_mdm(Real01,shrinkagePercent)},
                {"shrinkage_percentage",2,0,1,0,0,0.,0.,-1,N_mdm(Real01,shrinkagePercent)}
                }
```

KeyWord kw_177[4] [static]

Initial value:

```
= {
                {"below_limit",10,2,1,1,kw_176,0.,0.,0,N_mdm(litp,TYPE_DATA_replacementType_below_limit)},
                {"elitist",8,0,1,1,0,0.,0.,0,N_mdm(lit,replacementType_elitist)},
                {"roulette_wheel",8,0,1,1,0,0.,0.,0,N_mdm(lit,replacementType_roulette_wheel)},
                {"unique_roulette_wheel",8,0,1,1,0,0.,0.,0,N_mdm(lit,replacementType_unique_roulette_wheel)
      }
                }
```

KeyWord kw_178[8] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {0,0,7,0,0,kw_170},
                {"convergence_type",8,3,4,0,kw_171},
                {"fitness_type",8,2,1,0,kw_172},
                {"niching_type",8,3,3,0,kw_174},
                {"postprocessor_type",8,1,5,0,kw_175},
                {"replacement_type",8,4,2,0,kw_177}
                }
```

KeyWord kw_179[1] [static]

Initial value:

```
= {
                {"model_pointer",11,0,1,0,0,0.,0.,0,N_mdm(str,subModelPointer)}
                }
```

KeyWord kw_180[1] [static]

Initial value:

```
= {
                {"seed",9,0,1,0,0,0.,0.,0,N_mdm(int,randomSeed)}
                }
```

KeyWord kw_181[5] [static]

Initial value:

```
= {
                {0,0,3,0,0,kw_133},
                {"method_name",11,1,1,1,kw_179,0.,0.,0,N_mdm(str,subMethodName)},
                {"method_pointer",11,0,1,1,0,0.,0.,0,N_mdm(str,subMethodPointer)},
                {"random_starts",9,1,2,0,kw_180,0.,0.,0,N_mdm(int,concurrentRandomJobs)},
                {"starting_points",14,0,3,0,0,0.,0.,0,N_mdm(RealDL,concurrentParameterSets)}
                }
```

KeyWord kw_182[2] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"partitions",13,0,1,1,0,0.,0.,0,N_mdm(usharray,varPartitions)}
                }
```

KeyWord kw_183[5] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"min_boxsize_limit",10,0,2,0,0,0.,0.,0,N_mdm(Real,minBoxSize)},
                {"solution_accuracy",2,0,1,0,0,0.,0.,1,N_mdm(Real,solnTarget)},
                {"solution_target",10,0,1,0,0,0.,0.,0,N_mdm(Real,solnTarget)},
                {"volume_boxsize_limit",10,0,3,0,0,0.,0.,0,N_mdm(Real,volBoxSize)}
                }
```

KeyWord kw_184[10] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"absolute_conv_tol",10,0,2,0,0,0.,0.,0,N_mdm(Real,absConvTol)},
                {"covariance",9,0,8,0,0,0.,0.,0,N_mdm(int,covarianceType)},
                {"false_conv_tol",10,0,6,0,0,0.,0.,0,N_mdm(Real,falseConvTol)},
                {"function_precision",10,0,1,0,0,0.,0.,0,N_mdm(Real,functionPrecision)},
                {"initial_trust_radius",10,0,7,0,0,0.,0.,0,N_mdm(Real,initTRRadius)},
                {"regression_diagnostics",8,0,9,0,0,0.,0.,0,N_mdm(true,regressDiag)},
                {"singular_conv_tol",10,0,4,0,0,0.,0.,0,N_mdm(Real,singConvTol)},
                {"singular_radius",10,0,5,0,0,0.,0.,0,N_mdm(Real,singRadius)},
                {"x_conv_tol",10,0,3,0,0,0.,0.,0,N_mdm(Real,xConvTol)}
                }
```

KeyWord kw_185[1] [static]

Initial value:

```
= {
                {"surrogate_order",9,0,1,0,0,0.,0.,0,N_mdm(int,emulatorOrder)}
                }
```

KeyWord kw_186[2] [static]

Initial value:

```
= {
                {"gaussian_process",8,0,1,1,0,0.,0.,0,N_mdm(type,emulatorType_KRIGING_EMULATOR)},
                {"voronoi_surrogate",8,1,1,1,kw_185,0.,0.,0,N_mdm(type,emulatorType_VPS_EMULATOR)}
                }
```

KeyWord kw_187[2] [static]

Initial value:

```
= {
                {"global",8,0,1,1,0,0.,0.,0,N_mdm(lit,lipschitzType_global)},
                {"local",8,0,1,1,0,0.,0.,0,N_mdm(lit,lipschitzType_local)}
                }
```

KeyWord kw_188[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
                {"series",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
                }
```

KeyWord kw_189[3] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
                {"probabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
                {"system",8,2,2,0,kw_188}
                }
```

KeyWord kw_190[2] [static]

Initial value:

```
= {
                {"compute",8,3,2,0,kw_189},
                {"num_response_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,responseLevels)}
                }
```

KeyWord kw_191[7] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {0,0,4,0,0,kw_28},
                {"emulator",8,2,2,0,kw_186},
                {"emulator_samples",9,0,3,0,0,0.,0.,0,N_mdm(int,emulatorSamples)},
                {"lipschitz",8,2,1,0,kw_187},
                {"response_levels",14,2,4,0,kw_190,0.,0.,0,N_mdm(resplevs,responseLevels)}
                }
```

KeyWord kw_192[1] [static]

Initial value:

```
= {
                {"num_reliability_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,reliabilityLevels)}
                }
```

KeyWord kw_193[2] [static]

Initial value:

```
= {
                {"parallel",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
                {"series",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
                }
```

KeyWord kw_194[4] [static]

Initial value:

```
= {
                {"gen_reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
                {"probabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
                {"reliabilities",8,0,1,1,0,0.,0.,0,N_mdm(type,responseLevelTarget_RELIABILITIES)},
                {"system",8,2,2,0,kw_193}
                }
```

KeyWord kw_195[2] [static]

Initial value:

```
= {
                {"compute",8,4,2,0,kw_194},
                {"num_response_levels",13,0,1,0,0,0.,0.,0,N_mdm(num_resplevs,responseLevels)}
                }
```

KeyWord kw_196[2] [static]

Initial value:

```
= {
                {"reliability_levels",14,1,1,0,kw_192,0.,0.,0,N_mdm(resplevs,reliabilityLevels)},
                {"response_levels",14,2,2,0,kw_195,0.,0.,0,N_mdm(resplevs,responseLevels)}
                }
```

KeyWord kw_197[1] [static]

Initial value:

```
= {
                {"fixed_seed",8,0,1,0,0,0.,0.,0,N_mdm(true,fixedSeedFlag)}
                }
```

KeyWord kw_198[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxImportAnnotated)}
                }
```

KeyWord kw_199[1] [static]

Initial value:

```
= {
                {"import_points_file",11,3,1,0,kw_198,0.,0.,0,N_mdm(str,approxImportFile)}
                }
```

KeyWord kw_200[2] [static]

Initial value:

```
= {
                {"advancements",9,0,1,0,0,0.,0.,0,N_mdm(ushint,adaptedBasisAdvancements)},
                {"soft_convergence_limit",9,0,2,0,0,0.,0.,0,N_mdm(ushint,softConvLimit)}
                }
```

KeyWord kw_201[3] [static]

Initial value:

```
= {
                {"adapted",8,2,1,1,kw_200,0.,0.,0,N_mdm(type,
      expansionBasisType_ADAPTED_BASIS_EXPANDING_FRONT)},
                {"tensor_product",8,0,1,1,0,0.,0.,0,N_mdm(type,expansionBasisType_TENSOR_PRODUCT_BASIS)},
                {"total_order",8,0,1,1,0,0.,0.,0,N_mdm(type,expansionBasisType_TOTAL_ORDER_BASIS)}
                }
```

KeyWord kw_202[1] [static]

Initial value:

```
= {
                {"noise_tolerance",14,0,1,0,0,0.,0.,0,N_mdm(RealDL,regressionNoiseTol)}
                }
```

KeyWord kw_203[1] [static]

Initial value:

```
= {
                {"noise_tolerance",14,0,1,0,0,0.,0.,0,N_mdm(RealDL,regressionNoiseTol)}
                }
```

KeyWord kw_204[2] [static]

Initial value:

```
= {
                {"l2_penalty",10,0,2,0,0,0.,0.,0,N_mdm(Real,regressionL2Penalty)},
                {"noise_tolerance",14,0,1,0,0,0.,0.,0,N_mdm(RealDL,regressionNoiseTol)}
                }
```

KeyWord kw_205[2] [static]

Initial value:

```
= {
                {"equality_constrained",8,0,1,0,0,0.,0.,0,N_mdm(type,lsRegressionType_EQ_CON_LS)},
                {"svd",8,0,1,0,0,0.,0.,0,N_mdm(type,lsRegressionType_SVD_LS)}
                }
```

KeyWord kw_206[1] [static]

Initial value:

```
= {
                {"noise_tolerance",14,0,1,0,0,0.,0.,0,N_mdm(RealDL,regressionNoiseTol)}
                }
```

KeyWord kw_207[17] [static]

Initial value:

```
= {
                {"basis_pursuit",8,0,2,0,0,0.,0.,0,N_mdm(type,regressionType_BASIS_PURSUIT)},
                {"basis_pursuit_denoising",8,1,2,0,kw_202,0.,0.,0,N_mdm(type,
      regressionType_BASIS_PURSUIT_DENOISING)},
                {"bp",0,0,2,0,0,0.,0.,-2,N_mdm(type,regressionType_BASIS_PURSUIT)},
                {"bpdn",0,1,2,0,kw_202,0.,0.,-2,N_mdm(type,regressionType_BASIS_PURSUIT_DENOISING)},
                {"cross_validation",8,0,3,0,0,0.,0.,0,N_mdm(true,crossValidation)},
                {"lars",0,1,2,0,kw_203,0.,0.,3,N_mdm(type,regressionType_LEAST_ANGLE_REGRESSION)},
                {"lasso",0,2,2,0,kw_204,0.,0.,1,N_mdm(type,regressionType_LASSO_REGRESSION)},
                {"least_absolute_shrinkage",8,2,2,0,kw_204,0.,0.,0,N_mdm(type,
      regressionType_LASSO_REGRESSION)},
                {"least_angle_regression",8,1,2,0,kw_203,0.,0.,0,N_mdm(type,
      regressionType_LEAST_ANGLE_REGRESSION)},
                {"least_squares",8,2,2,0,kw_205,0.,0.,0,N_mdm(type,
      regressionType_DEFAULT_LEAST_SQ_REGRESSION)},
                {"omp",0,1,2,0,kw_206,0.,0.,1,N_mdm(type,regressionType_ORTHOG_MATCH_PURSUIT)},
                {"orthogonal_matching_pursuit",8,1,2,0,kw_206,0.,0.,0,N_mdm(type,
      regressionType_ORTHOG_MATCH_PURSUIT)},
                {"ratio_order",10,0,1,0,0,0.,0.,0,N_mdm(Realp,collocRatioTermsOrder)},
                {"reuse_points",8,0,6,0,0,0.,0.,0,N_mdm(lit,pointReuse_all)},
                {"reuse_samples",0,0,6,0,0,0.,0.,-1,N_mdm(lit,pointReuse_all)},
                {"tensor_grid",8,0,5,0,0,0.,0.,0,N_mdm(true,tensorGridFlag)},
                {"use_derivatives",8,0,4,0,0,0.,0.,0,N_mdm(true,methodUseDerivsFlag)}
                }
```

KeyWord kw_208[3] [static]

Initial value:

```
= {
                {"incremental_lhs",8,0,2,0,0,0.,0.,0,N_mdm(lit,expansionSampleType_incremental_lhs)},
                {"reuse_points",8,0,1,0,0,0.,0.,0,N_mdm(lit,pointReuse_all)},
                {"reuse_samples",0,0,1,0,0,0.,0.,-1,N_mdm(lit,pointReuse_all)}
                }
```

KeyWord kw_209[6] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_199},
                {"basis_type",8,3,2,0,kw_201},
                {"collocation_points",13,17,3,1,kw_207,0.,0.,0,N_mdm(szarray,collocationPoints)},
                {"collocation_ratio",10,17,3,1,kw_207,0.,0.,0,N_mdm(Realp,collocationRatio)},
                {"dimension_preference",14,0,1,0,0,0.,0.,0,N_mdm(RealDL,anisoDimPref)},
                {"expansion_samples",13,3,3,1,kw_208,0.,0.,0,N_mdm(szarray,expansionSamples)}
                }
```

KeyWord kw_210[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_211[6] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_199},
                {"collocation_points",13,0,1,1,0,0.,0.,0,N_mdm(szarray,collocationPoints)},
                {"cross_validation",8,0,2,0,0,0.,0.,0,N_mdm(true,crossValidation)},
                {"reuse_points",8,0,4,0,0,0.,0.,0,N_mdm(lit,pointReuse_all)},
                {"reuse_samples",0,0,4,0,0,0.,0.,-1,N_mdm(lit,pointReuse_all)},
                {"tensor_grid",13,0,3,0,0,0.,0.,0,N_mdm(usharray,tensorGridOrder)}
                }
```

KeyWord kw_212[3] [static]

Initial value:

```
= {
                {"decay",8,0,1,1,0,0.,0.,0,N_mdm(type,refinementControl_DIMENSION_ADAPTIVE_CONTROL_DECAY)},
                {"generalized",8,0,1,1,0,0.,0.,0,N_mdm(type,
     refinementControl_DIMENSION_ADAPTIVE_CONTROL_GENERALIZED)},
                {"sobol",8,0,1,1,0,0.,0.,0,N_mdm(type,refinementControl_DIMENSION_ADAPTIVE_CONTROL_SOBOL)}
                }
```

KeyWord kw_213[2] [static]

Initial value:

```
= {
                {"dimension_adaptive",8,3,1,1,kw_212},
                {"uniform",8,0,1,1,0,0.,0.,0,N_mdm(type,refinementControl_UNIFORM_CONTROL)}
                }
```

KeyWord kw_214[4] [static]

Initial value:

```
= {
                {"adapt_import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_AIS)},
                {"import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_IS)},
                {"mm_adapt_import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_MMAIS)},
                {"refinement_samples",9,0,2,0,0,0.,0.,0,N_mdm(int,refineSamples)}
                }
```

KeyWord kw_215[3] [static]

Initial value:

```
= {
                {"dimension_preference",14,0,1,0,0,0.,0.,0,N_mdm(RealDL,anisoDimPref)},
                {"nested",8,0,2,0,0,0.,0.,0,N_mdm(type,nestingOverride_NESTED)},
                {"non_nested",8,0,2,0,0,0.,0.,0,N_mdm(type,nestingOverride_NON_NESTED)}
                }
```

KeyWord kw_216[2] [static]

Initial value:

```
= {
                {"lhs",8,0,1,1,0,0.,0.,0,N_mdm(utype,sampleType_SUBMETHOD_LHS)},
                {"random",8,0,1,1,0,0.,0.,0,N_mdm(utype,sampleType_SUBMETHOD_RANDOM)}
                }
```

KeyWord kw_217[3] [static]

Initial value:

```
= {
                {0,0,3,0,0,kw_215},
                {"restricted",8,0,1,0,0,0.,0.,0,N_mdm(type,growthOverride_RESTRICTED)},
                {"unrestricted",8,0,1,0,0,0.,0.,0,N_mdm(type,growthOverride_UNRESTRICTED)}
                }
```

KeyWord kw_218[2] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,2,0,0,0.,0.,0,N_mdm(Real,vbdDropTolerance)},
                {"interaction_order",0x19,0,1,0,0,0.,0.,0,N_mdm(ushint,vbdOrder)}
                }
```

KeyWord kw_219[25] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {0,0,4,0,0,kw_28},
                {0,0,2,0,0,kw_196},
                {0,0,1,0,0,kw_197},
                {"askey",8,0,2,0,0,0.,0.,0,N_mdm(type,expansionType_ASKEY_U)},
                {"cubature_integrand",9,0,3,1,0,0.,0.,0,N_mdm(ushint,cubIntOrder)},
                {"diagonal_covariance",8,0,5,0,0,0.,0.,0,N_mdm(type,covarianceControl_DIAGONAL_COVARIANCE)}
      ,
                {"expansion_order",13,5,3,1,kw_209,0.,0.,0,N_mdm(usharray,expansionOrder)},
                {"export_expansion_file",11,0,10,0,0,0.,0.,0,N_mdm(str,expansionExportFile)},
                {"export_points_file",11,2,9,0,kw_210,0.,0.,0,N_mdm(str,approxExportFile)},
                {"full_covariance",8,0,5,0,0,0.,0.,0,N_mdm(type,covarianceControl_FULL_COVARIANCE)},
                {"import_expansion_file",11,0,3,1,0,0.,0.,0,N_mdm(str,expansionImportFile)},
                {"least_interpolation",0,5,3,1,kw_211,0.,0.,3,N_mdm(type,
      regressionType_ORTHOG_LEAST_INTERPOLATION)},
                {"normalized",8,0,6,0,0,0.,0.,0,N_mdm(true,normalizedCoeffs)},
                {"oli",0,5,3,1,kw_211,0.,0.,1,N_mdm(type,regressionType_ORTHOG_LEAST_INTERPOLATION)},
                {"orthogonal_least_interpolation",8,5,3,1,kw_211,0.,0.,0,N_mdm(type,
      regressionType_ORTHOG_LEAST_INTERPOLATION)},
                {"p_refinement",8,2,1,0,kw_213,0.,0.,0,N_mdm(type,refinementType_P_REFINEMENT)},
                {"probability_refinement",8,4,8,0,kw_214},
                {"quadrature_order",13,3,3,1,kw_215,0.,0.,0,N_mdm(usharray,quadratureOrder)},
                {"sample_refinement",0,4,8,0,kw_214,0.,0.,-2},
                {"sample_type",8,2,7,0,kw_216},
                {"sparse_grid_level",13,2,3,1,kw_217,0.,0.,0,N_mdm(usharray,sparseGridLevel)},
                {"variance_based_decomp",8,2,4,0,kw_218,0.,0.,0,N_mdm(true,vbdFlag)},
                {"wiener",8,0,2,0,0,0.,0.,0,N_mdm(type,expansionType_STD_NORMAL_U)}
                }
```

KeyWord kw_220[1] [static]

Initial value:

```
= {
                {"previous_samples",9,0,1,1,0,0.,0.,0,N_mdm(int,previousSamples)}
                }
```

KeyWord kw_221[4] [static]

Initial value:

```
= {
                {"incremental_lhs",8,1,1,1,kw_220,0.,0.,0,N_mdm(utype,sampleType_SUBMETHOD_INCREMENTAL_LHS)
      },
                {"incremental_random",8,1,1,1,kw_220,0.,0.,0,N_mdm(utype,
      sampleType_SUBMETHOD_INCREMENTAL_RANDOM)},
                {"lhs",8,0,1,1,0,0.,0.,0,N_mdm(utype,sampleType_SUBMETHOD_LHS)},
                {"random",8,0,1,1,0,0.,0.,0,N_mdm(utype,sampleType_SUBMETHOD_RANDOM)}
                }
```

KeyWord kw_222[1] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,1,0,0,0.,0.,0,N_mdm(Real,vbdDropTolerance)}
                }
```

KeyWord kw_223[8] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {0,0,4,0,0,kw_28},
                {0,0,2,0,0,kw_196},
                {0,0,1,0,0,kw_197},
                {"backfill",8,0,3,0,0,0.,0.,0,N_mdm(true,backfillFlag)},
                {"sample_type",8,4,1,0,kw_221},
                {"variance_based_decomp",8,1,2,0,kw_222,0.,0.,0,N_mdm(true,vbdFlag)}
                }
```

KeyWord kw_224[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mdm(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mdm(false,approxExportAnnotated)}
                }
```

KeyWord kw_225[2] [static]

Initial value:

```
= {
                {"generalized",8,0,1,1,0,0.,0.,0,N_mdm(type,
      refinementControl_DIMENSION_ADAPTIVE_CONTROL_GENERALIZED)},
                {"sobol",8,0,1,1,0,0.,0.,0,N_mdm(type,refinementControl_DIMENSION_ADAPTIVE_CONTROL_SOBOL)}
                }
```

KeyWord kw_226[3] [static]

Initial value:

```
= {
                {"dimension_adaptive",8,2,1,1,kw_225},
                {"local_adaptive",8,0,1,1,0,0.,0.,0,N_mdm(type,refinementControl_LOCAL_ADAPTIVE_CONTROL)},
                {"uniform",8,0,1,1,0,0.,0.,0,N_mdm(type,refinementControl_UNIFORM_CONTROL)}
                }
```

KeyWord kw_227[2] [static]

Initial value:

```
= {
                {"generalized",8,0,1,1,0,0.,0.,0,N_mdm(type,
      refinementControl_DIMENSION_ADAPTIVE_CONTROL_GENERALIZED)},
                {"sobol",8,0,1,1,0,0.,0.,0,N_mdm(type,refinementControl_DIMENSION_ADAPTIVE_CONTROL_SOBOL)}
                }
```

KeyWord kw_228[2] [static]

Initial value:

```
= {
                {"dimension_adaptive",8,2,1,1,kw_227},
                {"uniform",8,0,1,1,0,0.,0.,0,N_mdm(type,refinementControl_UNIFORM_CONTROL)}
                }
```

KeyWord kw_229[4] [static]

Initial value:

```
= {
                {"adapt_import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_AIS)},
                {"import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_IS)},
                {"mm_adapt_import",8,0,1,1,0,0.,0.,0,N_mdm(utype,integrationRefine_MMAIS)},
                {"refinement_samples",9,0,2,0,0,0.,0.,0,N_mdm(int,refineSamples)}
                }
```

KeyWord kw_230[2] [static]

Initial value:

```
= {
                {"lhs",8,0,1,1,0,0.,0.,0,N_mdm(utype,sampleType_SUBMETHOD_LHS)},
                {"random",8,0,1,1,0,0.,0.,0,N_mdm(utype,sampleType_SUBMETHOD_RANDOM)}
                }
```

KeyWord kw_231[4] [static]

Initial value:

```
= {
                {"hierarchical",8,0,2,0,0,0.,0.,0,N_mdm(type,expansionBasisType_HIERARCHICAL_INTERPOLANT)},
                {"nodal",8,0,2,0,0,0.,0.,0,N_mdm(type,expansionBasisType_NODAL_INTERPOLANT)},
                {"restricted",8,0,1,0,0,0.,0.,0,N_mdm(type,growthOverride_RESTRICTED)},
                {"unrestricted",8,0,1,0,0,0.,0.,0,N_mdm(type,growthOverride_UNRESTRICTED)}
                }
```

KeyWord kw_232[2] [static]

Initial value:

```
= {
                {"drop_tolerance",10,0,2,0,0,0.,0.,0,N_mdm(Real,vbdDropTolerance)},
                {"interaction_order",0x19,0,1,0,0,0.,0.,0,N_mdm(ushint,vbdOrder)}
                }
```

KeyWord kw_233[23] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {0,0,4,0,0,kw_28},
                {0,0,2,0,0,kw_196},
                {0,0,1,0,0,kw_197},
                {"askey",8,0,2,0,0,0.,0.,0,N_mdm(type,expansionType_ASKEY_U)},
                {"diagonal_covariance",8,0,8,0,0,0.,0.,0,N_mdm(type,covarianceControl_DIAGONAL_COVARIANCE)}
      ,
                {"dimension_preference",14,0,4,0,0,0.,0.,0,N_mdm(RealDL,anisoDimPref)},
                {"export_points_file",11,2,11,0,kw_224,0.,0.,0,N_mdm(str,approxExportFile)},
                {"full_covariance",8,0,8,0,0,0.,0.,0,N_mdm(type,covarianceControl_FULL_COVARIANCE)},
                {"h_refinement",8,3,1,0,kw_226,0.,0.,0,N_mdm(type,refinementType_H_REFINEMENT)},
                {"nested",8,0,6,0,0,0.,0.,0,N_mdm(type,nestingOverride_NESTED)},
                {"non_nested",8,0,6,0,0,0.,0.,0,N_mdm(type,nestingOverride_NON_NESTED)},
                {"p_refinement",8,2,1,0,kw_228,0.,0.,0,N_mdm(type,refinementType_P_REFINEMENT)},
                {"piecewise",8,0,2,0,0,0.,0.,0,NIDRProblemDescDB::method_piecewise},
                {"probability_refinement",8,4,10,0,kw_229},
```

```
                {"quadrature_order",13,0,3,1,0,0.,0.,0,N_mdm(usharray,quadratureOrder)},
                {"sample_refinement",0,4,10,0,kw_229,0.,0.,-2},
                {"sample_type",8,2,9,0,kw_230},
                {"sparse_grid_level",13,4,3,1,kw_231,0.,0.,0,N_mdm(usharray,sparseGridLevel)},
                {"use_derivatives",8,0,5,0,0,0.,0.,0,N_mdm(true,methodUseDerivsFlag)},
                {"variance_based_decomp",8,2,7,0,kw_232,0.,0.,0,N_mdm(true,vbdFlag)},
                {"wiener",8,0,2,0,0,0.,0.,0,N_mdm(type,expansionType_STD_NORMAL_U)}
                }
```

KeyWord kw_234[2] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"misc_options",15,0,1,0,0,0.,0.,0,N_mdm(strL,miscOptions)}
                }
```

KeyWord kw_235[3] [static]

Initial value:

```
= {
                {"function_precision",10,0,2,0,0,0.,0.,0,N_mdm(Real,functionPrecision)},
                {"linesearch_tolerance",10,0,3,0,0,0.,0.,0,N_mdm(Real,lineSearchTolerance)},
                {"verify_level",9,0,1,0,0,0.,0.,0,N_mdm(int,verifyLevel)}
                }
```

KeyWord kw_236[4] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {0,0,3,0,0,kw_235},
                {""}
                }
```

KeyWord kw_237[2] [static]

Initial value:

```
= {
                {"gradient_tolerance",10,0,2,0,0,0.,0.,0,N_mdm(Real,gradientTolerance)},
                {"max_step",10,0,1,0,0,0.,0.,0,N_mdm(Real,maxStep)}
                }
```

KeyWord kw_238[4] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {0,0,2,0,0,kw_237},
                {""}
                }
```

KeyWord kw_239[3] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {"search_scheme_size",9,0,1,0,0,0.,0.,0,N_mdm(int,searchSchemeSize)}
                }
```

KeyWord kw_240[3] [static]

Initial value:

```
= {
                {"argaez_tapia",8,0,1,1,0,0.,0.,0,N_mdm(type,meritFn_ArgaezTapia)},
                {"el_bakry",8,0,1,1,0,0.,0.,0,N_mdm(type,meritFn_NormFmu)},
                {"van_shanno",8,0,1,1,0,0.,0.,0,N_mdm(type,meritFn_VanShanno)}
                }
```

KeyWord kw_241[4] [static]

Initial value:

```
= {
                {"gradient_based_line_search",8,0,1,1,0,0.,0.,0,N_mdm(lit,
      searchMethod_gradient_based_line_search)},
                {"tr_pds",8,0,1,1,0,0.,0.,0,N_mdm(lit,searchMethod_tr_pds)},
                {"trust_region",8,0,1,1,0,0.,0.,0,N_mdm(lit,searchMethod_trust_region)},
                {"value_based_line_search",8,0,1,1,0,0.,0.,0,N_mdm(lit,searchMethod_value_based_line_search
      )}
                }
```

KeyWord kw_242[7] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {0,0,2,0,0,kw_237},
                {"centering_parameter",10,0,4,0,0,0.,0.,0,N_mdm(Real,centeringParam)},
                {"merit_function",8,3,2,0,kw_240},
                {"search_method",8,4,1,0,kw_241},
                {"steplength_to_boundary",10,0,3,0,0,0.,0.,0,N_mdm(Real,stepLenToBoundary)}
                }
```

KeyWord kw_243[5] [static]

Initial value:

```
= {
                {"debug",8,0,1,1,0,0.,0.,0,N_mdm(type,methodOutput_DEBUG_OUTPUT)},
                {"normal",8,0,1,1,0,0.,0.,0,N_mdm(type,methodOutput_NORMAL_OUTPUT)},
                {"quiet",8,0,1,1,0,0.,0.,0,N_mdm(type,methodOutput_QUIET_OUTPUT)},
                {"silent",8,0,1,1,0,0.,0.,0,N_mdm(type,methodOutput_SILENT_OUTPUT)},
                {"verbose",8,0,1,1,0,0.,0.,0,N_mdm(type,methodOutput_VERBOSE_OUTPUT)}
                }
```

KeyWord kw_244[2] [static]

Initial value:

```
= {
                {"model_pointer",11,0,1,0,0,0.,0.,0,N_mdm(str,subModelPointer)},
                {"opt_model_pointer",3,0,1,0,0,0.,0.,-1,N_mdm(str,subModelPointer)}
                }
```

KeyWord kw_245[1] [static]

Initial value:

```
= {
                {"seed",9,0,1,0,0,0.,0.,0,N_mdm(int,randomSeed)}
                }
```

KeyWord kw_246[8] [static]

Initial value:

```
= {
                {0,0,3,0,0,kw_133},
                {"method_name",11,2,1,1,kw_244,0.,0.,0,N_mdm(str,subMethodName)},
                {"method_pointer",11,0,1,1,0,0.,0.,0,N_mdm(str,subMethodPointer)},
                {"multi_objective_weight_sets",6,0,3,0,0,0.,0.,4,N_mdm(RealDL,concurrentParameterSets)},
                {"opt_method_name",3,2,1,1,kw_244,0.,0.,-3,N_mdm(str,subMethodName)},
                {"opt_method_pointer",3,0,1,1,0,0.,0.,-3,N_mdm(str,subMethodPointer)},
                {"random_weight_sets",9,1,2,0,kw_245,0.,0.,0,N_mdm(int,concurrentRandomJobs)},
                {"weight_sets",14,0,3,0,0,0.,0.,0,N_mdm(RealDL,concurrentParameterSets)}
                }
```

KeyWord kw_247[3] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,2,0,0,kw_23},
                {"partitions",13,0,1,0,0,0.,0.,0,N_mdm(usharray,varPartitions)}
                }
```

KeyWord kw_248[5] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"converge_order",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_CONVERGE_ORDER)},
                {"converge_qoi",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_CONVERGE_QOI)},
                {"estimate_order",8,0,1,1,0,0.,0.,0,N_mdm(utype,subMethod_SUBMETHOD_ESTIMATE_ORDER)},
                {"refinement_rate",10,0,2,0,0,0.,0.,0,N_mdm(Real,refinementRate)}
                }
```

KeyWord kw_249[2] [static]

Initial value:

```
= {
                {"num_generations",0x29,0,2,0,0,0.,0.,0,N_mdm(sizet,numGenerations)},
                {"percent_change",10,0,1,0,0,0.,0.,0,N_mdm(Realz,convergenceTolerance)}
                }
```

KeyWord kw_250[2] [static]

Initial value:

```
= {
                {"num_generations",0x29,0,2,0,0,0.,0.,0,N_mdm(sizet,numGenerations)},
                {"percent_change",10,0,1,0,0,0.,0.,0,N_mdm(Realz,convergenceTolerance)}
                }
```

KeyWord kw_251[2] [static]

Initial value:

```
= {
                {"average_fitness_tracker",8,2,1,1,kw_249,0.,0.,0,N_mdm(lit,
      convergenceType_average_fitness_tracker)},
                {"best_fitness_tracker",8,2,1,1,kw_250,0.,0.,0,N_mdm(lit,
      convergenceType_best_fitness_tracker)}
                }
```

KeyWord kw_252[2] [static]

Initial value:

```
= {
                {"constraint_penalty",10,0,2,0,0,0.,0.,0,N_mdm(Realp,constraintTolerance)},
                {"merit_function",8,0,1,1,0,0.,0.,0,N_mdm(lit,fitnessType_merit_function)}
                }
```

KeyWord kw_253[4] [static]

Initial value:

```
= {
                {"elitist",8,0,1,1,0,0.,0.,0,N_mdm(lit,replacementType_elitist)},
                {"favor_feasible",8,0,1,1,0,0.,0.,0,N_mdm(lit,replacementType_favor_feasible)},
                {"roulette_wheel",8,0,1,1,0,0.,0.,0,N_mdm(lit,replacementType_roulette_wheel)},
                {"unique_roulette_wheel",8,0,1,1,0,0.,0.,0,N_mdm(lit,replacementType_unique_roulette_wheel)
      }
                }
```

KeyWord kw_254[6] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {0,0,7,0,0,kw_170},
                {"convergence_type",8,2,3,0,kw_251},
                {"fitness_type",8,2,1,0,kw_252},
                {"replacement_type",8,4,2,0,kw_253}
                }
```

KeyWord kw_255[5] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {0,0,9,0,0,kw_37},
                {0,0,3,0,0,kw_235},
                {"nlssol",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_NLSSOL_SQP)},
                {"npsol",8,0,1,1,0,0.,0.,0,N_mdm(utype,methodName_NPSOL_SQP)}
                }
```

KeyWord kw_256[7] [static]

Initial value:

```
= {
                {"approx_method_name",3,0,1,1,0,0.,0.,3,N_mdm(str,subMethodName)},
                {"approx_method_pointer",3,0,1,1,0,0.,0.,3,N_mdm(str,subMethodPointer)},
                {"approx_model_pointer",3,0,2,2,0,0.,0.,3,N_mdm(str,modelPointer)},
                {"method_name",11,0,1,1,0,0.,0.,0,N_mdm(str,subMethodName)},
                {"method_pointer",11,0,1,1,0,0.,0.,0,N_mdm(str,subMethodPointer)},
                {"model_pointer",11,0,2,2,0,0.,0.,0,N_mdm(str,modelPointer)},
                {"replace_points",8,0,3,0,0,0.,0.,0,N_mdm(true,surrBasedGlobalReplacePts)}
                }
```

KeyWord kw_257[2] [static]

Initial value:

```
= {
                {"filter",8,0,1,1,0,0.,0.,0,N_mdm(type,surrBasedLocalAcceptLogic_FILTER)},
                {"tr_ratio",8,0,1,1,0,0.,0.,0,N_mdm(type,surrBasedLocalAcceptLogic_TR_RATIO)}
                }
```

KeyWord kw_258[7] [static]

Initial value:

```
= {
                {"augmented_lagrangian_objective",8,0,1,1,0,0.,0.,0,N_mdm(type,
      surrBasedLocalSubProbObj_AUGMENTED_LAGRANGIAN_OBJECTIVE)},
                {"lagrangian_objective",8,0,1,1,0,0.,0.,0,N_mdm(type,
      surrBasedLocalSubProbObj_LAGRANGIAN_OBJECTIVE)},
                {"linearized_constraints",8,0,2,2,0,0.,0.,0,N_mdm(type,
      surrBasedLocalSubProbCon_LINEARIZED_CONSTRAINTS)},
                {"no_constraints",8,0,2,2,0,0.,0.,0,N_mdm(type,surrBasedLocalSubProbCon_NO_CONSTRAINTS)},
                {"original_constraints",8,0,2,2,0,0.,0.,0,N_mdm(type,
      surrBasedLocalSubProbCon_ORIGINAL_CONSTRAINTS)},
                {"original_primary",8,0,1,1,0,0.,0.,0,N_mdm(type,surrBasedLocalSubProbObj_ORIGINAL_PRIMARY)
      },
                {"single_objective",8,0,1,1,0,0.,0.,0,N_mdm(type,surrBasedLocalSubProbObj_SINGLE_OBJECTIVE)
      }
                }
```

KeyWord kw_259[1] [static]

Initial value:

```
= {
                {"homotopy",8,0,1,1,0,0.,0.,0,N_mdm(type,surrBasedLocalConstrRelax_HOMOTOPY)}
                }
```

KeyWord kw_260[4] [static]

Initial value:

```
= {
                {"adaptive_penalty_merit",8,0,1,1,0,0.,0.,0,N_mdm(type,
      surrBasedLocalMeritFn_ADAPTIVE_PENALTY_MERIT)},
                {"augmented_lagrangian_merit",8,0,1,1,0,0.,0.,0,N_mdm(type,
      surrBasedLocalMeritFn_AUGMENTED_LAGRANGIAN_MERIT)},
                {"lagrangian_merit",8,0,1,1,0,0.,0.,0,N_mdm(type,surrBasedLocalMeritFn_LAGRANGIAN_MERIT)},
                {"penalty_merit",8,0,1,1,0,0.,0.,0,N_mdm(type,surrBasedLocalMeritFn_PENALTY_MERIT)}
                }
```

KeyWord kw_261[6] [static]

Initial value:

```
= {
                {"contract_threshold",10,0,3,0,0,0.,0.,0,N_mdm(Real,surrBasedLocalTRContractTrigger)},
                {"contraction_factor",10,0,5,0,0,0.,0.,0,N_mdm(Real,surrBasedLocalTRContract)},
                {"expand_threshold",10,0,4,0,0,0.,0.,0,N_mdm(Real,surrBasedLocalTRExpandTrigger)},
                {"expansion_factor",10,0,6,0,0,0.,0.,0,N_mdm(Real,surrBasedLocalTRExpand)},
                {"initial_size",10,0,1,0,0,0.,0.,0,N_mdm(Real,surrBasedLocalTRInitSize)},
                {"minimum_size",10,0,2,0,0,0.,0.,0,N_mdm(Real,surrBasedLocalTRMinSize)}
                }
```

KeyWord kw_262[13] [static]

Initial value:

```
= {
                {"acceptance_logic",8,2,8,0,kw_257},
                {"approx_method_name",3,0,1,1,0,0.,0.,6,N_mdm(str,subMethodName)},
                {"approx_method_pointer",3,0,1,1,0,0.,0.,6,N_mdm(str,subMethodPointer)},
                {"approx_model_pointer",3,0,2,2,0,0.,0.,6,N_mdm(str,modelPointer)},
                {"approx_subproblem",8,7,6,0,kw_258},
                {"constraint_relax",8,1,9,0,kw_259},
                {"merit_function",8,4,7,0,kw_260},
                {"method_name",11,0,1,1,0,0.,0.,0,N_mdm(str,subMethodName)},
                {"method_pointer",11,0,1,1,0,0.,0.,0,N_mdm(str,subMethodPointer)},
                {"model_pointer",11,0,2,2,0,0.,0.,0,N_mdm(str,modelPointer)},
                {"soft_convergence_limit",9,0,3,0,0,0.,0.,0,N_mdm(ushint,softConvLimit)},
                {"trust_region",8,6,5,0,kw_261,0.,0.,0,0,0,NIDRProblemDescDB::method_tr_final},
                {"truth_surrogate_bypass",8,0,4,0,0,0.,0.,0,N_mdm(true,surrBasedLocalLayerBypass)}
                }
```

KeyWord kw_263[4] [static]

Initial value:

```
= {
                {0,0,1,0,0,kw_22},
                {"final_point",14,0,1,1,0,0.,0.,0,N_mdm(RealDL,finalPoint)},
                {"num_steps",9,0,2,2,0,0.,0.,0,N_mdm(int,numSteps)},
                {"step_vector",14,0,1,1,0,0.,0.,0,N_mdm(RealDL,stepVector)}
                }
```

KeyWord kw_265[1] [static]

Initial value:

```
= {
                {"optional_interface_responses_pointer",11,0,1,0,0,0.,0.,0,N_mom(str,
      optionalInterfRespPointer)}
                }
```

KeyWord kw_266[2] [static]

Initial value:

```
= {
                {"master",8,0,1,1,0,0.,0.,0,N_mom(type,subMethodScheduling_MASTER_SCHEDULING)},
                {"peer",8,0,1,1,0,0.,0.,0,N_mom(type,subMethodScheduling_PEER_SCHEDULING)}
                }
```

KeyWord kw_267[7] [static]

Initial value:

```
= {
                {"iterator_scheduling",8,2,2,0,kw_266},
                {"iterator_servers",0x19,0,1,0,0,0.,0.,0,N_mom(pint,subMethodServers)},
                {"primary_response_mapping",14,0,6,0,0,0.,0.,0,N_mom(RealDL,primaryRespCoeffs)},
                {"primary_variable_mapping",15,0,4,0,0,0.,0.,0,N_mom(strL,primaryVarMaps)},
                {"processors_per_iterator",0x19,0,3,0,0,0.,0.,0,N_mom(pint,subMethodProcs)},
                {"secondary_response_mapping",14,0,7,0,0,0.,0.,0,N_mom(RealDL,secondaryRespCoeffs)},
                {"secondary_variable_mapping",15,0,5,0,0,0.,0.,0,N_mom(strL,secondaryVarMaps)}
                }
```

KeyWord kw_268[2] [static]

Initial value:

```
= {
                {"optional_interface_pointer",11,1,1,0,kw_265,0.,0.,0,N_mom(str,interfacePointer)},
                {"sub_method_pointer",11,7,2,1,kw_267,0.,0.,0,N_mom(str,subMethodPointer)}
                }
```

KeyWord kw_269[1] [static]

Initial value:

```
= {
                {"interface_pointer",11,0,1,0,0,0.,0.,0,N_mom(str,interfacePointer)}
                }
```

KeyWord kw_270[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mom(true,approxChallengeActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mom(true,approxChallengeAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mom(false,approxChallengeAnnotated)}
                }
```

KeyWord kw_271[6] [static]

Initial value:

```
= {
                {"additive",8,0,2,2,0,0.,0.,0,N_mom(type,approxCorrectionType_ADDITIVE_CORRECTION)},
                {"combined",8,0,2,2,0,0.,0.,0,N_mom(type,approxCorrectionType_COMBINED_CORRECTION)},
                {"first_order",8,0,1,1,0,0.,0.,0,N_mom(order,approxCorrectionOrder_1)},
                {"multiplicative",8,0,2,2,0,0.,0.,0,N_mom(type,
     approxCorrectionType_MULTIPLICATIVE_CORRECTION)},
                {"second_order",8,0,1,1,0,0.,0.,0,N_mom(order,approxCorrectionOrder_2)},
                {"zeroth_order",8,0,1,1,0,0.,0.,0,N_mom(order,approxCorrectionOrder_0)}
                }
```

KeyWord kw_272[2] [static]

Initial value:

```
= {
                {"folds",9,0,1,0,0,0.,0.,0,N_mom(int,numFolds)},
                {"percent",10,0,1,0,0,0.,0.,0,N_mom(Real,percentFold)}
                }
```

KeyWord kw_273[2] [static]

Initial value:

```
= {
                {"cross_validation",8,2,1,0,kw_272,0.,0.,0,N_mom(true,crossValidateFlag)},
                {"press",8,0,2,0,0,0.,0.,0,N_mom(true,pressFlag)}
                }
```

KeyWord kw_274[2] [static]

Initial value:

```
= {
                {"annotated",8,0,1,0,0,0.,0.,0,N_mom(true,approxExportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mom(false,approxExportAnnotated)}
                }
```

KeyWord kw_275[3] [static]

Initial value:

```
= {
                {"constant",8,0,1,1,0,0.,0.,0,N_mom(lit,trendOrder_constant)},
                {"linear",8,0,1,1,0,0.,0.,0,N_mom(lit,trendOrder_linear)},
                {"reduced_quadratic",8,0,1,1,0,0.,0.,0,N_mom(lit,trendOrder_reduced_quadratic)}
                }
```

KeyWord kw_276[2] [static]

Initial value:

```
= {
                {"point_selection",8,0,1,0,0,0.,0.,0,N_mom(true,pointSelection)},
                {"trend",8,3,2,0,kw_275}
                }
```

KeyWord kw_277[4] [static]

Initial value:

```
= {
                {"constant",8,0,1,1,0,0.,0.,0,N_mom(lit,trendOrder_constant)},
                {"linear",8,0,1,1,0,0.,0.,0,N_mom(lit,trendOrder_linear)},
                {"quadratic",8,0,1,1,0,0.,0.,0,N_mom(lit,trendOrder_quadratic)},
                {"reduced_quadratic",8,0,1,1,0,0.,0.,0,N_mom(lit,trendOrder_reduced_quadratic)}
                }
```

KeyWord kw_278[7] [static]

Initial value:

```
= {
                {"correlation_lengths",14,0,5,0,0,0.,0.,0,N_mom(RealDL,krigingCorrelations)},
                {"export_model_file",11,0,6,0,0,0.,0.,0,N_mom(str, approxExportModelFile)},
                {"find_nugget",9,0,4,0,0,0.,0.,0,N_mom(shint,krigingFindNugget)},
                {"max_trials",0x19,0,3,0,0,0.,0.,0,N_mom(shint,krigingMaxTrials)},
                {"nugget",0x1a,0,4,0,0,0.,0.,0,N_mom(Real,krigingNugget)},
                {"optimization_method",11,0,2,0,0,0.,0.,0,N_mom(str,krigingOptMethod)},
                {"trend",8,4,1,0,kw_277}
                }
```

KeyWord kw_279[2] [static]

Initial value:

```
= {
                {"dakota",8,2,1,1,kw_276,0.,0.,0,N_mom(lit,surrogateType_global_gaussian)},
                {"surfpack",8,7,1,1,kw_278,0.,0.,0,N_mom(lit,surrogateType_global_kriging)}
                }
```

KeyWord kw_280[3] [static]

Initial value:

```
= {
                {"active_only",8,0,2,0,0,0.,0.,0,N_mdm(true,approxImportActive)},
                {"annotated",8,0,1,0,0,0.,0.,0,N_mom(true,approxImportAnnotated)},
                {"freeform",8,0,1,0,0,0.,0.,0,N_mom(false,approxImportAnnotated)}
                }
```

KeyWord kw_281[2] [static]

Initial value:

```
= {
                {"cubic",8,0,1,1,0,0.,0.,0,N_mom(lit,marsInterpolation_cubic)},
                {"linear",8,0,1,1,0,0.,0.,0,N_mom(lit,marsInterpolation_linear)}
                }
```

KeyWord kw_282[3] [static]

Initial value:

```
= {
                {"export_model_file",11,0,3,0,0,0.,0.,0,N_mom(str, approxExportModelFile)},
                {"interpolation",8,2,2,0,kw_281},
                {"max_bases",9,0,1,0,0,0.,0.,0,N_mom(shint,marsMaxBases)}
                }
```

KeyWord kw_283[3] [static]

Initial value:

```
= {
                {"export_model_file",11,0,3,0,0,0.,0.,0,N_mom(str, approxExportModelFile)},
                {"poly_order",9,0,1,0,0,0.,0.,0,N_mom(shint,mlsPolyOrder)},
                {"weight_function",9,0,2,0,0,0.,0.,0,N_mom(shint,mlsWeightFunction)}
                }
```

KeyWord kw_284[5] [static]

Initial value:

```
= {
                {"export_model_file",11,0,4,0,0,0.,0.,0,N_mom(str, approxExportModelFile)},
                {"max_nodes",9,0,1,0,0,0.,0.,0,N_mom(shint,annNodes)},
                {"nodes",1,0,1,0,0,0.,0.,-1,N_mom(shint,annNodes)},
                {"random_weight",9,0,3,0,0,0.,0.,0,N_mom(shint,annRandomWeight)},
                {"range",10,0,2,0,0,0.,0.,0,N_mom(Real,annRange)}
                }
```

KeyWord kw_285[4] [static]

Initial value:

```
= {
                {"cubic",8,0,1,1,0,0.,0.,0,N_mom(order,polynomialOrder_3)},
                {"export_model_file",11,0,2,0,0,0.,0.,0,N_mom(str, approxExportModelFile)},
                {"linear",8,0,1,1,0,0.,0.,0,N_mom(order,polynomialOrder_1)},
                {"quadratic",8,0,1,1,0,0.,0.,0,N_mom(order,polynomialOrder_2)}
                }
```

KeyWord kw_286[5] [static]

Initial value:

```
= {
                {"bases",9,0,1,0,0,0.,0.,0,N_mom(shint,rbfBases)},
                {"export_model_file",11,0,5,0,0,0.,0.,0,N_mom(str, approxExportModelFile)},
                {"max_pts",9,0,2,0,0,0.,0.,0,N_mom(shint,rbfMaxPts)},
                {"max_subsets",9,0,4,0,0,0.,0.,0,N_mom(shint,rbfMaxSubsets)},
                {"min_partition",9,0,3,0,0,0.,0.,0,N_mom(shint,rbfMinPartition)}
                }
```

KeyWord kw_287[3] [static]

Initial value:

```
= {
                {"all",8,0,1,1,0,0.,0.,0,N_mom(lit,approxPointReuse_all)},
                {"none",8,0,1,1,0,0.,0.,0,N_mom(lit,approxPointReuse_none)},
                {"region",8,0,1,1,0,0.,0.,0,N_mom(lit,approxPointReuse_region)}
                }
```

KeyWord kw_288[21] [static]

Initial value:

```
= {
                {"challenge_points_file",11,3,10,0,kw_270,0.,0.,0,N_mom(str,approxChallengeFile)},
                {"correction",8,6,8,0,kw_271},
                {"dace_method_pointer",11,0,3,0,0,0.,0.,0,N_mom(str,subMethodPointer)},
                {"diagnostics",7,2,9,0,kw_273,0.,0.,6,N_mom(strL,diagMetrics)},
                {"export_points_file",11,2,6,0,kw_274,0.,0.,0,N_mom(str,approxExportFile)},
                {"gaussian_process",8,2,1,1,kw_279},
                {"import_points_file",11,3,5,0,kw_280,0.,0.,0,N_mom(str,approxImportFile)},
                {"kriging",0,2,1,1,kw_279,0.,0.,-2},
                {"mars",8,3,1,1,kw_282,0.,0.,0,N_mom(lit,surrogateType_global_mars)},
                {"metrics",15,2,9,0,kw_273,0.,0.,0,N_mom(strL,diagMetrics)},
                {"minimum_points",8,0,2,0,0,0.,0.,0,N_mom(type,pointsManagement_MINIMUM_POINTS)},
                {"moving_least_squares",8,3,1,1,kw_283,0.,0.,0,N_mom(lit,
```

```
surrogateType_global_moving_least_squares)},
          {"neural_network",8,5,1,1,kw_284,0.,0.,0,N_mom(lit,surrogateType_global_neural_network)},
          {"polynomial",8,4,1,1,kw_285,0.,0.,0,N_mom(lit,surrogateType_global_polynomial)},
          {"radial_basis",8,5,1,1,kw_286,0.,0.,0,N_mom(lit,surrogateType_global_radial_basis)},
          {"recommended_points",8,0,2,0,0,0.,0.,0,N_mom(type,pointsManagement_RECOMMENDED_POINTS)},
          {"reuse_points",8,3,4,0,kw_287},
          {"reuse_samples",0,3,4,0,kw_287,0.,0.,-1},
          {"samples_file",3,3,5,0,kw_280,0.,0.,-12,N_mom(str,approxImportFile)},
          {"total_points",9,0,2,0,0,0.,0.,0,N_mom(int,pointsTotal)},
          {"use_derivatives",8,0,7,0,0,0.,0.,0,N_mom(true,modelUseDerivsFlag)}
          }
```

KeyWord kw_289[6] [static]

Initial value:

```
= {
          {"additive",8,0,2,2,0,0.,0.,0,N_mom(type,approxCorrectionType_ADDITIVE_CORRECTION)},
          {"combined",8,0,2,2,0,0.,0.,0,N_mom(type,approxCorrectionType_COMBINED_CORRECTION)},
          {"first_order",8,0,1,1,0,0.,0.,0,N_mom(order,approxCorrectionOrder_1)},
          {"multiplicative",8,0,2,2,0,0.,0.,0,N_mom(type,
approxCorrectionType_MULTIPLICATIVE_CORRECTION)},
          {"second_order",8,0,1,1,0,0.,0.,0,N_mom(order,approxCorrectionOrder_2)},
          {"zeroth_order",8,0,1,1,0,0.,0.,0,N_mom(order,approxCorrectionOrder_0)}
          }
```

KeyWord kw_290[3] [static]

Initial value:

```
= {
          {"correction",8,6,3,3,kw_289},
          {"high_fidelity_model_pointer",11,0,2,2,0,0.,0.,0,N_mom(str,truthModelPointer)},
          {"low_fidelity_model_pointer",11,0,1,1,0,0.,0.,0,N_mom(str,lowFidelityModelPointer)}
          }
```

KeyWord kw_291[1] [static]

Initial value:

```
= {
          {"actual_model_pointer",11,0,1,1,0,0.,0.,0,N_mom(str,truthModelPointer)}
          }
```

KeyWord kw_292[2] [static]

Initial value:

```
= {
          {0,0,1,0,0,kw_291},
          {"taylor_series",8,0,1,1}
          }
```

KeyWord kw_293[2] [static]

Initial value:

```
= {
          {0,0,1,0,0,kw_291},
          {"tana",8,0,1,1}
          }
```

KeyWord kw_294[5] [static]

Initial value:

```
= {
                {"global",8,21,2,1,kw_288},
                {"hierarchical",8,3,2,1,kw_290,0.,0.,0,N_mom(lit,surrogateType_hierarchical)},
                {"id_surrogates",13,0,1,0,0,0.,0.,0,N_mom(intsetm1,surrogateFnIndices)},
                {"local",8,1,2,1,kw_292,0.,0.,0,N_mom(lit,surrogateType_local_taylor)},
                {"multipoint",8,1,2,1,kw_293,0.,0.,0,N_mom(lit,surrogateType_multipoint_tana)}
                }
```

KeyWord kw_295[7] [static]

Initial value:

```
= {
                {"hierarchical_tagging",8,0,4,0,0,0.,0.,0,N_mom(true,hierarchicalTags)},
                {"id_model",11,0,1,0,0,0.,0.,0,N_mom(str,idModel)},
                {"nested",8,2,5,1,kw_268,0.,0.,0,N_mom(lit,modelType_nested)},
                {"responses_pointer",11,0,3,0,0,0.,0.,0,N_mom(str,responsesPointer)},
                {"single",8,1,5,1,kw_269,0.,0.,0,N_mom(lit,modelType_single)},
                {"surrogate",8,5,5,1,kw_294,0.,0.,0,N_mom(lit,modelType_surrogate)},
                {"variables_pointer",11,0,2,0,0,0.,0.,0,N_mom(str,variablesPointer)}
                }
```

KeyWord kw_296[6] [static]

Initial value:

```
= {
                {"annotated",8,0,2,0,0,0.,0.,0,N_rem(true,expDataFileAnnotated)},
                {"freeform",8,0,2,0,0,0.,0.,0,N_rem(false,expDataFileAnnotated)},
                {"num_config_variables",0x29,0,3,0,0,0.,0.,0,N_rem(sizet,numExpConfigVars)},
                {"num_experiments",0x29,0,1,0,0,0.,0.,0,N_rem(sizet,numExperiments)},
                {"num_std_deviations",0x29,0,5,0,0,0.,0.,0,N_rem(sizet,numExpStdDeviations)},
                {"sigma_type",0x80f,0,4,0,0,0.,0.,0,N_rem(strL,sigmaType)}
                }
```

KeyWord kw_297[9] [static]

Initial value:

```
= {
                {"annotated",8,0,2,0,0,0.,0.,0,N_rem(true,expDataFileAnnotated)},
                {"config_data_file",11,0,6,0,0,0.,0.,0,N_rem(str,configDataFileName)},
                {"field_coordinate_data_file",11,0,7,0,0,0.,0.,0,N_rem(str,fieldCoordDataFileName)},
                {"field_data_file",11,0,5,0,0,0.,0.,0,N_rem(str,fieldDataFileName)},
                {"freeform",8,0,2,0,0,0.,0.,0,N_rem(false,expDataFileAnnotated)},
                {"num_config_variables",0x29,0,3,0,0,0.,0.,0,N_rem(sizet,numExpConfigVars)},
                {"num_experiments",0x29,0,1,0,0,0.,0.,0,N_rem(sizet,numExperiments)},
                {"sigma_data_file",11,0,8,0,0,0.,0.,0,N_rem(str,sigmaDataFileName)},
                {"sigma_type",0x80f,0,4,0,0,0.,0.,0,N_rem(strL,sigmaType)}
                }
```

KeyWord kw_298[5] [static]

Initial value:

```
= {
                {"coordinate_data_file",11,0,3,0,0,0.,0.,0,N_rem(str,coordDataFileName)},
                {"coordinate_list",14,0,3,0,0,0.,0.,0,N_rem(RealL,coordsPerField)},
                {"field_data",8,9,4,0,kw_297},
                {"lengths",13,0,1,1,0,0.,0.,0,N_rem(ivec,fieldLengths)},
                {"num_coordinates_per_field",13,0,2,0,0,0.,0.,0,N_rem(ivec,numCoordsPerField)}
                }
```

KeyWord kw_299[6] [static]

Initial value:

```
= {
                {"nonlinear_equality_scale_types",0x807,0,2,0,0,0.,0.,3,N_rem(strL,nonlinearEqScaleTypes)},
                {"nonlinear_equality_scales",0x806,0,3,0,0,0.,0.,3,N_rem(RealDL,nonlinearEqScales)},
                {"nonlinear_equality_targets",6,0,1,0,0,0.,0.,3,N_rem(RealDL,nonlinearEqTargets)},
                {"scale_types",0x80f,0,2,0,0,0.,0.,0,N_rem(strL,nonlinearEqScaleTypes)},
                {"scales",0x80e,0,3,0,0,0.,0.,0,N_rem(RealDL,nonlinearEqScales)},
                {"targets",14,0,1,0,0,0.,0.,0,N_rem(RealDL,nonlinearEqTargets)}
                }
```

KeyWord kw_300[8] [static]

Initial value:

```
= {
                {"lower_bounds",14,0,1,0,0,0.,0.,0,N_rem(RealDL,nonlinearIneqLowerBnds)},
                {"nonlinear_inequality_lower_bounds",6,0,1,0,0,0.,0.,-1,N_rem(RealDL,nonlinearIneqLowerBnds
      )},
                {"nonlinear_inequality_scale_types",0x807,0,3,0,0,0.,0.,3,N_rem(strL,
      nonlinearIneqScaleTypes)},
                {"nonlinear_inequality_scales",0x806,0,4,0,0,0.,0.,3,N_rem(RealDL,nonlinearIneqScales)},
                {"nonlinear_inequality_upper_bounds",6,0,2,0,0,0.,0.,3,N_rem(RealDL,nonlinearIneqUpperBnds)
      },
                {"scale_types",0x80f,0,3,0,0,0.,0.,0,N_rem(strL,nonlinearIneqScaleTypes)},
                {"scales",0x80e,0,4,0,0,0.,0.,0,N_rem(RealDL,nonlinearIneqScales)},
                {"upper_bounds",14,0,2,0,0,0.,0.,0,N_rem(RealDL,nonlinearIneqUpperBnds)}
                }
```

KeyWord kw_301[19] [static]

Initial value:

```
= {
                {"calibration_data_file",11,6,5,0,kw_296,0.,0.,0,N_rem(str,expDataFileName)},
                {"calibration_term_scale_types",0x807,0,1,0,0,0.,0.,14,N_rem(strL,primaryRespFnScaleTypes)}
      ,
                {"calibration_term_scales",0x806,0,2,0,0,0.,0.,14,N_rem(RealDL,primaryRespFnScales)},
                {"calibration_weights",6,0,3,0,0,0.,0.,15,N_rem(RealDL,primaryRespFnWeights)},
                {"field_calibration_terms",0x29,5,8,0,kw_298,0.,0.,0,N_rem(sizet,numFieldLeastSqTerms)},
                {"least_squares_data_file",3,6,5,0,kw_296,0.,0.,-5,N_rem(str,expDataFileName)},
                {"least_squares_term_scale_types",0x807,0,1,0,0,0.,0.,9,N_rem(strL,primaryRespFnScaleTypes)
      },
                {"least_squares_term_scales",0x806,0,2,0,0,0.,0.,9,N_rem(RealDL,primaryRespFnScales)},
                {"least_squares_weights",6,0,3,0,0,0.,0.,10,N_rem(RealDL,primaryRespFnWeights)},
                {"nonlinear_equality_constraints",0x29,6,7,0,kw_299,0.,0.,0,N_rem(sizet,
      numNonlinearEqConstraints)},
                {"nonlinear_inequality_constraints",0x29,8,6,0,kw_300,0.,0.,0,N_rem(sizet,
      numNonlinearIneqConstraints)},
                {"num_field_calibration_terms",0x21,5,8,0,kw_298,0.,0.,-7,N_rem(sizet,numFieldLeastSqTerms)
      },
                {"num_nonlinear_equality_constraints",0x21,6,7,0,kw_299,0.,0.,-3,N_rem(sizet,
      numNonlinearEqConstraints)},
                {"num_nonlinear_inequality_constraints",0x21,8,6,0,kw_300,0.,0.,-3,N_rem(sizet,
      numNonlinearIneqConstraints)},
                {"num_scalar_calibration_terms",0x21,0,4,0,0,0.,0.,3,N_rem(sizet,numScalarLeastSqTerms)},
                {"primary_scale_types",0x80f,0,1,0,0,0.,0.,0,N_rem(strL,primaryRespFnScaleTypes)},
                {"primary_scales",0x80e,0,2,0,0,0.,0.,0,N_rem(RealDL,primaryRespFnScales)},
                {"scalar_calibration_terms",0x29,0,4,0,0,0.,0.,0,N_rem(sizet,numScalarLeastSqTerms)},
                {"weights",14,0,3,0,0,0.,0.,0,N_rem(RealDL,primaryRespFnWeights)}
                }
```

KeyWord kw_302[4] [static]

Initial value:

```
= {
                {"absolute",8,0,2,0,0,0.,0.,0,N_rem(lit,fdGradStepType_absolute)},
                {"bounds",8,0,2,0,0,0.,0.,0,N_rem(lit,fdGradStepType_bounds)},
                {"ignore_bounds",8,0,1,0,0,0.,0.,0,N_rem(true,ignoreBounds)},
                {"relative",8,0,2,0,0,0.,0.,0,N_rem(lit,fdGradStepType_relative)}
                }
```

KeyWord kw_303[8] [static]

Initial value:

```
= {
                {"central",8,0,4,0,0,0.,0.,0,N_rem(lit,intervalType_central)},
                {"dakota",8,4,2,0,kw_302,0.,0.,0,N_rem(lit,methodSource_dakota)},
                {"fd_gradient_step_size",6,0,5,0,0,0.,0.,1,N_rem(RealL,fdGradStepSize)},
                {"fd_step_size",14,0,5,0,0,0.,0.,0,N_rem(RealL,fdGradStepSize)},
                {"forward",8,0,4,0,0,0.,0.,0,N_rem(lit,intervalType_forward)},
                {"interval_type",8,0,3},
                {"method_source",8,0,1},
                {"vendor",8,0,2,0,0,0.,0.,0,N_rem(lit,methodSource_vendor)}
                }
```

KeyWord kw_304[3] [static]

Initial value:

```
= {
                {0,0,8,0,0,kw_303},
                {"id_analytic_gradients",13,0,2,2,0,0.,0.,0,N_rem(intset,idAnalyticGrads)},
                {"id_numerical_gradients",13,0,1,1,0,0.,0.,0,N_rem(intset,idNumericalGrads)}
                }
```

KeyWord kw_305[2] [static]

Initial value:

```
= {
                {"fd_hessian_step_size",6,0,1,0,0,0.,0.,1,N_rem(RealL,fdHessStepSize)},
                {"fd_step_size",14,0,1,0,0,0.,0.,0,N_rem(RealL,fdHessStepSize)}
                }
```

KeyWord kw_306[1] [static]

Initial value:

```
= {
                {"damped",8,0,1,0,0,0.,0.,0,N_rem(lit,quasiHessianType_damped_bfgs)}
                }
```

KeyWord kw_307[2] [static]

Initial value:

```
= {
                {"bfgs",8,1,1,1,kw_306,0.,0.,0,N_rem(lit,quasiHessianType_bfgs)},
                {"sr1",8,0,1,1,0,0.,0.,0,N_rem(lit,quasiHessianType_sr1)}
                }
```

KeyWord kw_308[8] [static]

Initial value:

```
= {
                {"absolute",8,0,2,0,0,0.,0.,0,N_rem(lit,fdHessStepType_absolute)},
                {"bounds",8,0,2,0,0,0.,0.,0,N_rem(lit,fdHessStepType_bounds)},
                {"central",8,0,3,0,0,0.,0.,0,N_rem(true,centralHess)},
                {"forward",8,0,3,0,0,0.,0.,0,N_rem(false,centralHess)},
                {"id_analytic_hessians",13,0,5,0,0,0.,0.,0,N_rem(intset,idAnalyticHessians)},
                {"id_numerical_hessians",13,2,1,0,kw_305,0.,0.,0,N_rem(intset,idNumericalHessians)},
                {"id_quasi_hessians",13,2,4,0,kw_307,0.,0.,0,N_rem(intset,idQuasiHessians)},
                {"relative",8,0,2,0,0,0.,0.,0,N_rem(lit,fdHessStepType_relative)}
                }
```

KeyWord kw_309[4] [static]

Initial value:

```
= {
                {"coordinate_data_file",11,0,3,0,0,0.,0.,0,N_rem(str,coordDataFileName)},
                {"coordinate_list",14,0,3,0,0,0.,0.,0,N_rem(RealL,coordsPerField)},
                {"lengths",13,0,1,1,0,0.,0.,0,N_rem(ivec,fieldLengths)},
                {"num_coordinates_per_field",13,0,2,0,0,0.,0.,0,N_rem(ivec,numCoordsPerField)}
                }
```

KeyWord kw_310[6] [static]

Initial value:

```
= {
                {"nonlinear_equality_scale_types",0x807,0,2,0,0,0.,0.,3,N_rem(strL,nonlinearEqScaleTypes)},
                {"nonlinear_equality_scales",0x806,0,3,0,0,0.,0.,3,N_rem(RealDL,nonlinearEqScales)},
                {"nonlinear_equality_targets",6,0,1,0,0,0.,0.,3,N_rem(RealDL,nonlinearEqTargets)},
                {"scale_types",0x80f,0,2,0,0,0.,0.,0,N_rem(strL,nonlinearEqScaleTypes)},
                {"scales",0x80e,0,3,0,0,0.,0.,0,N_rem(RealDL,nonlinearEqScales)},
                {"targets",14,0,1,0,0,0.,0.,0,N_rem(RealDL,nonlinearEqTargets)}
                }
```

KeyWord kw_311[8] [static]

Initial value:

```
= {
                {"lower_bounds",14,0,1,0,0,0.,0.,0,N_rem(RealDL,nonlinearIneqLowerBnds)},
                {"nonlinear_inequality_lower_bounds",6,0,1,0,0,0.,0.,-1,N_rem(RealDL,nonlinearIneqLowerBnds
        )},
                {"nonlinear_inequality_scale_types",0x807,0,3,0,0,0.,0.,3,N_rem(strL,
        nonlinearIneqScaleTypes)},
                {"nonlinear_inequality_scales",0x806,0,4,0,0,0.,0.,3,N_rem(RealDL,nonlinearIneqScales)},
                {"nonlinear_inequality_upper_bounds",6,0,2,0,0,0.,0.,3,N_rem(RealDL,nonlinearIneqUpperBnds)
        },
                {"scale_types",0x80f,0,3,0,0,0.,0.,0,N_rem(strL,nonlinearIneqScaleTypes)},
                {"scales",0x80e,0,4,0,0,0.,0.,0,N_rem(RealDL,nonlinearIneqScales)},
                {"upper_bounds",14,0,2,0,0,0.,0.,0,N_rem(RealDL,nonlinearIneqUpperBnds)}
                }
```

KeyWord kw_312[15] [static]

Initial value:

```
= {
                {"field_objectives",0x29,4,8,0,kw_309,0.,0.,0,N_rem(sizet,numFieldObjectiveFunctions)},
                {"multi_objective_weights",6,0,4,0,0,0.,0.,13,N_rem(RealDL,primaryRespFnWeights)},
                {"nonlinear_equality_constraints",0x29,6,6,0,kw_310,0.,0.,0,N_rem(sizet,
      numNonlinearEqConstraints)},
                {"nonlinear_inequality_constraints",0x29,8,5,0,kw_311,0.,0.,0,N_rem(sizet,
      numNonlinearIneqConstraints)},
                {"num_field_objectives",0x21,4,8,0,kw_309,0.,0.,-4,N_rem(sizet,numFieldObjectiveFunctions)}
      ,
                {"num_nonlinear_equality_constraints",0x21,6,6,0,kw_310,0.,0.,-3,N_rem(sizet,
      numNonlinearEqConstraints)},
                {"num_nonlinear_inequality_constraints",0x21,8,5,0,kw_311,0.,0.,-3,N_rem(sizet,
      numNonlinearIneqConstraints)},
                {"num_scalar_objectives",0x21,0,7,0,0,0.,0.,5,N_rem(sizet,numScalarObjectiveFunctions)},
                {"objective_function_scale_types",0x807,0,2,0,0,0.,0.,2,N_rem(strL,primaryRespFnScaleTypes)
      },
                {"objective_function_scales",0x806,0,3,0,0,0.,0.,2,N_rem(RealDL,primaryRespFnScales)},
                {"primary_scale_types",0x80f,0,2,0,0,0.,0.,0,N_rem(strL,primaryRespFnScaleTypes)},
                {"primary_scales",0x80e,0,3,0,0,0.,0.,0,N_rem(RealDL,primaryRespFnScales)},
                {"scalar_objectives",0x29,0,7,0,0,0.,0.,0,N_rem(sizet,numScalarObjectiveFunctions)},
                {"sense",0x80f,0,1,0,0,0.,0.,0,N_rem(strL,primaryRespFnSense)},
                {"weights",14,0,4,0,0,0.,0.,0,N_rem(RealDL,primaryRespFnWeights)}
                }
```

KeyWord kw_313[4] [static]

Initial value:

```
= {
                {"coordinate_data_file",11,0,3,0,0,0.,0.,0,N_rem(str,coordDataFileName)},
                {"coordinate_list",14,0,3,0,0,0.,0.,0,N_rem(RealL,coordsPerField)},
                {"lengths",13,0,1,1,0,0.,0.,0,N_rem(ivec,fieldLengths)},
                {"num_coordinates_per_field",13,0,2,0,0,0.,0.,0,N_rem(ivec,numCoordsPerField)}
                }
```

KeyWord kw_314[4] [static]

Initial value:

```
= {
                {"field_responses",0x29,4,2,0,kw_313,0.,0.,0,N_rem(sizet,numFieldResponseFunctions)},
                {"num_field_responses",0x21,4,2,0,kw_313,0.,0.,-1,N_rem(sizet,numFieldResponseFunctions)},
                {"num_scalar_responses",0x21,0,1,0,0,0.,0.,1,N_rem(sizet,numScalarResponseFunctions)},
                {"scalar_responses",0x29,0,1,0,0,0.,0.,0,N_rem(sizet,numScalarResponseFunctions)}
                }
```

KeyWord kw_315[7] [static]

Initial value:

```
= {
                {"absolute",8,0,2,0,0,0.,0.,0,N_rem(lit,fdHessStepType_absolute)},
                {"bounds",8,0,2,0,0,0.,0.,0,N_rem(lit,fdHessStepType_bounds)},
                {"central",8,0,3,0,0,0.,0.,0,N_rem(true,centralHess)},
                {"fd_hessian_step_size",6,0,1,0,0,0.,0.,1,N_rem(RealL,fdHessStepSize)},
                {"fd_step_size",14,0,1,0,0,0.,0.,0,N_rem(RealL,fdHessStepSize)},
                {"forward",8,0,3,0,0,0.,0.,0,N_rem(false,centralHess)},
                {"relative",8,0,2,0,0,0.,0.,0,N_rem(lit,fdHessStepType_relative)}
                }
```

KeyWord kw_316[1] [static]

Initial value:

```
= {
                {"damped",8,0,1,0,0,0.,0.,0,N_rem(lit,quasiHessianType_damped_bfgs)}
                }
```

KeyWord kw_317[2] [static]

Initial value:

```
= {
                {"bfgs",8,1,1,1,kw_316,0.,0.,0,N_rem(lit,quasiHessianType_bfgs)},
                {"sr1",8,0,1,1,0,0.,0.,0,N_rem(lit,quasiHessianType_sr1)}
                }
```

KeyWord kw_318[19] [static]

Initial value:

```
= {
                {"analytic_gradients",8,0,4,2,0,0.,0.,0,N_rem(lit,gradientType_analytic)},
                {"analytic_hessians",8,0,5,3,0,0.,0.,0,N_rem(lit,hessianType_analytic)},
                {"calibration_terms",0x29,19,3,1,kw_301,0.,0.,0,N_rem(sizet,numLeastSqTerms)},
                {"descriptors",15,0,2,0,0,0.,0.,0,N_rem(strL,responseLabels)},
                {"id_responses",11,0,1,0,0,0.,0.,0,N_rem(str,idResponses)},
                {"least_squares_terms",0x21,19,3,1,kw_301,0.,0.,-3,N_rem(sizet,numLeastSqTerms)},
                {"mixed_gradients",8,2,4,2,kw_304,0.,0.,0,N_rem(lit,gradientType_mixed)},
                {"mixed_hessians",8,8,5,3,kw_308,0.,0.,0,N_rem(lit,hessianType_mixed)},
                {"no_gradients",8,0,4,2,0,0.,0.,0,N_rem(lit,gradientType_none)},
                {"no_hessians",8,0,5,3,0,0.,0.,0,N_rem(lit,hessianType_none)},
                {"num_least_squares_terms",0x21,19,3,1,kw_301,0.,0.,-8,N_rem(sizet,numLeastSqTerms)},
                {"num_objective_functions",0x21,15,3,1,kw_312,0.,0.,4,N_rem(sizet,numObjectiveFunctions)},
                {"num_response_functions",0x21,4,3,1,kw_314,0.,0.,6,N_rem(sizet,numResponseFunctions)},
                {"numerical_gradients",8,8,4,2,kw_303,0.,0.,0,N_rem(lit,gradientType_numerical)},
                {"numerical_hessians",8,7,5,3,kw_315,0.,0.,0,N_rem(lit,hessianType_numerical)},
                {"objective_functions",0x29,15,3,1,kw_312,0.,0.,0,N_rem(sizet,numObjectiveFunctions)},
                {"quasi_hessians",8,2,5,3,kw_317,0.,0.,0,N_rem(lit,hessianType_quasi)},
                {"response_descriptors",7,0,2,0,0,0.,0.,-14,N_rem(strL,responseLabels)},
                {"response_functions",0x29,4,3,1,kw_314,0.,0.,0,N_rem(sizet,numResponseFunctions)}
                }
```

KeyWord kw_319[6] [static]

Initial value:

```
= {
                {"aleatory",8,0,1,1,0,0.,0.,0,N_vam(type,varsView_ALEATORY_UNCERTAIN_VIEW)},
                {"all",8,0,1,1,0,0.,0.,0,N_vam(type,varsView_ALL_VIEW)},
                {"design",8,0,1,1,0,0.,0.,0,N_vam(type,varsView_DESIGN_VIEW)},
                {"epistemic",8,0,1,1,0,0.,0.,0,N_vam(type,varsView_EPISTEMIC_UNCERTAIN_VIEW)},
                {"state",8,0,1,1,0,0.,0.,0,N_vam(type,varsView_STATE_VIEW)},
                {"uncertain",8,0,1,1,0,0.,0.,0,N_vam(type,varsView_UNCERTAIN_VIEW)}
                }
```

KeyWord kw_320[11] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,0,0.,0.,0,N_vam(RealLb,betaUncAlphas)},
                {"betas",14,0,2,2,0,0.,0.,0,N_vam(RealLb,betaUncBetas)},
                {"buv_alphas",6,0,1,1,0,0.,0.,-2,N_vam(RealLb,betaUncAlphas)},
                {"buv_betas",6,0,2,2,0,0.,0.,-2,N_vam(RealLb,betaUncBetas)},
                {"buv_descriptors",7,0,6,0,0,0.,0.,3,N_vae(caulbl,CAUVar_beta)},
                {"buv_lower_bounds",6,0,3,3,0,0.,0.,4,N_vam(rvec,betaUncLowerBnds)},
                {"buv_upper_bounds",6,0,4,4,0,0.,0.,4,N_vam(rvec,betaUncUpperBnds)},
                {"descriptors",15,0,6,0,0,0.,0.,0,N_vae(caulbl,CAUVar_beta)},
                {"initial_point",14,0,5,0,0,0.,0.,0,N_vam(rvec,betaUncVars)},
                {"lower_bounds",14,0,3,3,0,0.,0.,0,N_vam(rvec,betaUncLowerBnds)},
                {"upper_bounds",14,0,4,4,0,0.,0.,0,N_vam(rvec,betaUncUpperBnds)}
                }
```

KeyWord kw_321[5] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vae(dauilbl,DAUIVar_binomial)},
                {"initial_point",13,0,3,0,0,0.,0.,0,N_vam(IntLb,binomialUncVars)},
                {"num_trials",13,0,2,2,0,0.,0.,0,N_vam(IntLb,binomialUncNumTrials)},
                {"prob_per_trial",6,0,1,1,0,0.,0.,1,N_vam(rvec,binomialUncProbPerTrial)},
                {"probability_per_trial",14,0,1,1,0,0.,0.,0,N_vam(rvec,binomialUncProbPerTrial)}
                }
```

KeyWord kw_322[12] [static]

Initial value:

```
= {
                {"cdv_descriptors",7,0,6,0,0,0.,0.,6,N_vam(strL,continuousDesignLabels)},
                {"cdv_initial_point",6,0,1,0,0,0.,0.,6,N_vam(rvec,continuousDesignVars)},
                {"cdv_lower_bounds",6,0,2,0,0,0.,0.,6,N_vam(rvec,continuousDesignLowerBnds)},
                {"cdv_scale_types",0x807,0,4,0,0,0.,0.,6,N_vam(strL,continuousDesignScaleTypes)},
                {"cdv_scales",0x806,0,5,0,0,0.,0.,6,N_vam(rvec,continuousDesignScales)},
                {"cdv_upper_bounds",6,0,3,0,0,0.,0.,6,N_vam(rvec,continuousDesignUpperBnds)},
                {"descriptors",15,0,6,0,0,0.,0.,0,N_vam(strL,continuousDesignLabels)},
                {"initial_point",14,0,1,0,0,0.,0.,0,N_vam(rvec,continuousDesignVars)},
                {"lower_bounds",14,0,2,0,0,0.,0.,0,N_vam(rvec,continuousDesignLowerBnds)},
                {"scale_types",0x80f,0,4,0,0,0.,0.,0,N_vam(strL,continuousDesignScaleTypes)},
                {"scales",0x80e,0,5,0,0,0.,0.,0,N_vam(rvec,continuousDesignScales)},
                {"upper_bounds",14,0,3,0,0,0.,0.,0,N_vam(rvec,continuousDesignUpperBnds)}
                }
```

KeyWord kw_323[10] [static]

Initial value:

```
= {
                {"descriptors",15,0,6,0,0,0.,0.,0,N_vae(ceulbl,CEUVar_interval)},
                {"initial_point",14,0,5,0,0,0.,0.,0,N_vam(rvec,continuousIntervalUncVars)},
                {"interval_probabilities",14,0,2,0,0,0.,0.,0,N_vam(newrvec,Var_Info_CIp)},
                {"interval_probs",6,0,2,0,0,0.,0.,-1,N_vam(newrvec,Var_Info_CIp)},
                {"iuv_descriptors",7,0,6,0,0,0.,0.,-4,N_vae(ceulbl,CEUVar_interval)},
                {"iuv_interval_probs",6,0,2,0,0,0.,0.,-3,N_vam(newrvec,Var_Info_CIp)},
                {"iuv_num_intervals",5,0,1,0,0,0.,0.,2,N_vam(newiarray,Var_Info_nCI)},
                {"lower_bounds",14,0,3,1,0,0.,0.,0,N_vam(newrvec,Var_Info_CIlb)},
                {"num_intervals",13,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nCI)},
                {"upper_bounds",14,0,4,2,0,0.,0.,0,N_vam(newrvec,Var_Info_CIub)}
                }
```

KeyWord kw_324[8] [static]

Initial value:

```
= {
                {"csv_descriptors",7,0,4,0,0,0.,0.,4,N_vam(strL,continuousStateLabels)},
                {"csv_initial_state",6,0,1,0,0,0.,0.,4,N_vam(rvec,continuousStateVars)},
                {"csv_lower_bounds",6,0,2,0,0,0.,0.,4,N_vam(rvec,continuousStateLowerBnds)},
                {"csv_upper_bounds",6,0,3,0,0,0.,0.,4,N_vam(rvec,continuousStateUpperBnds)},
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vam(strL,continuousStateLabels)},
                {"initial_state",14,0,1,0,0,0.,0.,0,N_vam(rvec,continuousStateVars)},
                {"lower_bounds",14,0,2,0,0,0.,0.,0,N_vam(rvec,continuousStateLowerBnds)},
                {"upper_bounds",14,0,3,0,0,0.,0.,0,N_vam(rvec,continuousStateUpperBnds)}
                }
```

KeyWord kw_325[8] [static]

Initial value:

```
= {
                {"ddv_descriptors",7,0,4,0,0,0.,0.,4,N_vam(strL,discreteDesignRangeLabels)},
                {"ddv_initial_point",5,0,1,0,0,0.,0.,4,N_vam(ivec,discreteDesignRangeVars)},
                {"ddv_lower_bounds",5,0,2,0,0,0.,0.,4,N_vam(ivec,discreteDesignRangeLowerBnds)},
                {"ddv_upper_bounds",5,0,3,0,0,0.,0.,4,N_vam(ivec,discreteDesignRangeUpperBnds)},
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vam(strL,discreteDesignRangeLabels)},
                {"initial_point",13,0,1,0,0,0.,0.,0,N_vam(ivec,discreteDesignRangeVars)},
                {"lower_bounds",13,0,2,0,0,0.,0.,0,N_vam(ivec,discreteDesignRangeLowerBnds)},
                {"upper_bounds",13,0,3,0,0,0.,0.,0,N_vam(ivec,discreteDesignRangeUpperBnds)}
                }
```

KeyWord kw_326[7] [static]

Initial value:

```
= {
                {"categorical",15,0,3,0,0,0.,0.,0,N_vam(categorical,discreteDesignSetIntCat)},
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vam(strL,discreteDesignSetIntLabels)},
                {"elements",13,0,2,1,0,0.,0.,0,N_vam(newivec,Var_Info_ddsi)},
                {"elements_per_variable",0x80d,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nddsi)},
                {"initial_point",13,0,4,0,0,0.,0.,0,N_vam(ivec,discreteDesignSetIntVars)},
                {"num_set_values",0x805,0,1,0,0,0.,0.,-2,N_vam(newiarray,Var_Info_nddsi)},
                {"set_values",5,0,2,1,0,0.,0.,-4,N_vam(newivec,Var_Info_ddsi)}
                }
```

KeyWord kw_327[7] [static]

Initial value:

```
= {
                {"categorical",15,0,3,0,0,0.,0.,0,N_vam(categorical,discreteDesignSetRealCat)},
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vam(strL,discreteDesignSetRealLabels)},
                {"elements",14,0,2,1,0,0.,0.,0,N_vam(newrvec,Var_Info_ddsr)},
                {"elements_per_variable",0x80d,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nddsr)},
                {"initial_point",14,0,4,0,0,0.,0.,0,N_vam(rvec,discreteDesignSetRealVars)},
                {"num_set_values",0x805,0,1,0,0,0.,0.,-2,N_vam(newiarray,Var_Info_nddsr)},
                {"set_values",6,0,2,1,0,0.,0.,-4,N_vam(newrvec,Var_Info_ddsr)}
                }
```

KeyWord kw_328[6] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vam(strL,discreteDesignSetStrLabels)},
                {"elements",15,0,2,1,0,0.,0.,0,N_vam(newsarray,Var_Info_ddss)},
                {"elements_per_variable",0x80d,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nddss)},
                {"initial_point",15,0,3,0,0,0.,0.,0,N_vam(strL,discreteDesignSetStrVars)},
                {"num_set_values",0x805,0,1,0,0,0.,0.,-2,N_vam(newiarray,Var_Info_nddss)},
                {"set_values",7,0,2,1,0,0.,0.,-4,N_vam(newsarray,Var_Info_ddss)}
                }
```

KeyWord kw_329[3] [static]

Initial value:

```
= {
                {"integer",0x19,7,1,0,kw_326,0.,0.,0,N_vam(pintz,numDiscreteDesSetIntVars)},
                {"real",0x19,7,3,0,kw_327,0.,0.,0,N_vam(pintz,numDiscreteDesSetRealVars)},
                {"string",0x19,6,2,0,kw_328,0.,0.,0,N_vam(pintz,numDiscreteDesSetStrVars)}
                }
```

KeyWord kw_330[9] [static]

Initial value:

```
= {
                {"descriptors",15,0,6,0,0,0.,0.,0,N_vae(deuilbl,DEUIVar_interval)},
                {"initial_point",13,0,5,0,0,0.,0.,0,N_vam(ivec,discreteIntervalUncVars)},
                {"interval_probabilities",14,0,2,0,0,0.,0.,0,N_vam(newrvec,Var_Info_DIp)},
                {"interval_probs",6,0,2,0,0,0.,0.,-1,N_vam(newrvec,Var_Info_DIp)},
                {"lower_bounds",13,0,3,1,0,0.,0.,0,N_vam(newivec,Var_Info_DIlb)},
                {"num_intervals",13,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nDI)},
                {"range_probabilities",6,0,2,0,0,0.,0.,-4,N_vam(newrvec,Var_Info_DIp)},
                {"range_probs",6,0,2,0,0,0.,0.,-5,N_vam(newrvec,Var_Info_DIp)},
                {"upper_bounds",13,0,4,2,0,0.,0.,0,N_vam(newivec,Var_Info_DIub)}
                }
```

KeyWord kw_331[8] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vam(strL,discreteStateRangeLabels)},
                {"dsv_descriptors",7,0,4,0,0,0.,0.,-1,N_vam(strL,discreteStateRangeLabels)},
                {"dsv_initial_state",5,0,1,0,0,0.,0.,3,N_vam(ivec,discreteStateRangeVars)},
                {"dsv_lower_bounds",5,0,2,0,0,0.,0.,3,N_vam(ivec,discreteStateRangeLowerBnds)},
                {"dsv_upper_bounds",5,0,3,0,0,0.,0.,3,N_vam(ivec,discreteStateRangeUpperBnds)},
                {"initial_state",13,0,1,0,0,0.,0.,0,N_vam(ivec,discreteStateRangeVars)},
                {"lower_bounds",13,0,2,0,0,0.,0.,0,N_vam(ivec,discreteStateRangeLowerBnds)},
                {"upper_bounds",13,0,3,0,0,0.,0.,0,N_vam(ivec,discreteStateRangeUpperBnds)}
                }
```

KeyWord kw_332[7] [static]

Initial value:

```
= {
                {"categorical",15,0,3,0,0,0.,0.,0,N_vam(categorical,discreteStateSetIntCat)},
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vam(strL,discreteStateSetIntLabels)},
                {"elements",13,0,2,1,0,0.,0.,0,N_vam(newivec,Var_Info_dssi)},
                {"elements_per_variable",0x80d,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_ndssi)},
                {"initial_state",13,0,4,0,0,0.,0.,0,N_vam(ivec,discreteStateSetIntVars)},
                {"num_set_values",0x805,0,1,0,0,0.,0.,-2,N_vam(newiarray,Var_Info_ndssi)},
                {"set_values",5,0,2,1,0,0.,0.,-4,N_vam(newivec,Var_Info_dssi)}
                }
```

KeyWord kw_333[7] [static]

Initial value:

```
= {
                {"categorical",15,0,3,0,0,0.,0.,0,N_vam(categorical,discreteStateSetRealCat)},
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vam(strL,discreteStateSetRealLabels)},
                {"elements",14,0,2,1,0,0.,0.,0,N_vam(newrvec,Var_Info_dssr)},
                {"elements_per_variable",0x80d,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_ndssr)},
                {"initial_state",14,0,4,0,0,0.,0.,0,N_vam(rvec,discreteStateSetRealVars)},
                {"num_set_values",0x805,0,1,0,0,0.,0.,-2,N_vam(newiarray,Var_Info_ndssr)},
                {"set_values",6,0,2,1,0,0.,0.,-4,N_vam(newrvec,Var_Info_dssr)}
                }
```

KeyWord kw_334[6] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vam(strL,discreteStateSetStrLabels)},
                {"elements",15,0,2,1,0,0.,0.,0,N_vam(newsarray,Var_Info_dsss)},
                {"elements_per_variable",0x80d,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_ndsss)},
                {"initial_state",15,0,3,0,0,0.,0.,0,N_vam(strL,discreteStateSetStrVars)},
                {"num_set_values",0x805,0,1,0,0,0.,0.,-2,N_vam(newiarray,Var_Info_ndsss)},
                {"set_values",7,0,2,1,0,0.,0.,-4,N_vam(newsarray,Var_Info_dsss)}
                }
```

KeyWord kw_335[3] [static]

Initial value:

```
= {
                {"integer",0x19,7,1,0,kw_332,0.,0.,0,N_vam(pintz,numDiscreteStateSetIntVars)},
                {"real",0x19,7,3,0,kw_333,0.,0.,0,N_vam(pintz,numDiscreteStateSetRealVars)},
                {"string",0x19,6,2,0,kw_334,0.,0.,0,N_vam(pintz,numDiscreteStateSetStrVars)}
                }
```

KeyWord kw_336[9] [static]

Initial value:

```
= {
                {"categorical",15,0,4,0,0,0.,0.,0,N_vam(categorical,discreteUncSetIntCat)},
                {"descriptors",15,0,6,0,0,0.,0.,0,N_vae(deuilbl,DEUIVar_set_int)},
                {"elements",13,0,2,1,0,0.,0.,0,N_vam(newivec,Var_Info_dusi)},
                {"elements_per_variable",13,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_ndusi)},
                {"initial_point",13,0,5,0,0,0.,0.,0,N_vam(ivec,discreteUncSetIntVars)},
                {"num_set_values",5,0,1,0,0,0.,0.,-2,N_vam(newiarray,Var_Info_ndusi)},
                {"set_probabilities",14,0,3,0,0,0.,0.,0,N_vam(newrvec,Var_Info_DSIp)},
                {"set_probs",6,0,3,0,0,0.,0.,-1,N_vam(newrvec,Var_Info_DSIp)},
                {"set_values",5,0,2,1,0,0.,0.,-6,N_vam(newivec,Var_Info_dusi)}
                }
```

KeyWord kw_337[9] [static]

Initial value:

```
= {
                {"categorical",15,0,4,0,0,0.,0.,0,N_vam(categorical,discreteUncSetRealCat)},
                {"descriptors",15,0,6,0,0,0.,0.,0,N_vae(deurlbl,DEURVar_set_real)},
                {"elements",14,0,2,1,0,0.,0.,0,N_vam(newrvec,Var_Info_dusr)},
                {"elements_per_variable",13,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_ndusr)},
                {"initial_point",14,0,5,0,0,0.,0.,0,N_vam(rvec,discreteUncSetRealVars)},
                {"num_set_values",5,0,1,0,0,0.,0.,-2,N_vam(newiarray,Var_Info_ndusr)},
                {"set_probabilities",14,0,3,0,0,0.,0.,0,N_vam(newrvec,Var_Info_DSRp)},
                {"set_probs",6,0,3,0,0,0.,0.,-1,N_vam(newrvec,Var_Info_DSRp)},
                {"set_values",6,0,2,1,0,0.,0.,-6,N_vam(newrvec,Var_Info_dusr)}
                }
```

KeyWord kw_338[8] [static]

Initial value:

```
= {
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vae(deuslbl,DEUSVar_set_str)},
                {"elements",15,0,2,1,0,0.,0.,0,N_vam(newsarray,Var_Info_duss)},
                {"elements_per_variable",13,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nduss)},
                {"initial_point",15,0,4,0,0,0.,0.,0,N_vam(strL,discreteUncSetStrVars)},
                {"num_set_values",5,0,1,0,0,0.,0.,-2,N_vam(newiarray,Var_Info_nduss)},
                {"set_probabilities",14,0,3,0,0,0.,0.,0,N_vam(newrvec,Var_Info_DSSp)},
                {"set_probs",6,0,3,0,0,0.,0.,-1,N_vam(newrvec,Var_Info_DSSp)},
                {"set_values",7,0,2,1,0,0.,0.,-6,N_vam(newsarray,Var_Info_duss)}
                }
```

KeyWord kw_339[3] [static]

Initial value:

```
= {
                {"integer",0x19,9,1,0,kw_336,0.,0.,0,N_vam(pintz,numDiscreteUncSetIntVars)},
                {"real",0x19,9,3,0,kw_337,0.,0.,0,N_vam(pintz,numDiscreteUncSetRealVars)},
                {"string",0x19,8,2,0,kw_338,0.,0.,0,N_vam(pintz,numDiscreteUncSetStrVars)}
                }
```

KeyWord kw_340[5] [static]

Initial value:

```
= {
                {"betas",14,0,1,1,0,0.,0.,0,N_vam(RealLb,exponentialUncBetas)},
                {"descriptors",15,0,3,0,0,0.,0.,0,N_vae(caulbl,CAUVar_exponential)},
                {"euv_betas",6,0,1,1,0,0.,0.,-2,N_vam(RealLb,exponentialUncBetas)},
                {"euv_descriptors",7,0,3,0,0,0.,0.,-2,N_vae(caulbl,CAUVar_exponential)},
                {"initial_point",14,0,2,0,0,0.,0.,0,N_vam(RealLb,exponentialUncVars)}
                }
```

KeyWord kw_341[7] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,0,0.,0.,0,N_vam(RealLb,frechetUncAlphas)},
                {"betas",14,0,2,2,0,0.,0.,0,N_vam(rvec,frechetUncBetas)},
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vae(caulbl,CAUVar_frechet)},
                {"fuv_alphas",6,0,1,1,0,0.,0.,-3,N_vam(RealLb,frechetUncAlphas)},
                {"fuv_betas",6,0,2,2,0,0.,0.,-3,N_vam(rvec,frechetUncBetas)},
                {"fuv_descriptors",7,0,4,0,0,0.,0.,-3,N_vae(caulbl,CAUVar_frechet)},
                {"initial_point",14,0,3,0,0,0.,0.,0,N_vam(rvec,frechetUncVars)}
                }
```

KeyWord kw_342[7] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,0,0.,0.,0,N_vam(RealLb,gammaUncAlphas)},
                {"betas",14,0,2,2,0,0.,0.,0,N_vam(RealLb,gammaUncBetas)},
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vae(caulbl,CAUVar_gamma)},
                {"gauv_alphas",6,0,1,1,0,0.,0.,-3,N_vam(RealLb,gammaUncAlphas)},
                {"gauv_betas",6,0,2,2,0,0.,0.,-3,N_vam(RealLb,gammaUncBetas)},
                {"gauv_descriptors",7,0,4,0,0,0.,0.,-3,N_vae(caulbl,CAUVar_gamma)},
                {"initial_point",14,0,3,0,0,0.,0.,0,N_vam(RealLb,gammaUncVars)}
                }
```

KeyWord kw_343[4] [static]

Initial value:

```
= {
                {"descriptors",15,0,3,0,0,0.,0.,0,N_vae(dauilbl,DAUIVar_geometric)},
                {"initial_point",13,0,2,0,0,0.,0.,0,N_vam(IntLb,geometricUncVars)},
                {"prob_per_trial",6,0,1,1,0,0.,0.,1,N_vam(rvec,geometricUncProbPerTrial)},
                {"probability_per_trial",14,0,1,1,0,0.,0.,0,N_vam(rvec,geometricUncProbPerTrial)}
                }
```

KeyWord kw_344[7] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,0,0.,0.,0,N_vam(RealLb,gumbelUncAlphas)},
                {"betas",14,0,2,2,0,0.,0.,0,N_vam(rvec,gumbelUncBetas)},
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vae(caulbl,CAUVar_gumbel)},
                {"guuv_alphas",6,0,1,1,0,0.,0.,-3,N_vam(RealLb,gumbelUncAlphas)},
                {"guuv_betas",6,0,2,2,0,0.,0.,-3,N_vam(rvec,gumbelUncBetas)},
                {"guuv_descriptors",7,0,4,0,0,0.,0.,-3,N_vae(caulbl,CAUVar_gumbel)},
                {"initial_point",14,0,3,0,0,0.,0.,0,N_vam(rvec,gumbelUncVars)}
                }
```

KeyWord kw_345[11] [static]

Initial value:

```
= {
                {"abscissas",14,0,2,1,0,0.,0.,0,N_vam(newrvec,Var_Info_hba)},
                {"counts",14,0,3,2,0,0.,0.,0,N_vam(newrvec,Var_Info_hbc)},
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vae(caulbl,CAUVar_histogram_bin)},
                {"huv_bin_abscissas",6,0,2,1,0,0.,0.,-3,N_vam(newrvec,Var_Info_hba)},
                {"huv_bin_counts",6,0,3,2,0,0.,0.,-3,N_vam(newrvec,Var_Info_hbc)},
                {"huv_bin_descriptors",7,0,5,0,0,0.,0.,-3,N_vae(caulbl,CAUVar_histogram_bin)},
                {"huv_bin_ordinates",6,0,3,2,0,0.,0.,3,N_vam(newrvec,Var_Info_hbo)},
                {"initial_point",14,0,4,0,0,0.,0.,0,N_vam(rvec,histogramBinUncVars)},
                {"num_pairs",5,0,1,0,0,0.,0.,2,N_vam(newiarray,Var_Info_nhbp)},
                {"ordinates",14,0,3,2,0,0.,0.,0,N_vam(newrvec,Var_Info_hbo)},
                {"pairs_per_variable",13,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nhbp)}
                }
```

KeyWord kw_346[6] [static]

Initial value:

```
= {
                {"abscissas",13,0,2,1,0,0.,0.,0,N_vam(newivec,Var_Info_hpia)},
                {"counts",14,0,3,2,0,0.,0.,0,N_vam(newrvec,Var_Info_hpic)},
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vae(dauilbl,DAUIVar_histogram_point_int)},
                {"initial_point",13,0,4,0,0,0.,0.,0,N_vam(ivec,histogramPointIntUncVars)},
                {"num_pairs",5,0,1,0,0,0.,0.,1,N_vam(newiarray,Var_Info_nhpip)},
                {"pairs_per_variable",13,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nhpip)}
                }
```

KeyWord kw_347[6] [static]

Initial value:

```
= {
                {"abscissas",14,0,2,1,0,0.,0.,0,N_vam(newrvec,Var_Info_hpra)},
                {"counts",14,0,3,2,0,0.,0.,0,N_vam(newrvec,Var_Info_hprc)},
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vae(daurlbl,DAURVar_histogram_point_real)},
                {"initial_point",14,0,4,0,0,0.,0.,0,N_vam(rvec,histogramPointRealUncVars)},
                {"num_pairs",5,0,1,0,0,0.,0.,1,N_vam(newiarray,Var_Info_nhprp)},
                {"pairs_per_variable",13,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nhprp)}
                }
```

KeyWord kw_348[6] [static]

Initial value:

```
= {
                {"abscissas",15,0,2,1,0,0.,0.,0,N_vam(newsarray,Var_Info_hpsa)},
                {"counts",14,0,3,2,0,0.,0.,0,N_vam(newrvec,Var_Info_hpsc)},
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vae(dauslbl,DAUSVar_histogram_point_str)},
                {"initial_point",15,0,4,0,0,0.,0.,0,N_vam(strL,histogramPointStrUncVars)},
                {"num_pairs",5,0,1,0,0,0.,0.,1,N_vam(newiarray,Var_Info_nhpsp)},
                {"pairs_per_variable",13,0,1,0,0,0.,0.,0,N_vam(newiarray,Var_Info_nhpsp)}
                }
```

KeyWord kw_349[3] [static]

Initial value:

```
= {
                {"integer",0x19,6,1,0,kw_346,0.,0.,0,N_vam(pintz,numHistogramPtIntUncVars)},
                {"real",0x19,6,3,0,kw_347,0.,0.,0,N_vam(pintz,numHistogramPtRealUncVars)},
                {"string",0x19,6,2,0,kw_348,0.,0.,0,N_vam(pintz,numHistogramPtStrUncVars)}
                }
```

KeyWord kw_350[5] [static]

Initial value:

```
= {
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vae(dauilbl,DAUIVar_hypergeometric)},
                {"initial_point",13,0,4,0,0,0.,0.,0,N_vam(IntLb,hyperGeomUncVars)},
                {"num_drawn",13,0,3,3,0,0.,0.,0,N_vam(IntLb,hyperGeomUncNumDrawn)},
                {"selected_population",13,0,2,2,0,0.,0.,0,N_vam(IntLb,hyperGeomUncSelectedPop)},
                {"total_population",13,0,1,1,0,0.,0.,0,N_vam(IntLb,hyperGeomUncTotalPop)}
                }
```

KeyWord kw_351[2] [static]

Initial value:

```
= {
                {"lnuv_zetas",6,0,1,1,0,0.,0.,1,N_vam(RealLb,lognormalUncZetas)},
                {"zetas",14,0,1,1,0,0.,0.,0,N_vam(RealLb,lognormalUncZetas)}
                }
```

KeyWord kw_352[4] [static]

Initial value:

```
= {
                {"error_factors",14,0,1,1,0,0.,0.,0,N_vam(RealLb,lognormalUncErrFacts)},
                {"lnuv_error_factors",6,0,1,1,0,0.,0.,-1,N_vam(RealLb,lognormalUncErrFacts)},
                {"lnuv_std_deviations",6,0,1,1,0,0.,0.,1,N_vam(RealLb,lognormalUncStdDevs)},
                {"std_deviations",14,0,1,1,0,0.,0.,0,N_vam(RealLb,lognormalUncStdDevs)}
                }
```

KeyWord kw_353[11] [static]

Initial value:

```
= {
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vae(caulbl,CAUVar_lognormal)},
                {"initial_point",14,0,4,0,0,0.,0.,0,N_vam(RealLb,lognormalUncVars)},
                {"lambdas",14,2,1,1,kw_351,0.,0.,0,N_vam(rvec,lognormalUncLambdas)},
                {"lnuv_descriptors",7,0,5,0,0,0.,0.,-3,N_vae(caulbl,CAUVar_lognormal)},
                {"lnuv_lambdas",6,2,1,1,kw_351,0.,0.,-2,N_vam(rvec,lognormalUncLambdas)},
                {"lnuv_lower_bounds",6,0,2,0,0,0.,0.,3,N_vam(RealLb,lognormalUncLowerBnds)},
                {"lnuv_means",6,4,1,1,kw_352,0.,0.,3,N_vam(RealLb,lognormalUncMeans)},
                {"lnuv_upper_bounds",6,0,3,0,0,0.,0.,3,N_vam(RealUb,lognormalUncUpperBnds)},
                {"lower_bounds",14,0,2,0,0,0.,0.,0,N_vam(RealLb,lognormalUncLowerBnds)},
                {"means",14,4,1,1,kw_352,0.,0.,0,N_vam(RealLb,lognormalUncMeans)},
                {"upper_bounds",14,0,3,0,0,0.,0.,0,N_vam(RealUb,lognormalUncUpperBnds)}
                }
```

KeyWord kw_354[7] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vae(caulbl,CAUVar_loguniform)},
                {"initial_point",14,0,3,0,0,0.,0.,0,N_vam(RealLb,loguniformUncVars)},
                {"lower_bounds",14,0,1,1,0,0.,0.,0,N_vam(RealLb,loguniformUncLowerBnds)},
                {"luuv_descriptors",7,0,4,0,0,0.,0.,-3,N_vae(caulbl,CAUVar_loguniform)},
                {"luuv_lower_bounds",6,0,1,1,0,0.,0.,-2,N_vam(RealLb,loguniformUncLowerBnds)},
                {"luuv_upper_bounds",6,0,2,2,0,0.,0.,1,N_vam(RealUb,loguniformUncUpperBnds)},
                {"upper_bounds",14,0,2,2,0,0.,0.,0,N_vam(RealUb,loguniformUncUpperBnds)}
                }
```

KeyWord kw_355[5] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vae(dauilbl,DAUIVar_negative_binomial)},
                {"initial_point",13,0,3,0,0,0.,0.,0,N_vam(IntLb,negBinomialUncVars)},
                {"num_trials",13,0,2,2,0,0.,0.,0,N_vam(IntLb,negBinomialUncNumTrials)},
                {"prob_per_trial",6,0,1,1,0,0.,0.,1,N_vam(rvec,negBinomialUncProbPerTrial)},
                {"probability_per_trial",14,0,1,1,0,0.,0.,0,N_vam(rvec,negBinomialUncProbPerTrial)}
                }
```

KeyWord kw_356[11] [static]

Initial value:

```
= {
                {"descriptors",15,0,6,0,0,0.,0.,0,N_vae(caulbl,CAUVar_normal)},
                {"initial_point",14,0,5,0,0,0.,0.,0,N_vam(rvec,normalUncVars)},
                {"lower_bounds",14,0,3,0,0,0.,0.,0,N_vam(rvec,normalUncLowerBnds)},
                {"means",14,0,1,1,0,0.,0.,0,N_vam(rvec,normalUncMeans)},
                {"nuv_descriptors",7,0,6,0,0,0.,0.,-4,N_vae(caulbl,CAUVar_normal)},
                {"nuv_lower_bounds",6,0,3,0,0,0.,0.,-3,N_vam(rvec,normalUncLowerBnds)},
                {"nuv_means",6,0,1,1,0,0.,0.,-3,N_vam(rvec,normalUncMeans)},
                {"nuv_std_deviations",6,0,2,2,0,0.,0.,2,N_vam(RealLb,normalUncStdDevs)},
                {"nuv_upper_bounds",6,0,4,0,0,0.,0.,2,N_vam(rvec,normalUncUpperBnds)},
                {"std_deviations",14,0,2,2,0,0.,0.,0,N_vam(RealLb,normalUncStdDevs)},
                {"upper_bounds",14,0,4,0,0,0.,0.,0,N_vam(rvec,normalUncUpperBnds)}
                }
```

KeyWord kw_357[3] [static]

Initial value:

```
= {
                {"descriptors",15,0,3,0,0,0.,0.,0,N_vae(dauilbl,DAUIVar_poisson)},
                {"initial_point",13,0,2,0,0,0.,0.,0,N_vam(IntLb,poissonUncVars)},
                {"lambdas",14,0,1,1,0,0.,0.,0,N_vam(RealLb,poissonUncLambdas)}
                }
```

KeyWord kw_358[9] [static]

Initial value:

```
= {
                {"descriptors",15,0,5,0,0,0.,0.,0,N_vae(caulbl,CAUVar_triangular)},
                {"initial_point",14,0,4,0,0,0.,0.,0,N_vam(rvec,triangularUncVars)},
                {"lower_bounds",14,0,2,2,0,0.,0.,0,N_vam(RealLb,triangularUncLowerBnds)},
                {"modes",14,0,1,1,0,0.,0.,0,N_vam(rvec,triangularUncModes)},
                {"tuv_descriptors",7,0,5,0,0,0.,0.,-4,N_vae(caulbl,CAUVar_triangular)},
                {"tuv_lower_bounds",6,0,2,2,0,0.,0.,-3,N_vam(RealLb,triangularUncLowerBnds)},
                {"tuv_modes",6,0,1,1,0,0.,0.,-3,N_vam(rvec,triangularUncModes)},
                {"tuv_upper_bounds",6,0,3,3,0,0.,0.,1,N_vam(RealUb,triangularUncUpperBnds)},
                {"upper_bounds",14,0,3,3,0,0.,0.,0,N_vam(RealUb,triangularUncUpperBnds)}
                }
```

KeyWord kw_359[7] [static]

Initial value:

```
= {
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vae(caulbl,CAUVar_uniform)},
                {"initial_point",14,0,3,0,0,0.,0.,0,N_vam(rvec,uniformUncVars)},
                {"lower_bounds",14,0,1,1,0,0.,0.,0,N_vam(RealLb,uniformUncLowerBnds)},
                {"upper_bounds",14,0,2,2,0,0.,0.,0,N_vam(RealUb,uniformUncUpperBnds)},
                {"uuv_descriptors",7,0,4,0,0,0.,0.,-4,N_vae(caulbl,CAUVar_uniform)},
                {"uuv_lower_bounds",6,0,1,1,0,0.,0.,-3,N_vam(RealLb,uniformUncLowerBnds)},
                {"uuv_upper_bounds",6,0,2,2,0,0.,0.,-3,N_vam(RealUb,uniformUncUpperBnds)}
                }
```

KeyWord kw_360[7] [static]

Initial value:

```
= {
                {"alphas",14,0,1,1,0,0.,0.,0,N_vam(RealLb,weibullUncAlphas)},
                {"betas",14,0,2,2,0,0.,0.,0,N_vam(RealLb,weibullUncBetas)},
                {"descriptors",15,0,4,0,0,0.,0.,0,N_vae(caulbl,CAUVar_weibull)},
                {"initial_point",14,0,3,0,0,0.,0.,0,N_vam(RealLb,weibullUncVars)},
                {"wuv_alphas",6,0,1,1,0,0.,0.,-4,N_vam(RealLb,weibullUncAlphas)},
                {"wuv_betas",6,0,2,2,0,0.,0.,-4,N_vam(RealLb,weibullUncBetas)},
                {"wuv_descriptors",7,0,4,0,0,0.,0.,-4,N_vae(caulbl,CAUVar_weibull)}
                }
```

KeyWord kw_362[6] [static]

Initial value:

```
= {
                {"environment",0x108,15,1,1,kw_7,0.,0.,0,NIDRProblemDescDB::env_start},
                {"interface",0x308,9,5,5,kw_21,0.,0.,0,N_ifm3(start,0,stop)},
                {"method",0x308,90,2,2,kw_264,0.,0.,0,N_mdm3(start,0,stop)},
                {"model",8,7,3,3,kw_295,0.,0.,0,N_mom3(start,0,stop)},
                {"responses",0x308,19,6,6,kw_318,0.,0.,0,N_rem3(start,0,stop)},
                {"variables",0x308,34,4,4,kw_361,0.,0.,0,N_vam3(start,0,stop)}
                }
```

Var_uinfo CAUVLbl[CAUVar_Nkinds] [static]

Initial value:

```
= {
  VarLabelInfo(nuv_, NormalUnc),
  VarLabelInfo(lnuv_, LognormalUnc),
  VarLabelInfo(uuv_, UniformUnc),
  VarLabelInfo(luuv_, LoguniformUnc),
  VarLabelInfo(tuv_, TriangularUnc),
  VarLabelInfo(euv_, ExponentialUnc),
  VarLabelInfo(beuv_, BetaUnc),
  VarLabelInfo(gauv_, GammaUnc),
  VarLabelInfo(guuv_, GumbelUnc),
  VarLabelInfo(fuv_, FrechetUnc),
  VarLabelInfo(wuv_, WeibullUnc),
  VarLabelInfo(hbuv_, HistogramBinUnc)
}
```

Var_uinfo DAUIVLbl[DAUIVar_Nkinds] [static]

Initial value:

```
= {
  VarLabelInfo(puv_, PoissonUnc),
  VarLabelInfo(biuv_, BinomialUnc),
  VarLabelInfo(nbuv_, NegBinomialUnc),
  VarLabelInfo(geuv_, GeometricUnc),
  VarLabelInfo(hguv_, HyperGeomUnc),
  VarLabelInfo(hpiuv_, HistogramPtIntUnc)
}
```

Var_uinfo DAUSVLbl[DAUSVar_Nkinds] [static]

Initial value:

```
= {
  VarLabelInfo(hpsuv_, HistogramPtStrUnc)
}
```

Var_uinfo DAURVLbl[DAURVar_Nkinds] [static]

Initial value:

```
= {
  VarLabelInfo(hpruv_, HistogramPtRealUnc)
}
```

Var_uinfo CEUVLbl[CEUVar_Nkinds] [static]

Initial value:

```
= {
  VarLabelInfo(ciuv_, ContinuousIntervalUnc)
}
```

Var_uinfo DEUIVLbl[DEUIVar_Nkinds] [static]

Initial value:

```
= {
  VarLabelInfo(diuv_, DiscreteIntervalUnc),
  VarLabelInfo(dusiv_, DiscreteUncSetInt)
}
```

Var_uinfo DEUSVLbl[DEUSVar_Nkinds] [static]

Initial value:

```
= {
  VarLabelInfo(dussv_, DiscreteUncSetStr)
}
```

Var_uinfo DEURVLbl[DEURVar_Nkinds] [static]

Initial value:

```
= {
  VarLabelInfo(dusrv_, DiscreteUncSetReal)
}
```

Var_uinfo DiscSetLbl[DiscSetVar_Nkinds] [static]

Initial value:

```
= {
  VarLabelInfo(ddsiv_, DiscreteDesSetInt),
  VarLabelInfo(ddssv_, DiscreteDesSetStr),
  VarLabelInfo(ddsrv_, DiscreteDesSetReal),
  VarLabelInfo(dssiv_, DiscreteStateSetInt),
  VarLabelInfo(dsssv_, DiscreteStateSetStr),
  VarLabelInfo(dssrv_, DiscreteStateSetReal)
}
```

VarLabelChk DesignAndStateLabelsCheck[ ] [static]

Initial value:

```
= {
  { AVI numContinuousDesVars, AVI continuousDesignLabels, "cdv_", "cdv_descriptors" },
  { AVI numDiscreteDesRangeVars, AVI discreteDesignRangeLabels, "ddriv_", "ddriv_descriptors" },
  { AVI numDiscreteDesSetIntVars, AVI discreteDesignSetIntLabels, "ddsiv_", "ddsiv_descriptors" },
  { AVI numDiscreteDesSetStrVars, AVI discreteDesignSetStrLabels, "ddssv_", "ddssv_descriptors" },
  { AVI numDiscreteDesSetRealVars, AVI discreteDesignSetRealLabels, "ddsrv_", "ddsrv_descriptors" },
  { AVI numContinuousStateVars, AVI continuousStateLabels, "csv_", "csv_descriptors" },
  { AVI numDiscreteStateRangeVars, AVI discreteStateRangeLabels, "dsriv_", "dsriv_descriptors" },
  { AVI numDiscreteStateSetIntVars, AVI discreteStateSetIntLabels, "dssiv_", "dssiv_descriptors" },
  { AVI numDiscreteStateSetStrVars, AVI discreteStateSetStrLabels, "dsssv_", "dsssv_descriptors" },
  { AVI numDiscreteStateSetRealVars, AVI discreteStateSetRealLabels, "dssrv_", "dssrv_descriptors" },
  { AVI numContinuousDesVars, AVI continuousDesignScaleTypes, 0, "cdv_scale_types" }
}
```

Variables label array designations for design and state. All non-uncertain variables need to be in this array. Used in check_variables_node to check lengths and make_variable_defaults to build labels.

Referenced by NIDRProblemDescDB::check_variables_node(), and NIDRProblemDescDB::make_variable_defaults().

VLreal VLUncertainReal[NUM_UNC_REAL_CONT] [static]

Initial value:

```
= {
 {CAUVar_Nkinds,  AVI CAUv,  CAUVLbl,
        DVR continuousAleatoryUncLabels,
        DVR continuousAleatoryUncLowerBnds,
        DVR continuousAleatoryUncUpperBnds,
        DVR continuousAleatoryUncVars},
 {CEUVar_Nkinds,  AVI CEUv,  CEUVLbl,
        DVR continuousEpistemicUncLabels,
        DVR continuousEpistemicUncLowerBnds,
        DVR continuousEpistemicUncUpperBnds,
        DVR continuousEpistemicUncVars},
 {DAURVar_Nkinds, AVI DAURv, DAURVLbl,
        DVR discreteRealAleatoryUncLabels,
        DVR discreteRealAleatoryUncLowerBnds,
        DVR discreteRealAleatoryUncUpperBnds,
        DVR discreteRealAleatoryUncVars},
 {DEURVar_Nkinds, AVI DEURv, DEURVLbl,
        DVR discreteRealEpistemicUncLabels,
        DVR discreteRealEpistemicUncLowerBnds,
        DVR discreteRealEpistemicUncUpperBnds,
        DVR discreteRealEpistemicUncVars}}
```

Variables labels/bounds/values check array for real-valued uncertain variables; one array entry per contiguous container. These associate the individual variables given by, e.g., CAUVLbl, with the contiguous container in which they are stored.

Referenced by NIDRProblemDescDB::check_variables_node(), and NIDRProblemDescDB::make_variable_defaults().

VLint VLUncertainInt[NUM_UNC_INT_CONT] [static]

Initial value:

```
= {
 {DAUIVar_Nkinds, AVI DAUIv, DAUIVLbl,
        DVR discreteIntAleatoryUncLabels,
        DVR discreteIntAleatoryUncLowerBnds,
        DVR discreteIntAleatoryUncUpperBnds,
        DVR discreteIntAleatoryUncVars},
 {DEUIVar_Nkinds, AVI DEUIv, DEUIVLbl,
        DVR discreteIntEpistemicUncLabels,
        DVR discreteIntEpistemicUncLowerBnds,
        DVR discreteIntEpistemicUncUpperBnds,
        DVR discreteIntEpistemicUncVars}}
```

Variables labels/bounds/values check array for integer-valued uncertain variables; one array entry per contiguous container. These associate the individual variables given by, e.g., DAUIVLbl, with the contiguous container in which they are stored.

Referenced by NIDRProblemDescDB::check_variables_node(), and NIDRProblemDescDB::make_variable_defaults().

VLstr VLUncertainStr[NUM_UNC_STR_CONT] [static]

Initial value:

```
= {
 {DAUSVar_Nkinds, AVI DAUSv, DAUSVLbl,
        DVR discreteStrAleatoryUncLabels,
        DVR discreteStrAleatoryUncLowerBnds,
        DVR discreteStrAleatoryUncUpperBnds,
        DVR discreteStrAleatoryUncVars},
 {DEUSVar_Nkinds, AVI DEUSv, DEUSVLbl,
        DVR discreteStrEpistemicUncLabels,
        DVR discreteStrEpistemicUncLowerBnds,
        DVR discreteStrEpistemicUncUpperBnds,
        DVR discreteStrEpistemicUncVars}}
```

Variables labels/bounds/values check array for string-valued uncertain variables; one array entry per contiguous container. These associate the individual variables given by, e.g., DAUSVLbl, with the contiguous container in which they are stored.

Referenced by NIDRProblemDescDB::check_variables_node(), and NIDRProblemDescDB::make_variable_defaults().

Var_check var_mp_check_cv[ ] [static]

Initial value:

```
= {
        Vchk_3(continuous_design,ContinuousDes),
        Vchk_3(continuous_state,ContinuousState) }
```

Var_check var_mp_check_dset[ ] [static]

Initial value:

```
= {
        Vchk_3(discrete_design_set_integer,DiscreteDesSetInt),
        Vchk_3(discrete_design_set_string,DiscreteDesSetStr),
        Vchk_3(discrete_design_set_real,DiscreteDesSetReal),
        Vchk_3(discrete_state_set_integer,DiscreteStateSetInt),
        Vchk_3(discrete_state_set_string,DiscreteStateSetStr),
        Vchk_3(discrete_state_set_real,DiscreteStateSetReal) }
```

Var_check var_mp_check_cau[ ] [static]

Initial value:

```
= {
        Vchk_3(normal_uncertain,NormalUnc),
        Vchk_3(lognormal_uncertain,LognormalUnc),
        Vchk_3(uniform_uncertain,UniformUnc),
        Vchk_3(loguniform_uncertain,LoguniformUnc),
        Vchk_3(triangular_uncertain,TriangularUnc),
        Vchk_3(exponential_uncertain,ExponentialUnc),
        Vchk_3(beta_uncertain,BetaUnc),
        Vchk_3(gamma_uncertain,GammaUnc),
        Vchk_3(gumbel_uncertain,GumbelUnc),
        Vchk_3(frechet_uncertain,FrechetUnc),
        Vchk_3(weibull_uncertain,WeibullUnc),
        Vchk_3(histogram_bin_uncertain,HistogramBinUnc) }
```

Var_check var_mp_check_daui[ ] [static]

Initial value:

```
= {
        Vchk_3(poisson_uncertain,PoissonUnc),
        Vchk_3(binomial_uncertain,BinomialUnc),
        Vchk_3(negative_binomial_uncertain,NegBinomialUnc),
        Vchk_3(geometric_uncertain,GeometricUnc),
        Vchk_3(hypergeometric_uncertain,HyperGeomUnc),
        Vchk_3(histogram_point_int_uncertain,HistogramPtIntUnc) }
```

Var_check var_mp_check_daus[ ] [static]

Initial value:

```
= {
        Vchk_3(histogram_point_str_uncertain,HistogramPtStrUnc) }
```

Var_check var_mp_check_daur[ ] [static]

Initial value:

```
= {
        Vchk_3(histogram_point_real_uncertain,HistogramPtRealUnc) }
```

Var_check var_mp_check_ceu[ ] [static]

Initial value:

```
= {
        Vchk_3(continuous_interval_uncertain,ContinuousIntervalUnc) }
```

Var_check var_mp_check_deui[ ] [static]

Initial value:

```
= {
        Vchk_3(discrete_interval_uncertain,DiscreteIntervalUnc),
        Vchk_3(discrete_uncertain_set_integer,DiscreteUncSetInt) }
```

Var_check var_mp_check_deus[ ] [static]

Initial value:

```
= {
        Vchk_3(discrete_uncertain_set_string,DiscreteUncSetStr) }
```

Var_check var_mp_check_deur[ ] [static]

Initial value:

```
= {
        Vchk_3(discrete_uncertain_set_real,DiscreteUncSetReal) }
```

Var_rcheck var_mp_cbound[ ] [static]

Initial value:

```
= {
        Vchk_7(continuous_design,ContinuousDes,continuousDesign),
        Vchk_7(continuous_state,ContinuousState,continuousState),



        Vchk_5(normal_uncertain,NormalUnc,normalUnc),
        Vchk_5(lognormal_uncertain,LognormalUnc,lognormalUnc),
        Vchk_5(uniform_uncertain,UniformUnc,uniformUnc),
        Vchk_5(loguniform_uncertain,LoguniformUnc,loguniformUnc),
        Vchk_5(triangular_uncertain,TriangularUnc,triangularUnc),
        Vchk_5(beta_uncertain,BetaUnc,betaUnc) }
```

This is used within check_variables_node(): Var_RealBoundIPCheck() is applied to validate bounds and initial points.

Referenced by NIDRProblemDescDB::check_variables_node().

Var_icheck var_mp_drange[ ] [static]

Initial value:

```
= {
        Vchk_7(discrete_design_range,DiscreteDesRange,discreteDesignRange),
        Vchk_7(discrete_state_range,DiscreteStateRange,discreteStateRange) }
```

This is used in check_variables_node(): Var_IntBoundIPCheck() is applied to validate bounds and initial points, and in make_variable_defaults(): Vgen_∗ is called to infer bounds.

Referenced by NIDRProblemDescDB::check_variables_node(), and NIDRProblemDescDB::make_variable_defaults().

const char∗ SCI_FIELD_NAMES[ ]

Initial value:

```
= { "dakota_type", "numFns", "numVars", "numACV", "numADIV",
                                    "numADRV", "numDerivVars", "xC", "xDI",
                                    "xDR", "xCLabels", "xDILabels",
                                    "xDRLabels", "directFnASV", "directFnASM",
                                    "directFnDVV", "directFnDVV_bool",
                                    "fnFlag", "gradFlag", "hessFlag",
                                    "fnVals",  "fnGrads",  "fnHessians",
                                    "fnLabels", "failure", "currEvalId" }
```

fields to pass to Scilab in Dakota structure

Referenced by ScilabInterface::scilab_engine_run().

const int SCI_NUMBER_OF_FIELDS = 26

number of fields in above structure
Referenced by ScilabInterface::scilab_engine_run().

12.2 SIM Namespace Reference

A sample namespace for derived classes that use assign_rep() to plug facilities into DAKOTA.

Classes

- class ParallelDirectApplicInterface

  Sample derived interface class for testing parallel simulator plug-ins using assign_rep().
- class SerialDirectApplicInterface

  Sample derived interface class for testing serial simulator plug-ins using assign_rep().

12.2.1 Detailed Description

A sample namespace for derived classes that use assign_rep() to plug facilities into DAKOTA. A typical use of plug-ins with assign_rep() is to publish a simulation interface for use in library mode See Interfacing with Dakota as a Library for more information.

Chapter 13

Class Documentation

13.1 ActiveSet Class Reference

Container class for active set tracking information. Contains the active set request vector and the derivative variables vector.

Public Member Functions

- ActiveSet ()

  default constructor
- ActiveSet (size_t num_fns, size_t num_deriv_vars)

  standard constructor
- ActiveSet (size_t num_fns)

  partial constructor
- ActiveSet (const ActiveSet &set)

  copy constructor
- ~ActiveSet ()

  destructor
- ActiveSet & operator= (const ActiveSet &set)

  assignment operator
- void reshape (size_t num_fns, size_t num_deriv_vars)

  reshape requestVector and derivVarsVector
- void reshape (size_t num_fns)

  reshape requestVector
- const ShortArray & request_vector () const

  return the request vector
- void request_vector (const ShortArray &rv)

  set the request vector
- void request_values (const short rv_val)

  set all request vector values
- short request_value (const size_t index) const

  get the value of an entry in the request vector

- void request_value (const short rv_val, const size_t index)

  set the value of an entry in the request vector
- const SizetArray & derivative_vector () const

  return the derivative variables vector
- void derivative_vector (const SizetArray &dvv)

  set the derivative variables vector from a SizetArray
- void derivative_vector (SizetMultiArrayConstView dvv)

  set the derivative variables vector from a SizetMultiArrayConstView
- void derivative_start_value (size_t dvv_start_val)

  set the derivative variables vector values
- void read (std::istream &s)

  read an active set object from an std::istream
- void write (std::ostream &s) const

  write an active set object to an std::ostream
- void write_annotated (std::ostream &s) const

  write an active set object to an std::ostream in annotated format
- void read (MPIUnpackBuffer &s)

  read an active set object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write an active set object to a packed MPI buffer

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  implementation of Boost serialize for ActiveSet

Private Attributes

- ShortArray requestVector

  the vector of response requests
- SizetArray derivVarsVector

  the vector of variable ids used for computing derivatives

Friends

- class **boost::serialization::access**
- bool operator== (const ActiveSet &set1, const ActiveSet &set2)

  equality operator
- bool operator!= (const ActiveSet &set1, const ActiveSet &set2)

  inequality operator

13.1.1 Detailed Description

Container class for active set tracking information. Contains the active set request vector and the derivative variables vector.

The ActiveSet class is a small class whose initial design function is to avoid having to pass the ASV and DVV separately. It is not part of a class hierarchy and does not employ reference-counting/ representation-sharing idioms (e.g., handle-body).

13.1.2 Member Data Documentation

ShortArray requestVector `[private]`

the vector of response requests

It uses a 0 value for inactive functions and sums 1 (value), 2 (gradient), and 4 (Hessian) for active functions.

Referenced by ActiveSet::ActiveSet(), ActiveSet::operator=(), ActiveSet::read(), ActiveSet::request_value(), ActiveSet::request_values(), ActiveSet::request_vector(), ActiveSet::reshape(), ActiveSet::write(), and ActiveSet::write_annotated().

SizetArray derivVarsVector `[private]`

the vector of variable ids used for computing derivatives

These ids will generally identify either the active continuous variables or the inactive continuous variables.

Referenced by ActiveSet::ActiveSet(), ActiveSet::derivative_start_value(), ActiveSet::derivative_vector(), ActiveSet::operator=(), ActiveSet::read(), ActiveSet::reshape(), ActiveSet::write(), and ActiveSet::write_annotated().

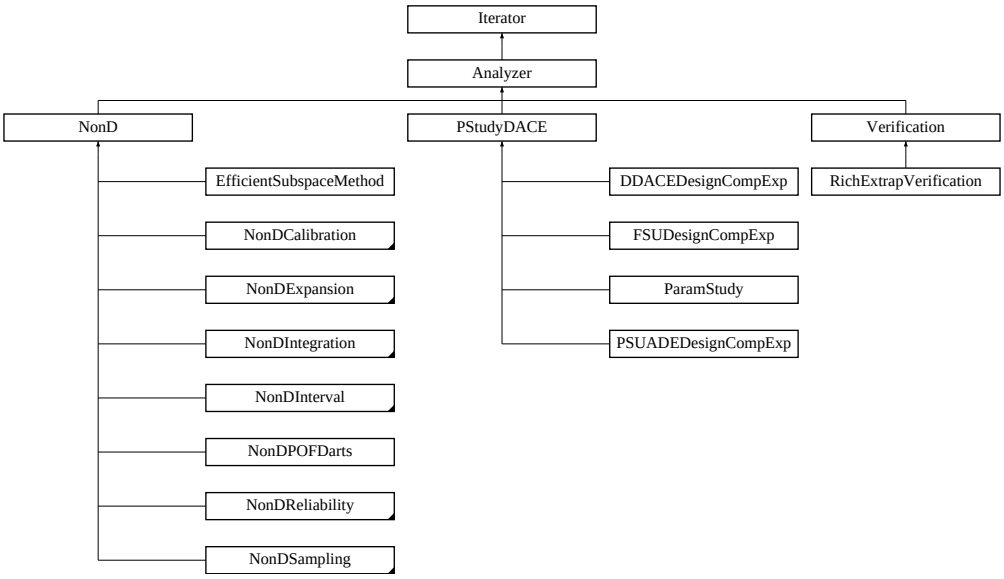
The documentation for this class was generated from the following file:

- DakotaActiveSet.hpp

13.2 Analyzer Class Reference

Base class for NonD, DACE, and ParamStudy branches of the iterator hierarchy.

Inheritance diagram for Analyzer:



Public Member Functions

- const VariablesArray & all_variables ()

  return the complete set of evaluated variables
- const RealMatrix & all_samples ()

  return the complete set of evaluated samples
- const IntResponseMap & all_responses () const

  return the complete set of computed responses
- int num_samples () const
- virtual void vary_pattern (bool pattern_flag)

  sets varyPattern in derived classes that support it

Protected Member Functions

- Analyzer ()

  default constructor
- Analyzer (ProblemDescDB &problem_db, Model &model)

  standard constructor
- Analyzer (unsigned short method_name, Model &model)

  alternate constructor for instantiations "on the fly" with a Model
- Analyzer (unsigned short method_name)

  alternate constructor for instantiations "on the fly" without a Model
- ~Analyzer ()

  destructor
- virtual void get_parameter_sets (Model &model)

  Returns one block of samples (ndim ∗ num_samples)
- virtual void update_model_from_sample (Model &model, const Real ∗sample_vars)

update model's current variables with data from sample

- virtual void update_model_from_variables (Model &model, const Variables &vars)

  update model's current variables with data from vars
- virtual void sample_to_variables (const Real *sample_vars, Variables &vars)

  convert column of samples array to variables; derived classes may reimplement for more than active continuous variables
- void update_from_model (const Model &model)

  set inherited data attributes based on extractions from incoming model
- void initialize_run ()

  utility function to perform common operations prior to pre_run(); typically memory initialization; setting of instance pointers
- void post_run (std::ostream &s)

  post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way
- void pre_output ()
- void print_results (std::ostream &s)

  print the final iterator results
- const Model & algorithm_space_model () const
- const Variables & variables_results () const

  return a single final iterator solution (variables)
- const Response & response_results () const

  return a single final iterator solution (response)
- const VariablesArray & variables_array_results ()

  return multiple final iterator solutions (variables). This should only be used if returns_multiple_points() returns true.
- const ResponseArray & response_array_results ()

  return multiple final iterator solutions (response). This should only be used if returns_multiple_points() returns true.
- void response_results_active_set (const ActiveSet &set)

  set the requested data for the final iterator response results
- bool compact_mode () const

  returns Analyzer::compactMode
- bool returns_multiple_points () const

  indicates if this iterator returns multiple final points. Default return is false. Override to return true if appropriate.
- void evaluate_parameter_sets (Model &model, bool log_resp_flag, bool log_best_flag)

  perform function evaluations to map parameter sets (allVariables) into response sets (allResponses)
- void variance_based_decomp (int ncont, int ndiscint, int ndiscreal, int num_samples)
- void read_variables_responses (int num_evals, size_t num_vars)

  convenience function for reading variables/responses (used in derived classes post_input)
- void print_sobol_indices (std::ostream &s) const

  Printing of VBD results.
- void samples_to_variables_array (const RealMatrix &sample_matrix, VariablesArray &vars_array)

  convert samples array to variables array; e.g., allSamples to allVariables
- virtual void variables_to_sample (const Variables &vars, Real *sample_c_vars)

  convert the active continuous variables into a column of allSamples
- void variables_array_to_samples (const VariablesArray &vars_array, RealMatrix &sample_matrix)

  convert variables array to samples array; e.g., allVariables to allSamples

Protected Attributes

- size_t numFunctions

  number of response functions
- size_t numContinuousVars

  number of active continuous vars
- size_t numDiscreteIntVars

  number of active discrete integer vars
- size_t numDiscreteStringVars

  number of active discrete string vars
- size_t numDiscreteRealVars

  number of active discrete real vars
- bool compactMode

  switch for allSamples (compact mode) instead of allVariables (normal mode)
- VariablesArray allVariables

  array of all variables to be evaluated in evaluate_parameter_sets()
- RealMatrix allSamples

  compact alternative to allVariables
- IntResponseMap allResponses

  array of all responses to be computed in evaluate_parameter_sets()
- StringArray allHeaders

  array of headers to insert into output while evaluating allVariables
- size_t numObjFns

  number of objective functions
- size_t numLSqTerms

  number of least squares terms
- RealPairPRPMultiMap bestVarsRespMap

  map which stores best set of solutions

Private Member Functions

- void compute_best_metrics (const Response &response, std::pair< Real, Real > &metrics)

  compares current evaluation to best evaluation and updates best
- void update_best (const Variables &vars, int eval_id, const Response &response)

  compares current evaluation to best evaluation and updates best
- void update_best (const Real *sample_c_vars, int eval_id, const Response &response)

  compares current evaluation to best evaluation and updates best

Private Attributes

- int writePrecision

  write precision as specified by the user
- Real vbdDropTol

  tolerance for omitting output of small VBD indices
- RealVectorArray S4

VBD main effect indices.

- RealVectorArray T4

  VBD total effect indices.

13.2.1 Detailed Description

Base class for NonD, DACE, and ParamStudy branches of the iterator hierarchy.

The Analyzer class provides common data and functionality for various types of systems analysis, including nondeterministic analysis, design of experiments, and parameter studies.

13.2.2 Member Function Documentation

int num_samples ( ) const `[inline], [virtual]`

Return current number of evaluation points. Since the calculation of samples, collocation points, etc. might be costly, provide a default implementation here that backs out from the maxEvalConcurrency.

Reimplemented from Iterator.

Reimplemented in NonDQuadrature, NonDSparseGrid, NonDSampling, NonDCubature, DDACEDesignCompExp, FSUDesignCompExp, and PSUADEDesignCompExp.

References Model::derivative_concurrency(), Iterator::iteratedModel, and Iterator::maxEvalConcurrency.

Referenced by NonDGlobalReliability::get_best_sample(), Analyzer::samples_to_variables_array(), Analyzer::variables_array_to_samples(), and Analyzer::variance_based_decomp().

void sample_to_variables (const Real * *sample_c_vars*, Variables & *vars*) `[protected], [virtual]`

convert column of samples array to variables; derived classes may reimplement for more than active continuous variables

Default mapping that maps into continuous part of Variables only

Reimplemented in NonDSampling.

References Variables::adiv(), Variables::adrv(), Variables::all_discrete_int_variables(), Variables::all_discrete_real_variables(), Variables::continuous_variable(), Model::current_variables(), Variables::inactive_continuous_variables(), Variables::is_null(), Iterator::iteratedModel, Analyzer::numContinuousVars, and Variables::shared_data().

Referenced by NonDLHSEvidence::post_process_samples(), Analyzer::pre_output(), Analyzer::samples_to_variables_array(), and Analyzer::update_best().

void initialize_run ( ) `[protected], [virtual]`

utility function to perform common operations prior to pre_run(); typically memory initialization; setting of instance pointers

Perform initialization phases of run sequence, like allocating memory and setting instance pointers. Commonly used in sub-iterator executions. This is a virtual function; when re-implementing, a derived class must call its nearest parent's initialize_run(), typically *before* performing its own implementation steps.

Reimplemented from Iterator.

Reimplemented in NonD.

References Model::is_null(), Iterator::iteratedModel, Model::set_evaluation_reference(), and Iterator::summaryOutputFlag.

Referenced by NonD::initialize_run().

void post_run (std::ostream & *s*) [protected], [virtual]

post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way

Post-run phase, which a derived iterator may optionally reimplement; when not present, post-run is likely integrated into run. This is a virtual function; when re-implementing, a derived class must call its nearest parent's post_run(), typically *after* performing its own implementation steps.

Reimplemented from Iterator.

Reimplemented in NonDLHSSampling, DDACEDesignCompExp, ParamStudy, FSUDesignCompExp, and PSUADEDesignCompExp.

References Model::is_null(), Iterator::iteratedModel, Model::print_evaluation_summary(), Analyzer::print_-results(), Iterator::resultsDB, Iterator::summaryOutputFlag, and ResultsManager::write_databases().

Referenced by PSUADEDesignCompExp::post_run(), FSUDesignCompExp::post_run(), ParamStudy::post_-run(), DDACEDesignCompExp::post_run(), and NonDLHSSampling::post_run().

void pre_output ( ) [protected], [virtual]

Generate tabular output with active variables (compactMode) or all variables with their labels and response labels, with no data. Variables are sequenced {cv, div, drv}

Reimplemented from Iterator.

References Analyzer::allSamples, Analyzer::allVariables, ParallelLibrary::command_line_pre_run_output(), Parallel-Library::command_line_user_modes(), Analyzer::compactMode, Variables::copy(), Model::current_response(), Model-::current_variables(), Model::interface_id(), Iterator::iteratedModel, Iterator::outputLevel, Iterator::parallelLib, Analyzer-::sample_to_variables(), Dakota::write_precision, Variables::write_tabular(), and Analyzer::writePrecision.

void print_results (std::ostream & *s*) [protected], [virtual]

print the final iterator results

This virtual function provides additional iterator-specific final results outputs beyond the function evaluation summary printed in finalize_run().

Reimplemented from Iterator.

Reimplemented in NonDLHSSampling, NonDLocalReliability, NonDPOFDarts, NonDAdaptiveSampling, NonDGPImpSampling, NonDAdaptImpSampling, NonDGlobalReliability, NonDIncremLHSSampling, NonD-Interval, NonDExpansion, PStudyDACE, RichExtrapVerification, and Verification.

References Analyzer::bestVarsRespMap, ParamResponsePair::eval_id(), Response::function_values(), Analyzer-::numLSqTerms, Analyzer::numObjFns, ParamResponsePair::prp_parameters(), and ParamResponsePair::prp_-response().

Referenced by Analyzer::post_run(), Verification::print_results(), PStudyDACE::print_results(), and NonDL-HSSampling::print_results().

const Model & algorithm_space_model ( ) const [inline], [protected], [virtual]

default definition that gets redefined in selected derived Minimizers

Reimplemented from Iterator.

Reimplemented in NonDExpansion, NonDGlobalInterval, NonDBayesCalibration, and NonDReliability.

References Iterator::iteratedModel.

void evaluate_parameter_sets (Model & *model*, bool *log_resp_flag*, bool *log_best_flag*) [protected]

perform function evaluations to map parameter sets (allVariables) into response sets (allResponses)

Convenience function for derived classes with sets of function evaluations to perform (e.g., NonDSampling, DDACEDesignCompExp, FSUDesignCompExp, ParamStudy).

References Iterator::activeSet, Analyzer::allHeaders, Analyzer::allResponses, Analyzer::allSamples, Analyzer::allVariables, Model::asynch_compute_response(), Model::asynch_flag(), Analyzer::compactMode, Model::compute_response(), Response::copy(), Model::current_response(), Model::current_variables(), Model::evaluation_id(), Model::synchronize(), Analyzer::update_best(), Analyzer::update_model_from_sample(), and Analyzer::update_model_from_variables().

Referenced by NonDSparseGrid::evaluate_grid_increment(), NonDSparseGrid::evaluate_set(), PSUADEDesignCompExp::extract_trends(), FSUDesignCompExp::extract_trends(), ParamStudy::extract_trends(), DDACEDesignCompExp::extract_trends(), NonDIncremLHSSampling::quantify_uncertainty(), NonDAdaptImpSampling::quantify_uncertainty(), NonDLHSSampling::quantify_uncertainty(), NonDIntegration::quantify_uncertainty(), and Analyzer::variance_based_decomp().

void variance_based_decomp (int *ncont*, int *ndiscint*, int *ndiscreal*, int *num_samples*) [protected]

Calculation of sensitivity indices obtained by variance based decomposition. These indices are obtained by the Saltelli version of the Sobol VBD which uses (K+2)∗N function evaluations, where K is the number of dimensions (uncertain vars) and N is the number of samples.

References Dakota::abort_handler(), Analyzer::allResponses, Analyzer::allSamples, Analyzer::allVariables, Analyzer::compactMode, Variables::continuous_variables(), Dakota::copy_data(), Variables::discrete_int_variables(), Variables::discrete_real_variables(), Analyzer::evaluate_parameter_sets(), Analyzer::get_parameter_sets(), Iterator::iteratedModel, Analyzer::num_samples(), Analyzer::numFunctions, Analyzer::S4, Analyzer::T4, and Analyzer::vary_pattern().

Referenced by FSUDesignCompExp::extract_trends(), DDACEDesignCompExp::extract_trends(), and NonDLHSSampling::quantify_uncertainty().

void read_variables_responses (int *num_evals*, size_t *num_vars*) [protected]

convenience function for reading variables/responses (used in derived classes post_input)

read num_evals variables/responses from file

References Dakota::abort_handler(), Analyzer::allResponses, Analyzer::allSamples, Analyzer::allVariables, ParallelLibrary::command_line_post_run_input(), ParallelLibrary::command_line_user_modes(), Analyzer::compactMode, Response::copy(), Variables::copy(), Model::current_response(), Model::current_variables(), Iterator::iteratedModel, Iterator::outputLevel, Iterator::parallelLib, Variables::read_tabular(), Analyzer::update_best(), and Analyzer::variables_to_sample().

Referenced by PSUADEDesignCompExp::post_input(), FSUDesignCompExp::post_input(), ParamStudy::post_input(), DDACEDesignCompExp::post_input(), and NonDLHSSampling::post_input().

void print_sobol_indices (std::ostream & *s*) const [protected]

Printing of VBD results.

printing of variance based decomposition indices.

References Model::continuous_variable_labels(), Model::discrete_int_variable_labels(), Model::discrete_real_variable_labels(), Iterator::iteratedModel, Analyzer::numContinuousVars, Analyzer::numDiscreteIntVars, Analyzer::numDiscreteRealVars, Analyzer::numFunctions, Model::response_labels(), Analyzer::S4, Analyzer::T4, Analyzer::vbdDropTol, and Dakota::write_precision.

Referenced by PStudyDACE::print_results(), and NonDLHSSampling::print_results().

void variables_to_sample (const Variables & *vars*, Real * *sample_c_vars*) [protected], [virtual]

convert the active continuous variables into a column of allSamples

Default implementation maps active continuous variables only

Reimplemented in NonDSampling.

References Variables::continuous_variables(), and Analyzer::numContinuousVars.

Referenced by Analyzer::read_variables_responses(), and Analyzer::variables_array_to_samples().

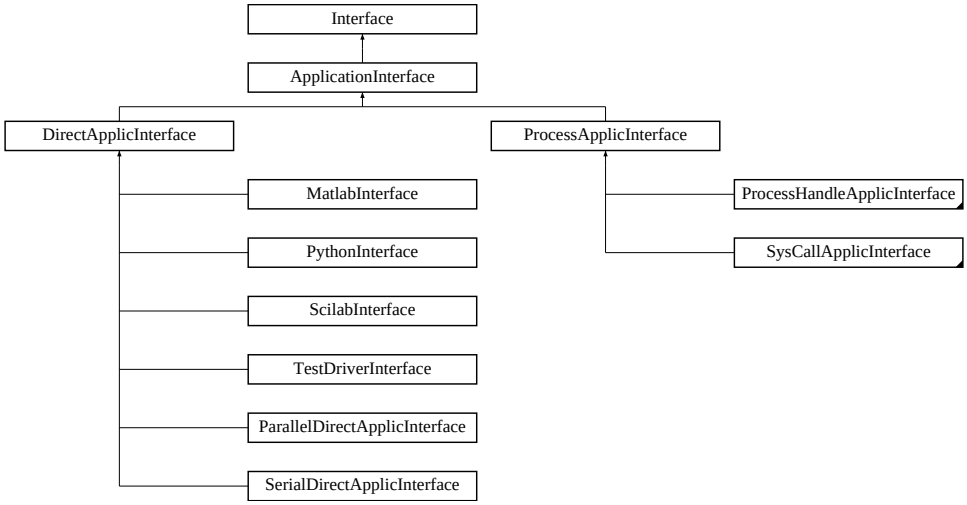
The documentation for this class was generated from the following files:

- DakotaAnalyzer.hpp
- DakotaAnalyzer.cpp

13.3 ApplicationInterface Class Reference

Derived class within the interface class hierarchy for supporting interfaces to simulation codes.

Inheritance diagram for ApplicationInterface:



Public Member Functions

- ApplicationInterface (const ProblemDescDB &problem_db)

  constructor
- ~ApplicationInterface ()

  destructor

Protected Member Functions

- void init_communicators (const IntArray &message_lengths, int max_eval_concurrency)

  allocate communicator partitions for concurrent evaluations within an iterator and concurrent multiprocessor analyses within an evaluation.
- void set_communicators (const IntArray &message_lengths, int max_eval_concurrency)

  set the local parallel partition data for an interface (the partitions are already allocated in ParallelLibrary).

- void free_communicators ()

  deallocate communicator partitions for concurrent evaluations within an iterator and concurrent multiprocessor analyses within an evaluation.
- void init_serial ()
- int asynch_local_evaluation_concurrency () const

  return asynchLocalEvalConcurrency
- short interface_synchronization () const

  return interfaceSynchronization
- bool evaluation_cache () const

  return evalCacheFlag
- void map (const Variables &vars, const ActiveSet &set, Response &response, bool asynch_flag=false)

  Provides a "mapping" of variables to responses using a simulation. Protected due to Interface letter-envelope idiom.
- void manage_failure (const Variables &vars, const ActiveSet &set, Response &response, int failed_eval_id)

  manages a simulation failure using abort/retry/recover/continuation
- const IntResponseMap & synch ()

  executes a blocking schedule for asynchronous evaluations in the beforeSynchCorePRPQueue and returns all jobs
- const IntResponseMap & synch_nowait ()

  executes a nonblocking schedule for asynchronous evaluations in the beforeSynchCorePRPQueue and returns a partial set of completed jobs
- void serve_evaluations ()

  run on evaluation servers to serve the iterator master
- void stop_evaluation_servers ()

  used by the iterator master to terminate evaluation servers
- bool check_multiprocessor_analysis (bool warn)

  checks on multiprocessor analysis configuration
- bool check_asynchronous (bool warn, int max_eval_concurrency)

  checks on asynchronous configuration (for direct interfaces)
- bool check_multiprocessor_asynchronous (bool warn, int max_eval_concurrency)

  checks on asynchronous settings for multiprocessor partitions
- virtual void derived_map (const Variables &vars, const ActiveSet &set, Response &response, int fn_eval_id)

  Called by map() and other functions to execute the simulation in synchronous mode. The portion of performing an evaluation that is specific to a derived class.
- virtual void derived_map_asynch (const ParamResponsePair &pair)

  Called by map() and other functions to execute the simulation in asynchronous mode. The portion of performing an asynchronous evaluation that is specific to a derived class.
- virtual void wait_local_evaluations (PRPQueue &prp_queue)

  For asynchronous function evaluations, this method is used to detect completion of jobs and process their results. It provides the processing code that is specific to derived classes. This version waits for at least one completion.
- virtual void test_local_evaluations (PRPQueue &prp_queue)

  For asynchronous function evaluations, this method is used to detect completion of jobs and process their results. It provides the processing code that is specific to derived classes. This version is nonblocking and will return without any completions if none are immediately available.
- virtual void init_communicators_checks (int max_eval_concurrency)

  perform construct-time error checks on the parallel configuration

- virtual void set_communicators_checks (int max_eval_concurrency)

  perform run-time error checks on the parallel configuration
- void master_dynamic_schedule_analyses ()

  blocking dynamic schedule of all analyses within a function evaluation using message passing
- void serve_analyses_synch ()

  serve the master analysis scheduler and manage one synchronous analysis job at a time
- virtual int synchronous_local_analysis (int analysis_id)

  Execute a particular analysis (identified by analysis_id) synchronously on the local processor. Used for the derived class specifics within ApplicationInterface::serve_analyses_synch().

Protected Attributes

- ParallelLibrary & parallelLib

  reference to the ParallelLibrary object used to manage MPI partitions for the concurrent evaluations and concurrent analyses parallelism levels
- bool suppressOutput

  flag for suppressing output on slave processors
- int evalCommSize

  size of evalComm
- int evalCommRank

  processor rank within evalComm
- int evalServerId

  evaluation server identifier
- bool eaDedMasterFlag

  flag for dedicated master partitioning at ea level
- int analysisCommSize

  size of analysisComm
- int analysisCommRank

  processor rank within analysisComm
- int analysisServerId

  analysis server identifier
- int numAnalysisServers

  current number of analysis servers
- bool multiProcAnalysisFlag

  flag for multiprocessor analysis partitions
- bool asynchLocalAnalysisFlag

  flag for asynchronous local parallelism of analyses
- int asynchLocalAnalysisConcurrency

  limits the number of concurrent analyses in asynchronous local scheduling and specifies hybrid concurrency when message passing
- int asynchLocalEvalConcSpec

  user specification for asynchronous local evaluation concurrency
- int asynchLocalAnalysisConcSpec

  user specification for asynchronous local analysis concurrency

- int numAnalysisDrivers

  the number of analysis drivers used for each function evaluation (from the analysis_drivers interface specification)
- IntSet completionSet

  the set of completed fn_eval_id's populated by wait_local_evaluations() and test_local_evaluations()
- String failureMessage

  base message for managing failed evals; will be followed with more details in screen output

Private Member Functions

- bool duplication_detect (const Variables &vars, Response &response, bool asynch_flag)

  checks data_pairs and beforeSynchCorePRPQueue to see if the current evaluation request has already been performed or queued
- void master_dynamic_schedule_evaluations ()

  blocking dynamic schedule of all evaluations in beforeSynchCorePRPQueue using message passing on a dedicated master partition; executes on iteratorComm master
- void peer_static_schedule_evaluations ()

  blocking static schedule of all evaluations in beforeSynchCorePRPQueue using message passing on a peer partition; executes on iteratorComm master
- void peer_dynamic_schedule_evaluations ()

  blocking dynamic schedule of all evaluations in beforeSynchCorePRPQueue using message passing on a peer partition; executes on iteratorComm master
- void asynchronous_local_evaluations (PRPQueue &prp_queue)

  perform all jobs in prp_queue using asynchronous approaches on the local processor
- void synchronous_local_evaluations (PRPQueue &prp_queue)

  perform all jobs in prp_queue using synchronous approaches on the local processor
- void master_dynamic_schedule_evaluations_nowait ()

  execute a nonblocking dynamic schedule in a master-slave partition
- void peer_dynamic_schedule_evaluations_nowait ()

  execute a nonblocking static/dynamic schedule in a peer partition
- void asynchronous_local_evaluations_nowait (PRPQueue &prp_queue)

  launch new jobs in prp_queue asynchronously (if capacity is available), perform nonblocking query of all running jobs, and process any completed jobs (handles both local master- and local peer-scheduling cases)
- void broadcast_evaluation (const ParamResponsePair &pair)

  convenience function for broadcasting an evaluation over an evalComm
- void broadcast_evaluation (int fn_eval_id, const Variables &vars, const ActiveSet &set)

  convenience function for broadcasting an evaluation over an evalComm
- void send_evaluation (PRPQueueIter &prp_it, size_t buff_index, int server_id, bool peer_flag)

  helper function for sending sendBuffers[buff_index] to server
- void receive_evaluation (PRPQueueIter &prp_it, size_t buff_index, int server_id, bool peer_flag)

  helper function for processing recvBuffers[buff_index] within scheduler
- void launch_asynch_local (PRPQueueIter &prp_it)

  launch an asynchronous local evaluation
- void process_asynch_local (int fn_eval_id)

  process a completed asynchronous local evaluation

- void process_synch_local (PRPQueueIter &prp_it)

  process a completed synchronous local evaluation
- void assign_asynch_local_queue (PRPQueue &local_prp_queue, PRPQueueIter &local_prp_iter)

  helper function for creating an initial active local queue by launching asynch local jobs from local_prp_queue, as limited by server capacity
- void assign_asynch_local_queue_nowait (PRPQueue &local_prp_queue, PRPQueueIter &local_prp_iter)

  helper function for updating an active local queue by backfilling asynch local jobs from local_prp_queue, as limited by server capacity
- size_t test_local_backfill (PRPQueue &assign_queue, PRPQueueIter &assign_iter)

  helper function for testing active asynch local jobs and then backfilling
- size_t test_receives_backfill (PRPQueueIter &assign_iter, bool peer_flag)

  helper function for testing receive requests and then backfilling jobs
- void serve_evaluations_synch ()

  serve the evaluation message passing schedulers and perform one synchronous evaluation at a time
- void serve_evaluations_synch_peer ()

  serve the evaluation message passing schedulers and perform one synchronous evaluation at a time as part of the 1st peer
- void serve_evaluations_asynch ()

  serve the evaluation message passing schedulers and manage multiple asynchronous evaluations
- void serve_evaluations_asynch_peer ()

  serve the evaluation message passing schedulers and perform multiple asynchronous evaluations as part of the 1st peer
- void set_evaluation_communicators (const IntArray &message_lengths)

  convenience function for updating the local evaluation partition data following ParallelLibrary::init_evaluation_communicators().
- void set_analysis_communicators ()

  convenience function for updating the local analysis partition data following ParallelLibrary::init_analysis_communicators().
- void init_serial_evaluations ()

  set concurrent evaluation configuration for serial operations
- void init_serial_analyses ()

  set concurrent analysis configuration for serial operations (e.g., for local executions on a dedicated master)
- const ParamResponsePair & get_source_pair (const Variables &target_vars)

  convenience function for the continuation approach in manage_failure() for finding the nearest successful "source" evaluation to the failed "target"
- void continuation (const Variables &target_vars, const ActiveSet &set, Response &response, const ParamResponsePair &source_pair, int failed_eval_id)

  performs a 0th order continuation method to step from a successful "source" evaluation to the failed "target". Invoked by manage_failure() for failAction == "continuation".
- void common_input_filtering (const Variables &vars)

  common input filtering operations, e.g. mesh movement
- void common_output_filtering (Response &response)

  common output filtering operations, e.g. data filtering

Private Attributes

- int worldSize

  size of MPI_COMM_WORLD
- int worldRank

  processor rank within MPI_COMM_WORLD
- int iteratorCommSize

  size of iteratorComm
- int iteratorCommRank

  processor rank within iteratorComm
- bool ieMessagePass

  flag for message passing at ie scheduling level
- int numEvalServers

  current number of evaluation servers
- int numEvalServersSpec

  user specification for number of evaluation servers
- int procsPerEvalSpec

  user specification for processors per analysis servers
- bool eaMessagePass

  flag for message passing at ea scheduling level
- int numAnalysisServersSpec

  user spec for number of analysis servers
- int procsPerAnalysisSpec

  user specification for processors per analysis servers
- int lenVarsMessage

  length of a MPIPackBuffer containing a Variables object; computed in Model::init_communicators()
- int lenVarsActSetMessage

  length of a MPIPackBuffer containing a Variables object and an ActiveSet object; computed in Model::init_communicators()
- int lenResponseMessage

  length of a MPIPackBuffer containing a Response object; computed in Model::init_communicators()
- int lenPRPairMessage

  length of a MPIPackBuffer containing a ParamResponsePair object; computed in Model::init_communicators()
- short evalScheduling

  user specification of evaluation scheduling algorithm: {DEFAULT,MASTER,PEER_DYNAMIC,PEER_STATIC}_SCHEDULING. Used for manual overrides of auto-configure logic in ParallelLibrary::resolve_inputs().
- short analysisScheduling

  user specification of analysis scheduling algorithm: {DEFAULT,MASTER,PEER}_SCHEDULING. Used for manual overrides of the auto-configure logic in ParallelLibrary::resolve_inputs().
- int asynchLocalEvalConcurrency

  limits the number of concurrent evaluations in asynchronous local scheduling and specifies hybrid concurrency when message passing
- bool asynchLocalEvalStatic

  whether the asynchronous local evaluations are to be performed with a static schedule (default false)

- BitArray localServerAssigned

  array with one bit per logical "server" indicating whether a job is currently running on the server (used for asynch local static schedules)
- short interfaceSynchronization

  interface synchronization specification: synchronous (default) or asynchronous
- bool asvControlFlag

  used to manage a user request to deactivate the active set vector control. true = modify the ASV each evaluation as appropriate (default); false = ASV values are static so that the user need not check them on each evaluation.
- bool evalCacheFlag

  used to manage a user request to deactivate the function evaluation cache (i.e., queries and insertions using the data_pairs cache).
- bool nearbyDuplicateDetect

  flag indicating optional usage of tolerance-based duplication detection (less efficient, but helpful when experiencing restart cache misses)
- Real nearbyTolerance

  tolerance value for tolerance-based duplication detection
- bool restartFileFlag

  used to manage a user request to deactivate the restart file (i.e., insertions into write_restart).
- ShortArray defaultASV

  the static ASV values used when the user has selected asvControl = off
- String failAction

  mitigation action for captured simulation failures: abort, retry, recover, or continuation
- int failRetryLimit

  limit on the number of retries for the retry failAction
- RealVector failRecoveryFnVals

  the dummy function values used for the recover failAction
- IntResponseMap historyDuplicateMap

  used to bookkeep asynchronous evaluations which duplicate data_pairs evaluations. Map key is evalIdCntr, map value is corresponding response.
- std::map< int, std::pair < PRPQueueHIter, Response > > beforeSynchDuplicateMap

  used to bookkeep evalIdCntr, beforeSynchCorePRPQueue iterator, and response of asynchronous evaluations which duplicate queued beforeSynchCorePRPQueue evaluations
- PRPQueue beforeSynchCorePRPQueue

  used to bookkeep vars/set/response of nonduplicate asynchronous core evaluations. This is the queue of jobs populated by asynchronous map() that is later scheduled in synch() or synch_nowait().
- PRPQueue beforeSynchAlgPRPQueue

  used to bookkeep vars/set/response of asynchronous algebraic evaluations. This is the queue of algebraic jobs populated by asynchronous map() that is later evaluated in synch() or synch_nowait().
- PRPQueue asynchLocalActivePRPQueue

  used by nonblocking asynchronous local schedulers to bookkeep active local jobs
- std::map< int, IntIntPair > msgPassRunningMap

  used by nonblocking message passing schedulers to bookkeep which jobs are running remotely
- MPIPackBuffer * sendBuffers

  array of pack buffers for evaluation jobs queued to a server

- MPIUnpackBuffer * recvBuffers

  array of unpack buffers for evaluation jobs returned by a server

- MPI_Request * recvRequests

  array of requests for nonblocking evaluation receives

13.3.1 Detailed Description

Derived class within the interface class hierarchy for supporting interfaces to simulation codes.

ApplicationInterface provides an interface class for performing parameter to response mappings using simulation code(s). It provides common functionality for a number of derived classes and contains the majority of all of the scheduling algorithms in DAKOTA. The derived classes provide the specifics for managing code invocations using system calls, forks, direct procedure calls, or distributed resource facilities.

13.3.2 Member Function Documentation

void init_serial ( ) `[inline], [protected], [virtual]`

DataInterface.cpp defaults of 0 servers are needed to distinguish an explicit user request for 1 server (serialization of a parallelism level) from no user request (use parallel auto-config). This default causes problems when init_communicators() is not called for an interface object (e.g., static scheduling fails in DirectApplicInterface::derived_map() for NestedModel::optionalInterface). This is the reason for this function: to reset certain defaults for interface objects that are used serially.

Reimplemented from Interface.

References ApplicationInterface::init_serial_analyses(), and ApplicationInterface::init_serial_evaluations().

void map (const Variables & *vars*, const ActiveSet & *set*, Response & *response*, bool *asynch_flag* = *`false`*) `[protected], [virtual]`

Provides a "mapping" of variables to responses using a simulation. Protected due to Interface letter-envelope idiom.

The function evaluator for application interfaces. Called from derived_compute_response() and derived_asynch_compute_response() in derived Model classes. If asynch_flag is not set, perform a blocking evaluation (using derived_map()). If asynch_flag is set, add the job to the beforeSynchCorePRPQueue queue for execution by one of the scheduler routines in synch() or synch_nowait(). Duplicate function evaluations are detected with duplication_detect().

Reimplemented from Interface.

References Response::active_set(), Interface::algebraic_mappings(), Interface::algebraicMappings, Interface::asv_mapping(), ApplicationInterface::asvControlFlag, ApplicationInterface::beforeSynchAlgPRPQueue, ApplicationInterface::beforeSynchCorePRPQueue, ApplicationInterface::broadcast_evaluation(), Response::copy(), Interface::coreMappings, Interface::currEvalId, Dakota::data_pairs, ApplicationInterface::defaultASV, ApplicationInterface::derived_map(), ApplicationInterface::duplication_detect(), ApplicationInterface::evalCacheFlag, Interface::evalIdCntr, Interface::fineGrainEvalCounters, Interface::fnGradCounter, Interface::fnHessCounter, Interface::fnLabels, Interface::fnValCounter, Response::function_labels(), Interface::init_algebraic_mappings(), Interface::interfaceId, ApplicationInterface::manage_failure(), Interface::multiProcEvalFlag, Interface::newEvalIdCntr, Interface::newFnGradCounter, Interface::newFnHessCounter, Interface::newFnValCounter, Interface::outputLevel, ApplicationInterface::parallelLib, ActiveSet::request_vector(), Interface::response_mapping(), ApplicationInterface::restartFileFlag, and ParallelLibrary::write_restart().

const IntResponseMap & synch ( ) [protected], [virtual]

executes a blocking schedule for asynchronous evaluations in the beforeSynchCorePRPQueue and returns all jobs

This function provides blocking synchronization for all cases of asynchronous evaluations, including the local asynchronous case (background system call, nonblocking fork, & multithreads), the message passing case, and the hybrid case. Called from derived_synchronize() in derived Model classes.

Reimplemented from Interface.

References Interface::algebraic_mappings(), Interface::algebraicMappings, Interface::asv_mapping(), ApplicationInterface::asynchLocalEvalStatic, ApplicationInterface::asynchronous_local_evaluations(), ApplicationInterface::beforeSynchAlgPRPQueue, ApplicationInterface::beforeSynchCorePRPQueue, ApplicationInterface::beforeSynchDuplicateMap, Interface::coreMappings, ApplicationInterface::evalScheduling, ApplicationInterface::historyDuplicateMap, Interface::ieDedMasterFlag, ApplicationInterface::ieMessagePass, Interface::interfaceId, Interface::interfaceType, ApplicationInterface::master_dynamic_schedule_evaluations(), Interface::multiProcEvalFlag, Interface::outputLevel, ApplicationInterface::peer_dynamic_schedule_evaluations(), ApplicationInterface::peer_static_schedule_evaluations(), Interface::rawResponseMap, and Interface::response_mapping().

const IntResponseMap & synch_nowait ( ) [protected], [virtual]

executes a nonblocking schedule for asynchronous evaluations in the beforeSynchCorePRPQueue and returns a partial set of completed jobs

This function provides nonblocking synchronization for the local asynchronous case and selected nonblocking message passing schedulers. Called from derived_synchronize_nowait() in derived Model classes.

Reimplemented from Interface.

References Dakota::abort_handler(), Interface::algebraic_mappings(), Interface::algebraicMappings, Interface::asv_mapping(), ApplicationInterface::asynchLocalEvalStatic, ApplicationInterface::asynchronous_local_evaluations_nowait(), ApplicationInterface::beforeSynchAlgPRPQueue, ApplicationInterface::beforeSynchCorePRPQueue, ApplicationInterface::beforeSynchDuplicateMap, Interface::coreMappings, ParamResponsePair::eval_id(), ApplicationInterface::evalScheduling, ApplicationInterface::historyDuplicateMap, Interface::ieDedMasterFlag, ApplicationInterface::ieMessagePass, Interface::interfaceId, Interface::interfaceType, Dakota::lookup_by_eval_id(), ApplicationInterface::master_dynamic_schedule_evaluations_nowait(), Interface::multiProcEvalFlag, Interface::outputLevel, ApplicationInterface::peer_dynamic_schedule_evaluations_nowait(), ParamResponsePair::prp_response(), Interface::rawResponseMap, Interface::response_mapping(), and Response::update().

void serve_evaluations ( ) [protected], [virtual]

run on evaluation servers to serve the iterator master

Invoked by the serve() function in derived Model classes. Passes control to serve_evaluations_synch(), serve_evaluations_asynch(), serve_evaluations_synch_peer(), or serve_evaluations_asynch_peer() according to specified concurrency, partition, and scheduler configuration.

Reimplemented from Interface.

References ApplicationInterface::asynchLocalEvalConcurrency, ApplicationInterface::evalServerId, Interface::ieDedMasterFlag, ApplicationInterface::serve_evaluations_asynch(), ApplicationInterface::serve_evaluations_asynch_peer(), ApplicationInterface::serve_evaluations_synch(), and ApplicationInterface::serve_evaluations_synch_peer().

void stop_evaluation_servers ( ) [protected], [virtual]

used by the iterator master to terminate evaluation servers

This code is executed on the iteratorComm rank 0 processor when iteration on a particular model is complete. It sends a termination signal (tag = 0 instead of a valid fn_eval_id) to each of the slave analysis servers. NOTE: This function is called from the Strategy layer even when in serial mode. Therefore, use iteratorCommSize to provide appropriate fall through behavior.

Reimplemented from Interface.

References ParallelLibrary::bcast_e(), ParallelLibrary::free(), ParallelConfiguration::ie_parallel_level(), Interface::ieDedMasterFlag, ParallelLibrary::isend_ie(), ApplicationInterface::iteratorCommSize, Interface::multiProcEvalFlag, ApplicationInterface::numEvalServers, Interface::outputLevel, ParallelLibrary::parallel_configuration(), and ApplicationInterface::parallelLib.

void init_communicators_checks (int *max_eval_concurrency*) [protected], [virtual]

perform construct-time error checks on the parallel configuration

Override DirectApplicInterface definition if plug-in to allow batch processing in Plugin{Serial,Parallel}DirectApplicInterface.cpp

Reimplemented in DirectApplicInterface, SysCallApplicInterface, and ProcessHandleApplicInterface.

Referenced by ApplicationInterface::init_communicators().

void set_communicators_checks (int *max_eval_concurrency*) [protected], [virtual]

perform run-time error checks on the parallel configuration

Override DirectApplicInterface definition if plug-in to allow batch processing in Plugin{Serial,Parallel}DirectApplicInterface.cpp

Reimplemented in DirectApplicInterface, SerialDirectApplicInterface, ParallelDirectApplicInterface, SysCallApplicInterface, and ProcessHandleApplicInterface.

Referenced by ApplicationInterface::set_communicators().

void master_dynamic_schedule_analyses ( ) [protected]

blocking dynamic schedule of all analyses within a function evaluation using message passing

This code is called from derived classes to provide the master portion of a master-slave algorithm for the dynamic scheduling of analyses among slave servers. It is patterned after master_dynamic_schedule_evaluations(). It performs no analyses locally and matches either serve_analyses_synch() or serve_analyses_asynch() on the slave servers, depending on the value of asynchLocalAnalysisConcurrency. Dynamic scheduling assigns jobs in 2 passes. The 1st pass gives each server the same number of jobs (equal to asynchLocalAnalysisConcurrency). The 2nd pass assigns the remaining jobs to slave servers as previous jobs are completed. Single- and multilevel parallel use intra- and inter-communicators, respectively, for send/receive. Specific syntax is encapsulated within ParallelLibrary.

References ApplicationInterface::asynchLocalAnalysisConcurrency, ParallelLibrary::free(), ParallelLibrary::irecv_ea(), ParallelLibrary::isend_ea(), ApplicationInterface::numAnalysisDrivers, ApplicationInterface::numAnalysisServers, ApplicationInterface::parallelLib, ParallelLibrary::waitall(), and ParallelLibrary::waitsome().

Referenced by ProcessHandleApplicInterface::create_evaluation_process(), SysCallApplicInterface::create_evaluation_process(), and DirectApplicInterface::derived_map().

void serve_analyses_synch ( ) [protected]

serve the master analysis scheduler and manage one synchronous analysis job at a time

This code is called from derived classes to run synchronous analyses on slave processors. The slaves receive requests (blocking receive), do local derived_map_ac's, and return codes. This is done continuously until a termination signal is received from the master. It is patterned after serve_evaluations_synch().

References ApplicationInterface::analysisCommRank, ParallelLibrary::bcast_a(), ParallelLibrary::isend_ea(), ApplicationInterface::multiProcAnalysisFlag, ApplicationInterface::parallelLib, ParallelLibrary::recv_ea(), ApplicationInterface::synchronous_local_analysis(), and ParallelLibrary::wait().

Referenced by ProcessHandleApplicInterface::create_evaluation_process(), SysCallApplicInterface::create_evaluation_process(), and DirectApplicInterface::derived_map().

bool duplication_detect (const Variables & *vars,* Response & *response,* bool *asynch_flag*) [private]

checks data_pairs and beforeSynchCorePRPQueue to see if the current evaluation request has already been performed or queued

Called from map() to check incoming evaluation request for duplication with content of data_pairs and beforeSynchCorePRPQueue. If duplication is detected, return true, else return false. Manage bookkeeping with historyDuplicateMap and beforeSynchDuplicateMap. Note that the list searches can get very expensive if a long list is searched on every new function evaluation (either from a large number of previous jobs, a large number of pending jobs, or both). For this reason, a user request for deactivation of the evaluation cache results in a complete bypass of duplication_detect(), even though a beforeSynchCorePRPQueue search would still be meaningful. Since the intent of this request is to streamline operations, both list searches are bypassed.

References Response::active_set(), ApplicationInterface::beforeSynchCorePRPQueue, ApplicationInterface::beforeSynchDuplicateMap, Response::copy(), Dakota::data_pairs, ParamResponsePair::eval_id(), Interface::evalIdCntr, ApplicationInterface::historyDuplicateMap, Interface::interfaceId, Dakota::lookup_by_val(), ApplicationInterface::nearbyDuplicateDetect, ApplicationInterface::nearbyTolerance, and Response::update().

Referenced by ApplicationInterface::map().

void master_dynamic_schedule_evaluations ( ) [private]

blocking dynamic schedule of all evaluations in beforeSynchCorePRPQueue using message passing on a dedicated master partition; executes on iteratorComm master

This code is called from synch() to provide the master portion of a master-slave algorithm for the dynamic scheduling of evaluations among slave servers. It performs no evaluations locally and matches either serve_evaluations_synch() or serve_evaluations_asynch() on the slave servers, depending on the value of asynchLocalEvalConcurrency. Dynamic scheduling assigns jobs in 2 passes. The 1st pass gives each server the same number of jobs (equal to asynchLocalEvalConcurrency). The 2nd pass assigns the remaining jobs to slave servers as previous jobs are completed and returned. Single- and multilevel parallel use intra- and inter-communicators, respectively, for send/receive. Specific syntax is encapsulated within ParallelLibrary. peer

References ApplicationInterface::asynchLocalEvalConcurrency, ApplicationInterface::beforeSynchCorePRPQueue, Dakota::lookup_by_eval_id(), ApplicationInterface::numEvalServers, Interface::outputLevel, ApplicationInterface::parallelLib, ApplicationInterface::receive_evaluation(), ApplicationInterface::recvBuffers, ApplicationInterface::recvRequests, ApplicationInterface::send_evaluation(), ApplicationInterface::sendBuffers, ParallelLibrary::waitall(), and ParallelLibrary::waitsome().

Referenced by ApplicationInterface::synch().

void peer_static_schedule_evaluations ( ) [private]

blocking static schedule of all evaluations in beforeSynchCorePRPQueue using message passing on a peer partition; executes on iteratorComm master

This code runs on the iteratorCommRank 0 processor (the iterator) and is called from synch() in order to manage a static schedule for cases where peer 1 must block when evaluating its local job allocation (e.g., single or multiprocessor direct interface evaluations). It matches serve_evaluations_peer() for any other processors within the first evaluation partition and serve_evaluations_{synch,asynch}() for all other evaluation partitions (depending on asynchLocalEvalConcurrency). It performs function evaluations locally for its portion of the job allocation using either asynchronous_local_evaluations() or synchronous_local_evaluations(). Single-level and multilevel parallel use intra- and inter-communicators, respectively, for send/receive. Specific syntax is encapsulated within ParallelLibrary. The iteratorCommRank 0 processor assigns the static schedule since it is the only processor with access to beforeSynchCorePRPQueue (it runs the iterator and calls synchronize). The alternate design of each peer selecting its own jobs using the modulus operator would be applicable if execution of this function (and therefore the job list) were distributed.

References ApplicationInterface::asynchLocalEvalConcurrency, ApplicationInterface::asynchronous_local_evaluations(), ApplicationInterface::beforeSynchCorePRPQueue, ApplicationInterface::numEvalServers, Interface::outputLevel, ApplicationInterface::parallelLib, ApplicationInterface::receive_evaluation(), ApplicationInterface::recvBuffers, ApplicationInterface::recvRequests, ApplicationInterface::send_evaluation(), ApplicationInterface::sendBuffers, ApplicationInterface::synchronous_local_evaluations(), and ParallelLibrary::waitall().

Referenced by ApplicationInterface::synch().

void peer_dynamic_schedule_evaluations ( ) [private]

blocking dynamic schedule of all evaluations in beforeSynchCorePRPQueue using message passing on a peer partition; executes on iteratorComm master

This code runs on the iteratorCommRank 0 processor (the iterator) and is called from synch() in order to manage a dynamic schedule, as enabled by nonblocking management of local asynchronous jobs. It matches serve_evaluations_{synch,asynch}() for other evaluation partitions, depending on asynchLocalEvalConcurrency; it does not match serve_evaluations_peer() since, for local asynchronous jobs, the first evaluation partition cannot be multiprocessor. It performs function evaluations locally for its portion of the job allocation using asynchronous_local_evaluations_nowait(). Single-level and multilevel parallel use intra- and inter-communicators, respectively, for send/receive. Specific syntax is encapsulated within ParallelLibrary.

References ApplicationInterface::assign_asynch_local_queue(), ApplicationInterface::asynchLocalEvalConcurrency, ApplicationInterface::beforeSynchCorePRPQueue, ApplicationInterface::msgPassRunningMap, ApplicationInterface::numEvalServers, Interface::outputLevel, ApplicationInterface::recvBuffers, ApplicationInterface::recvRequests, ApplicationInterface::send_evaluation(), ApplicationInterface::sendBuffers, ApplicationInterface::test_local_backfill(), and ApplicationInterface::test_receives_backfill().

Referenced by ApplicationInterface::synch().

void asynchronous_local_evaluations (PRPQueue & *local_prp_queue*) [private]

perform all jobs in prp_queue using asynchronous approaches on the local processor

This function provides blocking synchronization for the local asynch case (background system call, nonblocking fork, or threads). It can be called from synch() for a complete local scheduling of all asynchronous jobs or from peer_{static,dynamic}_schedule_evaluations() to perform a local portion of the total job set. It uses derived_map_asynch() to initiate asynchronous evaluations and wait_local_evaluations() to capture completed jobs, and mirrors the master_dynamic_schedule_evaluations() message passing scheduler as much as possible (wait_local_evaluations() is modeled after MPI_Waitsome()).

References ApplicationInterface::assign_asynch_local_queue(), ApplicationInterface::asynchLocalActivePRPQueue, ApplicationInterface::asynchLocalEvalConcurrency, ApplicationInterface::asynchLocalEvalStatic, ApplicationInterface::completionSet, ApplicationInterface::launch_asynch_local(), ApplicationInterface::localServerAssigned, Dakota::lookup_by_eval_id(), ApplicationInterface::numEvalServers, Interface::outputLevel, ApplicationInterface::process_asynch_local(), Interface::rawResponseMap, and ApplicationInterface::wait_local_evaluations().

Referenced by ApplicationInterface::peer_static_schedule_evaluations(), and ApplicationInterface::synch().

void synchronous_local_evaluations (PRPQueue & *local_prp_queue*) [private]

perform all jobs in prp_queue using synchronous approaches on the local processor

This function provides blocking synchronization for the local synchronous case (foreground system call, blocking fork, or procedure call from derived_map()). It is called from peer_static_schedule_evaluations() to perform a local portion of the total job set.

References ApplicationInterface::broadcast_evaluation(), Interface::currEvalId, ApplicationInterface::derived_map(), ApplicationInterface::manage_failure(), Interface::multiProcEvalFlag, and ApplicationInterface::process_synch_local().

Referenced by ApplicationInterface::peer_static_schedule_evaluations().

void master_dynamic_schedule_evaluations_nowait ( ) [private]

execute a nonblocking dynamic schedule in a master-slave partition

This code is called from synch_nowait() to provide the master portion of a nonblocking master-slave algorithm for the dynamic scheduling of evaluations among slave servers. It performs no evaluations locally and matches either serve_evaluations_synch() or serve_evaluations_asynch() on the slave servers, depending on the value of asynchLocalEvalConcurrency. Dynamic scheduling assigns jobs in 2 passes. The 1st pass gives each server the same number of jobs (equal to asynchLocalEvalConcurrency). The 2nd pass assigns the remaining jobs to slave servers as previous jobs are completed. Single- and multilevel parallel use intra- and inter-communicators, respectively, for send/receive. Specific syntax is encapsulated within ParallelLibrary.

References Dakota::abort_handler(), ApplicationInterface::asynchLocalEvalConcurrency, ApplicationInterface::beforeSynchCorePRPQueue, ApplicationInterface::msgPassRunningMap, ApplicationInterface::numEvalServers, ApplicationInterface::recvBuffers, ApplicationInterface::recvRequests, ApplicationInterface::send_evaluation(), ApplicationInterface::sendBuffers, and ApplicationInterface::test_receives_backfill().

Referenced by ApplicationInterface::synch_nowait().

void peer_dynamic_schedule_evaluations_nowait ( ) [private]

execute a nonblocking static/dynamic schedule in a peer partition

This code runs on the iteratorCommRank 0 processor (the iterator) and is called from synch_nowait() in order to manage a nonblocking static schedule. It matches serve_evaluations_{synch,asynch}() for other evaluation partitions (depending on asynchLocalEvalConcurrency). It performs nonblocking local function evaluations for its portion of the static schedule using asynchronous_local_evaluations(). Single-level and multilevel parallel use intra- and inter-communicators, respectively, for send/receive. Specific syntax is encapsulated within ParallelLibrary. The iteratorCommRank 0 processor assigns the static schedule since it is the only processor with access to beforeSynchCorePRPQueue (it runs the iterator and calls synchronize). The alternate design of each peer selecting its own jobs using the modulus operator would be applicable if execution of this function (and therefore the job list) were distributed.

References Dakota::abort_handler(), ApplicationInterface::assign_asynch_local_queue(), ApplicationInterface::assign_asynch_local_queue_nowait(), ApplicationInterface::asynchLocalActivePRPQueue, ApplicationInterface::asynchLocalEvalConcurrency, ApplicationInterface::beforeSynchCorePRPQueue, Dakota::lookup_by_eval_id(), ApplicationInterface::msgPassRunningMap, ApplicationInterface::numEvalServers, ApplicationInterface::recvBuffers, ApplicationInterface::recvRequests, ApplicationInterface::send_evaluation(), ApplicationInterface::sendBuffers, ApplicationInterface::test_local_backfill(), and ApplicationInterface::test_receives_backfill().

Referenced by ApplicationInterface::synch_nowait().

void asynchronous_local_evaluations_nowait (PRPQueue & *local_prp_queue*) [private]

launch new jobs in prp_queue asynchronously (if capacity is available), perform nonblocking query of all running jobs, and process any completed jobs (handles both local master- and local peer-scheduling cases)

This function provides nonblocking synchronization for the local asynch case (background system call, nonblocking fork, or threads). It is called from synch_nowait() and passed the complete set of all asynchronous jobs (beforeSynchCorePRPQueue). It uses derived_map_asynch() to initiate asynchronous evaluations and test_local_evaluations() to capture completed jobs in nonblocking mode. It mirrors a nonblocking message passing scheduler as much as possible (test_local_evaluations() modeled after MPI_Testsome()). The result of this function is rawResponseMap, which uses eval_id as a key. It is assumed that the incoming local_prp_queue contains only active and new jobs - i.e., all completed jobs are cleared by synch_nowait().

Also supports asynchronous local evaluations with static scheduling. This scheduling policy specifically ensures that a completed asynchronous evaluation eval_id is replaced with an equivalent one, modulo asynchLocalEvalConcurrency. In the nowait case, this could render some servers idle if evaluations don't come in eval_id order or some evaluations are cancelled by the caller in between calls. If this function is called with unlimited local eval concurrency, the static scheduling request is ignored.

References ApplicationInterface::assign_asynch_local_queue_nowait(), ApplicationInterface::asynchLocalActivePRPQueue, ApplicationInterface::asynchLocalEvalConcurrency, ApplicationInterface::asynchLocalEvalStatic, and ApplicationInterface::test_local_backfill().

Referenced by ApplicationInterface::synch_nowait().

void serve_evaluations_synch ( ) [private]

serve the evaluation message passing schedulers and perform one synchronous evaluation at a time

This code is invoked by serve_evaluations() to perform one synchronous job at a time on each slave/peer server. The servers receive requests (blocking receive), do local synchronous maps, and return results. This is done continuously until a termination signal is received from the master (sent via stop_evaluation_servers()).

References ParallelLibrary::bcast_e(), Interface::currEvalId, ApplicationInterface::derived_map(), ApplicationInterface::evalCommRank, ParallelLibrary::isend_ie(), ApplicationInterface::lenResponseMessage, ApplicationInterface::lenVarsActSetMessage, ApplicationInterface::manage_failure(), Interface::multiProcEvalFlag, ApplicationInterface::parallelLib, ParallelLibrary::recv_ie(), MPIPackBuffer::reset(), and ParallelLibrary::wait().

Referenced by ApplicationInterface::serve_evaluations().

void serve_evaluations_synch_peer ( ) [private]

serve the evaluation message passing schedulers and perform one synchronous evaluation at a time as part of the 1st peer

This code is invoked by serve_evaluations() to perform a synchronous evaluation in coordination with the iteratorCommRank 0 processor (the iterator) for static schedules. The bcast() matches either the bcast() in synchronous_local_evaluations(), which is invoked by peer_static_schedule_evaluations(), or the bcast() in map().

References ParallelLibrary::bcast_e(), Interface::currEvalId, ApplicationInterface::derived_map(), ApplicationInterface::lenVarsActSetMessage, ApplicationInterface::manage_failure(), and ApplicationInterface::parallelLib.

Referenced by ApplicationInterface::serve_evaluations().

void serve_evaluations_asynch ( ) [private]

serve the evaluation message passing schedulers and manage multiple asynchronous evaluations

This code is invoked by serve_evaluations() to perform multiple asynchronous jobs on each slave/peer server. The servers test for any incoming jobs, launch any new jobs, process any completed jobs, and return any results. Each of these components is nonblocking, although the server loop continues until a termination signal is received from the master (sent via stop_evaluation_servers()). In the master-slave case, the master maintains the correct number of jobs on each slave. In the static scheduling case, each server is responsible for limiting concurrency (since the entire static schedule is sent to the peers at start up).

References Dakota::abort_handler(), ApplicationInterface::asynchLocalActivePRPQueue, ApplicationInterface::asynchLocalEvalConcurrency, ParallelLibrary::bcast_e(), ApplicationInterface::completionSet, ApplicationInterface::derived_map_asynch(), ApplicationInterface::evalCommRank, Interface::interfaceId, ParallelLibrary::irecv_ie(), ApplicationInterface::lenResponseMessage, ApplicationInterface::lenVarsActSetMessage, Dakota::lookup_by_eval_id(), Interface::multiProcEvalFlag, ApplicationInterface::parallelLib, ParallelLibrary::recv_ie(), MPIUnpackBuffer::reset(), ParallelLibrary::send_ie(), ParallelLibrary::test(), and ApplicationInterface::test_local_evaluations().

Referenced by ApplicationInterface::serve_evaluations().

void serve_evaluations_asynch_peer ( ) [private]

serve the evaluation message passing schedulers and perform multiple asynchronous evaluations as part of the 1st peer

This code is invoked by serve_evaluations() to perform multiple asynchronous jobs on multiprocessor slave/peer servers. It matches the multiProcEvalFlag bcasts in ApplicationInterface::asynchronous_local_evaluations().

References Dakota::abort_handler(), ApplicationInterface::asynchLocalActivePRPQueue, ApplicationInterface::asynchLocalEvalConcurrency, ParallelLibrary::bcast_e(), ApplicationInterface::completionSet, ApplicationInterface::derived_map_asynch(), Interface::interfaceId, ApplicationInterface::lenVarsActSetMessage, Dakota::lookup_by_eval_id(), ApplicationInterface::parallelLib, MPIUnpackBuffer::reset(), and ApplicationInterface::test_local_evaluations().

Referenced by ApplicationInterface::serve_evaluations().

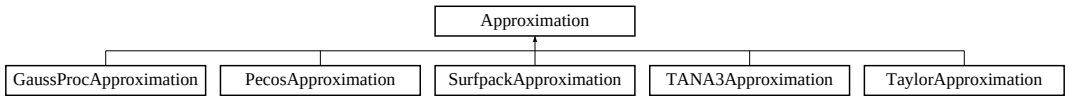
The documentation for this class was generated from the following files:

- ApplicationInterface.hpp
- ApplicationInterface.cpp

13.4 Approximation Class Reference

Base class for the approximation class hierarchy.

Inheritance diagram for Approximation:



Public Member Functions

- Approximation ()

  default constructor
- Approximation (ProblemDescDB &problem_db, const SharedApproxData &shared_data)

  standard constructor for envelope
- Approximation (const SharedApproxData &shared_data)

  alternate constructor
- Approximation (const Approximation &approx)

  copy constructor
- virtual ~Approximation ()

  destructor
- Approximation operator= (const Approximation &approx)

  assignment operator
- virtual void build ()

  builds the approximation from scratch
- virtual void rebuild ()

  rebuilds the approximation incrementally
- virtual void pop (bool save_data)

  removes entries from end of SurrogateData::{vars,resp}Data (last points appended, or as specified in args)

- virtual void restore ()

  restores state prior to previous append()
- virtual void finalize ()

  finalize approximation by applying all remaining trial sets
- virtual void store ()

  store current approximation for later combination
- virtual void combine (short corr_type)

  combine current approximation with previously stored approximation
- virtual Real value (const Variables &vars)

  retrieve the approximate function value for a given parameter vector
- virtual const RealVector & gradient (const Variables &vars)

  retrieve the approximate function gradient for a given parameter vector
- virtual const RealSymMatrix & hessian (const Variables &vars)

  retrieve the approximate function Hessian for a given parameter vector
- virtual Real prediction_variance (const Variables &vars)

  retrieve the variance of the predicted value for a given parameter vector
- virtual bool diagnostics_available ()

  check if diagnostics are available for this approximation type
- virtual Real diagnostic (const String &metric_type)

  retrieve a single diagnostic metric for the diagnostic type specified
- virtual void primary_diagnostics (int fn_index)

  compute and print all requested diagnostics and cross-validation
- virtual void challenge_diagnostics (const RealMatrix &challenge_points, int fn_index)

  compute and print all requested diagnostics for user provided challenge pts
- virtual RealVector approximation_coefficients (bool normalized) const

  return the coefficient array computed by build()/rebuild()
- virtual void approximation_coefficients (const RealVector &approx_coeffs, bool normalized)

  set the coefficient array from external sources, rather than computing with build()/rebuild()
- virtual void coefficient_labels (std::vector< std::string > &coeff_labels) const

  print the coefficient array computed in build()/rebuild()
- virtual void print_coefficients (std::ostream &s, bool normalized)

  print the coefficient array computed in build()/rebuild()
- virtual int min_coefficients () const

  return the minimum number of samples (unknowns) required to build the derived class approximation type in numVars dimensions
- virtual int recommended_coefficients () const

  return the recommended number of samples (unknowns) required to build the derived class approximation type in numVars dimensions
- virtual int num_constraints () const

  return the number of constraints to be enforced via an anchor point
- virtual void clear_current ()

  clear current build data in preparation for next build
- int min_points (bool constraint_flag) const

return the minimum number of points required to build the approximation type in numVars dimensions. Uses ∗_-coefficients() and num_constraints().

- int recommended_points (bool constraint_flag) const

  return the recommended number of samples to build the approximation type in numVars dimensions (default same as min_points)

- const Pecos::SurrogateData & approximation_data () const

  return approxData

- void add (const Pecos::SurrogateDataVars &sdv, bool anchor_flag)

  append to SurrogateData::varsData or assign to SurrogateData::anchorVars

- void add (const Variables &vars, bool anchor_flag, bool deep_copy)

  extract the relevant vectors from Variables and invoke add(RealVector&, IntVector&, RealVector&)

- void add (const Real ∗sample_c_vars, bool anchor_flag, bool deep_copy)

  create a RealVector view and invoke add(RealVector&, empty, empty)

- void add (const RealVector &c_vars, const IntVector &di_vars, const RealVector &dr_vars, bool anchor_flag, bool deep_copy)

  shared code among add(Variables&) and add(Real∗); adds a new data point by either appending to SurrogateData::varsData or assigning to SurrogateData::anchorVars, as dictated by anchor_flag. Uses add_point() and add_anchor().

- void add (const Pecos::SurrogateDataResp &sdr, bool anchor_flag)

  append to SurrogateData::respData or assign to SurrogateData::anchorResp

- void add (const Response &response, int fn_index, bool anchor_flag, bool deep_copy)

  adds a new data point by either appending to SurrogateData::respData or assigning to SurrogateData::anchorResp, as dictated by anchor_flag. Uses add_point() and add_anchor().

- void pop_count (size_t count)

  appends to popCountStack (number of entries to pop from end of SurrogateData::{vars,resp}Data, based on size of last data set appended)

- void clear_all ()

  clear all build data (current and history) to restore original state

- void clear_anchor ()

  clear SurrogateData::anchor{Vars,Resp}

- void clear_data ()

  clear SurrogateData::{vars,resp}Data

- void clear_saved ()

  clear popCountStack and SurrogateData::saved{Vars,Resp}Trials

- void set_bounds (const RealVector &c_l_bnds, const RealVector &c_u_bnds, const IntVector &di_l_bnds, const IntVector &di_u_bnds, const RealVector &dr_l_bnds, const RealVector &dr_u_bnds)

  set approximation lower and upper bounds (currently only used by graphics)

- Approximation ∗ approx_rep () const

  returns approxRep for access to derived class member functions that are not mapped to the top Approximation level

Protected Member Functions

- Approximation (BaseConstructor, const ProblemDescDB &problem_db, const SharedApproxData &shared_data)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
- Approximation (NoDBBaseConstructor, const SharedApproxData &shared_data)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

Protected Attributes

- RealVector approxGradient

  gradient of the approximation returned by gradient()
- RealSymMatrix approxHessian

  Hessian of the approximation returned by hessian()
- Pecos::SurrogateData approxData

  contains the variables/response data for constructing a single approximation model (one response function)
- SharedApproxData * sharedDataRep

  contains the approximation data that is shared among the response set

Private Member Functions

- Approximation * get_approx (ProblemDescDB &problem_db, const SharedApproxData &shared_data)

  Used only by the standard envelope constructor to initialize approxRep to the appropriate derived type.
- Approximation * get_approx (const SharedApproxData &shared_data)

  Used only by the alternate envelope constructor to initialize approxRep to the appropriate derived type.

Private Attributes

- SizetArray popCountStack

  a stack managing the number of points previously added by calls to append() that can be removed by calls to pop()
- Approximation * approxRep

  pointer to the letter (initialized only for the envelope)
- int referenceCount

  number of objects sharing approxRep

13.4.1 Detailed Description

Base class for the approximation class hierarchy.

The Approximation class is the base class for the response data fit approximation class hierarchy in DAKOTA. One instance of an Approximation must be created for each function to be approximated (a vector of Approximations is contained in ApproximationInterface). For memory efficiency and enhanced polymorphism, the approximation hierarchy employs the "letter/envelope idiom" (see Coplien "Advanced C++", p. 133), for which the base class (Approximation) serves as the envelope and one of the derived classes (selected in Approximation::get_approx()) serves as the letter.

13.4.2 Constructor & Destructor Documentation

Approximation ( )

default constructor

The default constructor is used in Array<Approximation> instantiations and by the alternate envelope constructor. approxRep is NULL in this case (problem_db is needed to build a meaningful Approximation object). This makes it necessary to check for NULL in the copy constructor, assignment operator, and destructor.

Approximation (ProblemDescDB & *problem_db*, const SharedApproxData & *shared_data*)

standard constructor for envelope

Envelope constructor only needs to extract enough data to properly execute get_approx, since Approximation(-BaseConstructor, problem_db) builds the actual base class data for the derived approximations.

References Dakota::abort_handler(), Approximation::approxRep, and Approximation::get_approx().

Approximation (const SharedApproxData & *shared_data*)

alternate constructor

This is the alternate envelope constructor for instantiations on the fly. Since it does not have access to problem_db, it utilizes the NoDBBaseConstructor constructor chain.

References Dakota::abort_handler(), Approximation::approxRep, and Approximation::get_approx().

Approximation (const Approximation & *approx*)

copy constructor

Copy constructor manages sharing of approxRep and incrementing of referenceCount.

References Approximation::approxRep, and Approximation::referenceCount.

~Approximation ( ) `[virtual]`

destructor

Destructor decrements referenceCount and only deletes approxRep when referenceCount reaches zero.

References Approximation::approxRep, and Approximation::referenceCount.

Approximation (BaseConstructor, const ProblemDescDB & *problem_db*, const SharedApproxData & *shared_data*) `[protected]`

constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor is the one which must build the base class data for all derived classes. get_approx() instantiates a derived class letter and the derived constructor selects this base class constructor in its initialization list (to avoid recursion in the base class constructor calling get_approx() again). Since the letter IS the representation, its rep pointer is set to NULL (an uninitialized pointer causes problems in ~Approximation).

Approximation (NoDBBaseConstructor, const SharedApproxData & *shared_data*) `[protected]`

constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor is the one which must build the base class data for all derived classes. get_approx() instantiates a derived class letter and the derived constructor selects this base class constructor in its initialization list (to avoid

recursion in the base class constructor calling get_approx() again). Since the letter IS the representation, its rep pointer is set to NULL (an uninitialized pointer causes problems in ~Approximation).

13.4.3 Member Function Documentation

Approximation operator= (const Approximation & *approx*)

assignment operator

Assignment operator decrements referenceCount for old approxRep, assigns new approxRep, and increments referenceCount for new approxRep.

References Approximation::approxRep, and Approximation::referenceCount.

void build ( ) [virtual]

builds the approximation from scratch

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented in PecosApproximation, GaussProcApproximation, SurfpackApproximation, TaylorApproximation, and TANA3Approximation.

References Dakota::abort_handler(), Approximation::approxData, Approximation::approxRep, Approximation::build(), Approximation::min_points(), SharedApproxData::numVars, and Approximation::sharedDataRep.

Referenced by TANA3Approximation::build(), TaylorApproximation::build(), Approximation::build(), SurfpackApproximation::build(), GaussProcApproximation::build(), PecosApproximation::build(), and Approximation::rebuild().

void rebuild ( ) [virtual]

rebuilds the approximation incrementally

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented in PecosApproximation.

References Approximation::approxRep, Approximation::build(), and Approximation::rebuild().

Referenced by Approximation::rebuild().

void pop (bool *save_data*) [virtual]

removes entries from end of SurrogateData::{vars,resp}Data (last points appended, or as specified in args)

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented in PecosApproximation.

References Dakota::abort_handler(), Approximation::approxData, Approximation::approxRep, Approximation::pop(), and Approximation::popCountStack.

Referenced by Approximation::pop(), and PecosApproximation::pop().

void restore ( ) [virtual]

restores state prior to previous append()

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented in PecosApproximation.

References Approximation::approxData, Approximation::approxRep, Approximation::popCountStack, SharedApproxData::restoration_index(), Approximation::restore(), and Approximation::sharedDataRep.

Referenced by Approximation::restore(), and PecosApproximation::restore().

void finalize ( ) [virtual]

finalize approximation by applying all remaining trial sets

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented in PecosApproximation.

References Approximation::approxData, Approximation::approxRep, Approximation::clear_saved(), SharedApproxData::finalization_index(), Approximation::finalize(), and Approximation::sharedDataRep.

Referenced by Approximation::finalize(), and PecosApproximation::finalize().

void clear_current ( ) [inline], [virtual]

clear current build data in preparation for next build

Redefined by TANA3Approximation to clear current data but preserve history.

Reimplemented in TANA3Approximation.

References Approximation::approxRep, Approximation::clear_all(), and Approximation::clear_current().

Referenced by Approximation::clear_current().

void clear_all ( ) [inline]

clear all build data (current and history) to restore original state

Clears out any history (e.g., TANA3Approximation use for a different response function in NonDReliability).

References Approximation::approxData, Approximation::approxRep, and Approximation::clear_all().

Referenced by Approximation::clear_all(), and Approximation::clear_current().

Approximation * get_approx (ProblemDescDB & *problem_db*, const SharedApproxData & *shared_data*) [private]

Used only by the standard envelope constructor to initialize approxRep to the appropriate derived type.

Used only by the envelope constructor to initialize approxRep to the appropriate derived type.

References SharedApproxData::approxType, SharedApproxData::data_rep(), and Dakota::strends().

Referenced by Approximation::Approximation().

Approximation * get_approx (const SharedApproxData & *shared_data*) [private]

Used only by the alternate envelope constructor to initialize approxRep to the appropriate derived type.

Used only by the envelope constructor to initialize approxRep to the appropriate derived type.

References SharedApproxData::approxType, SharedApproxData::data_rep(), and Dakota::strends().

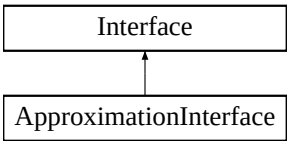
The documentation for this class was generated from the following files:

- DakotaApproximation.hpp
- DakotaApproximation.cpp

13.5 ApproximationInterface Class Reference

Derived class within the interface class hierarchy for supporting approximations to simulation-based results.

Inheritance diagram for ApproximationInterface:



Public Member Functions

- ApproximationInterface (ProblemDescDB &problem_db, const Variables &am_vars, bool am_cache, const String &am_interface_id, size_t num_fns)

  primary constructor
- ApproximationInterface (const String &approx_type, const UShortArray &approx_order, const Variables &am_vars, bool am_cache, const String &am_interface_id, size_t num_fns, short data_order, short output_level)

  alternate constructor for instantiations on the fly
- ~ApproximationInterface ()

  destructor

Protected Member Functions

- void map (const Variables &vars, const ActiveSet &set, Response &response, bool asynch_flag=false)

  the function evaluator: provides an approximate "mapping" from the variables to the responses using functionSurfaces
- int minimum_points (bool constraint_flag) const

  returns the minimum number of samples required to build the functionSurfaces
- int recommended_points (bool constraint_flag) const

  returns the recommended number of samples recommended to build the functionSurfaces
- void approximation_function_indices (const IntSet &approx_fn_indices)

  set the (currently active) approximation function index set
- void update_approximation (const Variables &vars, const IntResponsePair &response_pr)
- void update_approximation (const RealMatrix &samples, const IntResponseMap &resp_map)
- void update_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map)
- void append_approximation (const Variables &vars, const IntResponsePair &response_pr)
- void append_approximation (const RealMatrix &samples, const IntResponseMap &resp_map)
- void append_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map)
- void build_approximation (const RealVector &c_l_bnds, const RealVector &c_u_bnds, const IntVector &di_l_bnds, const IntVector &di_u_bnds, const RealVector &dr_l_bnds, const RealVector &dr_u_bnds)
- void rebuild_approximation (const BoolDeque &rebuild_deque)
- void pop_approximation (bool save_surr_data)
- void restore_approximation ()
- bool restore_available ()

queries the approximation for the ability to restore a previous increment

- void finalize_approximation ()

  finalizes the approximation by applying all trial increments
- void store_approximation ()

  move the current approximation into storage for later combination
- void combine_approximation (short corr_type)

  combine the current approximation with one previously stored
- void clear_current ()

  clears current data from an approximation interface
- void clear_all ()

  clears all data from an approximation interface
- void clear_saved ()

  clears saved data (from pop invocations) from an approximation interface
- SharedApproxData & shared_approximation ()

  retrieve the SharedApproxData within an ApproximationInterface
- std::vector< Approximation > & approximations ()

  retrieve the Approximations within an ApproximationInterface
- const Pecos::SurrogateData & approximation_data (size_t index)

  retrieve the approximation data from a particular Approximation within an ApproximationInterface
- const RealVectorArray & approximation_coefficients (bool normalized=false)

  retrieve the approximation coefficients from each Approximation within an ApproximationInterface
- void approximation_coefficients (const RealVectorArray &approx_coeffs, bool normalized=false)

  set the approximation coefficients within each Approximation within an ApproximationInterface
- const RealVector & approximation_variances (const Variables &vars)

  retrieve the approximation variances from each Approximation within an ApproximationInterface
- const IntResponseMap & synch ()

  recovers data from a series of asynchronous evaluations (blocking)
- const IntResponseMap & synch_nowait ()

  recovers data from a series of asynchronous evaluations (nonblocking)

Private Member Functions

- void mixed_add (const Variables &vars, const Response &response, bool anchor)

  add variables/response data to functionSurfaces using a mixture of shallow and deep copies
- void mixed_add (const Real *c_vars, const Response &response, bool anchor)

  add variables/response data to functionSurfaces using a mixture of shallow and deep copies
- void shallow_add (const Variables &vars, const Response &response, bool anchor)

  add variables/response data to functionSurfaces using a shallow copy
- void sample_to_variables (const Real *sample_c_vars, size_t num_cv, Variables &vars)

  populate continuous variables within vars from sample_c_vars
- void update_pop_counts (const IntResponseMap &resp_map)

  append to the popCountStack within each of the functionSurfaces based on the active set definitions within resp_map
- void read_challenge_points (bool active_only)

  Load approximation test points from user challenge points file.

Private Attributes

- IntSet approxFnIndices

  for incomplete approximation sets, this array specifies the response function subset that is approximated
- SharedApproxData sharedData

  data that is shared among all functionSurfaces
- std::vector< Approximation > functionSurfaces

  list of approximations, one per response function
- RealVectorArray functionSurfaceCoeffs

  array of approximation coefficient vectors, one per response function
- RealVector functionSurfaceVariances

  vector of approximation variances, one value per response function
- String challengeFile

  data file for user-supplied challenge data (per interface, since may contain multiple responses)
- bool challengeAnnotated

  whether the points file is annotated
- bool challengeActiveOnly

  whether to import active only
- RealMatrix challengePoints

  container for the challenge points data
- Variables actualModelVars

  copy of the actualModel variables object used to simplify conversion among differing variable views
- bool actualModelCache

  indicates usage of an evaluation cache by the actualModel
- String actualModelInterfaceId

  the interface id from the actualModel used for ordered PRPCache lookups
- IntResponseMap beforeSynchResponseMap

  bookkeeping map to catalogue responses generated in map() for use in synch() and synch_nowait(). This supports pseudo-asynchronous operations (approximate responses are always computed synchronously, but asynchronous virtual functions are supported through bookkeeping).

Additional Inherited Members

13.5.1 Detailed Description

Derived class within the interface class hierarchy for supporting approximations to simulation-based results.

ApproximationInterface provides an interface class for building a set of global/local/multipoint approximations and performing approximate function evaluations using them. It contains a list of Approximation objects, one for each response function.

13.5.2 Member Function Documentation

void update_approximation (const Variables & *vars*, const IntResponsePair & *response_pr*) `[protected], [virtual]`

This function populates/replaces each Approximation::anchorPoint with the incoming variables/response data point.

Reimplemented from Interface.

References ApproximationInterface::actualModelCache, ApproximationInterface::actualModelInterfaceId, Dakota::data_pairs, Dakota::lookup_by_ids(), ApproximationInterface::mixed_add(), and ApproximationInterface::shallow_add().

void update_approximation (const RealMatrix & *samples,* const IntResponseMap & *resp_map*) [protected], [virtual]

This function populates/replaces each Approximation::currentPoints with the incoming variables/response arrays.

Reimplemented from Interface.

References Dakota::abort_handler(), ApproximationInterface::actualModelCache, ApproximationInterface::actualModelInterfaceId, ApproximationInterface::actualModelVars, ApproximationInterface::approxFnIndices, Dakota::data_pairs, ApproximationInterface::functionSurfaces, Dakota::lookup_by_ids(), ApproximationInterface::mixed_add(), ApproximationInterface::sample_to_variables(), and ApproximationInterface::shallow_add().

void update_approximation (const VariablesArray & *vars_array,* const IntResponseMap & *resp_map*) [protected], [virtual]

This function populates/replaces each Approximation::currentPoints with the incoming variables/response arrays.

Reimplemented from Interface.

References Dakota::abort_handler(), ApproximationInterface::actualModelCache, ApproximationInterface::actualModelInterfaceId, ApproximationInterface::approxFnIndices, Dakota::data_pairs, ApproximationInterface::functionSurfaces, Dakota::lookup_by_ids(), ApproximationInterface::mixed_add(), and ApproximationInterface::shallow_add().

void append_approximation (const Variables & *vars,* const IntResponsePair & *response_pr*) [protected], [virtual]

This function appends to each Approximation::currentPoints with one incoming variables/response data point.

Reimplemented from Interface.

References ApproximationInterface::actualModelCache, ApproximationInterface::actualModelInterfaceId, ApproximationInterface::approxFnIndices, Dakota::data_pairs, ApproximationInterface::functionSurfaces, Dakota::lookup_by_ids(), ApproximationInterface::mixed_add(), and ApproximationInterface::shallow_add().

void append_approximation (const RealMatrix & *samples,* const IntResponseMap & *resp_map*) [protected], [virtual]

This function appends to each Approximation::currentPoints with multiple incoming variables/response data points.

Reimplemented from Interface.

References Dakota::abort_handler(), ApproximationInterface::actualModelCache, ApproximationInterface::actualModelInterfaceId, ApproximationInterface::actualModelVars, Dakota::data_pairs, Dakota::lookup_by_ids(), ApproximationInterface::mixed_add(), ApproximationInterface::sample_to_variables(), ApproximationInterface::shallow_add(), and ApproximationInterface::update_pop_counts().

void append_approximation (const VariablesArray & *vars_array,* const IntResponseMap & *resp_map*) [protected], [virtual]

This function appends to each Approximation::currentPoints with multiple incoming variables/response data points.

Reimplemented from Interface.

References Dakota::abort_handler(), ApproximationInterface::actualModelCache, ApproximationInterface::actualModelInterfaceId, Dakota::data_pairs, Dakota::lookup_by_ids(), ApproximationInterface::mixed_add(), ApproximationInterface::shallow_add(), and ApproximationInterface::update_pop_counts().

void build_approximation (const RealVector & *c_l_bnds,* const RealVector & *c_u_bnds,* const IntVector & *di_l_bnds,* const IntVector & *di_u_bnds,* const RealVector & *dr_l_bnds,* const RealVector & *dr_u_bnds*) `[protected], [virtual]`

This function finds the coefficients for each Approximation based on the data passed through update_approximation() calls. The bounds are used only for graphics visualization.

Reimplemented from Interface.

References ApproximationInterface::approxFnIndices, SharedApproxData::build(), ApproximationInterface::challengeFile, ApproximationInterface::challengePoints, ApproximationInterface::functionSurfaces, ApproximationInterface::read_challenge_points(), SharedApproxData::set_bounds(), and ApproximationInterface::sharedData.

void rebuild_approximation (const BoolDeque & *rebuild_deque*) `[protected], [virtual]`

This function updates the coefficients for each Approximation based on data increments provided by {update,append}_approximation().

Reimplemented from Interface.

References ApproximationInterface::approxFnIndices, ApproximationInterface::functionSurfaces, SharedApproxData::rebuild(), and ApproximationInterface::sharedData.

void pop_approximation (bool *save_surr_data*) `[inline], [protected], [virtual]`

This function removes data provided by a previous append_approximation() call, possibly different numbers for each function, or as specified in pop_count, which is assumed to be the same for all functions.

Reimplemented from Interface.

References ApproximationInterface::approxFnIndices, ApproximationInterface::functionSurfaces, SharedApproxData::pop(), and ApproximationInterface::sharedData.

void restore_approximation ( ) `[inline], [protected], [virtual]`

This function updates the coefficients for each Approximation based on data increments provided by {update,append}_approximation().

Reimplemented from Interface.

References ApproximationInterface::approxFnIndices, ApproximationInterface::functionSurfaces, SharedApproxData::post_restore(), SharedApproxData::pre_restore(), and ApproximationInterface::sharedData.

void read_challenge_points (bool *active_only*) `[private]`

Load approximation test points from user challenge points file.

Challenge data defaults to active/inactive, but user can override to active only.

References ApproximationInterface::actualModelVars, ApproximationInterface::challengeActiveOnly, ApproximationInterface::challengeAnnotated, ApproximationInterface::challengeFile, ApproximationInterface::challengePoints, Variables::copy(), Dakota::copy_data(), Variables::cv(), Variables::div(), Variables::drv(), Variables::dsv(), ApproximationInterface::functionSurfaces, and Variables::tv().

Referenced by ApproximationInterface::build_approximation().

13.5.3 Member Data Documentation

std::vector<Approximation> functionSurfaces `[private]`

list of approximations, one per response function

This formulation allows the use of mixed approximations (i.e., different approximations used for different response functions), although the input specification is not currently general enough to support it.

Referenced by ApproximationInterface::append_approximation(), ApproximationInterface::approximation_coefficients(), ApproximationInterface::approximation_data(), ApproximationInterface::approximation_variances(), ApproximationInterface::ApproximationInterface(), ApproximationInterface::approximations(), ApproximationInterface::build_approximation(), ApproximationInterface::clear_all(), ApproximationInterface::clear_current(), ApproximationInterface::clear_saved(), ApproximationInterface::combine_approximation(), ApproximationInterface::finalize_approximation(), ApproximationInterface::map(), ApproximationInterface::minimum_points(), ApproximationInterface::mixed_add(), ApproximationInterface::pop_approximation(), ApproximationInterface::read_challenge_points(), ApproximationInterface::rebuild_approximation(), ApproximationInterface::recommended_points(), ApproximationInterface::restore_approximation(), ApproximationInterface::shallow_add(), ApproximationInterface::store_approximation(), ApproximationInterface::update_approximation(), and ApproximationInterface::update_pop_counts().

The documentation for this class was generated from the following files:

- ApproximationInterface.hpp
- ApproximationInterface.cpp

13.6 APPSEvalMgr Class Reference

Evaluation manager class for APPSPACK.

Inherits Executor.

Public Member Functions

- APPSEvalMgr (Model &model)

  constructor
- ~APPSEvalMgr ()

  destructor
- bool isReadyForWork () const

  tells APPS whether or not there is a processor available to perform a function evaluation
- bool submit (const int apps_tag, const HOPSPACK::Vector &apps_xtrial, const HOPSPACK::EvalRequestType apps_request)

  performs a function evaluation at APPS-provided x_in
- int recv (int &apps_tag, HOPSPACK::Vector &apps_f, HOPSPACK::Vector &apps_cEqs, HOPSPACK::Vector &apps_cIneqs, string &apps_msg)

  returns a function value to APPS
- std::string getEvaluatorType (void) const

  return the type of the Dakota linked evaluator
- void printDebugInfo (void) const

  empty implementation of debug info needed to complete the interface
- void printTimingInfo (void) const

  empty implementation of timing info needed to complete the interface
- void set_asynch_flag (const bool dakotaAsynchFlag)

publishes whether or not to do asynchronous evaluations

- void set_blocking_synch (const bool blockingSynchFlag)

  publishes whether or not APPS is operating synchronously

- void set_total_workers (const int numDakotaWorkers)

  publishes the number of processors available for function evaluations

- void set_constraint_map (std::vector< int > constraintMapIndices, std::vector< double > constraintMapMultipliers, std::vector< double > constraintMapOffsets)

  publishes constraint transformation

Private Attributes

- Model & iteratedModel

  reference to the APPSOptimizer's model passed in the constructor

- bool modelAsynchFlag

  flag for asynchronous function evaluations

- bool blockingSynch

  flag for APPS synchronous behavior

- int numWorkersUsed

  number of processors actively performing function evaluations

- int numWorkersTotal

  total number of processors available for performing function evaluations

- std::vector< int > constrMapIndices

  map from Dakota constraint number to APPS constraint number

- std::vector< double > constrMapMultipliers

  multipliers for constraint transformations

- std::vector< double > constrMapOffsets

  offsets for constraint transformations

- RealVector xTrial

  trial iterate

- std::map< int, int > tagList

  map of DAKOTA eval id to APPS eval id (for asynchronous evaluations)

- std::map< int, RealVector > functionList

  map of APPS eval id to responses (for synchronous evaluations)

- IntResponseMap dakotaResponseMap

  map of DAKOTA responses returned by synchronize_nowait()

13.6.1 Detailed Description

Evaluation manager class for APPSPACK.

The APPSEvalMgr class is derived from APPSPACK's Executor class. It implements the methods of that class in such away that allows DAKOTA to manage the computation of responses instead of APPS. Iterate and response values are passed between Dakota and APPSPACK via this interface.

13.6.2 Constructor & Destructor Documentation

APPSEvalMgr (Model & *model*)

constructor

Evaluation manager class for APPSPACK.

The APPSEvalMgr class is derived from APPSPACK's Executor class. It implements the methods of that class in such away that allows DAKOTA to manage the computation of responses instead of APPS. Iterate and response values are passed between Dakota and APPSPACK via this interface.

13.6.3 Member Function Documentation

bool isReadyForWork ( ) const

tells APPS whether or not there is a processor available to perform a function evaluation

Check to see if all processors available for function evaluations are being used. If not, tell APPS that one is available.

References APPSEvalMgr::numWorkersTotal, and APPSEvalMgr::numWorkersUsed.

bool submit (const int *apps_tag,* const HOPSPACK::Vector & *apps_xtrial,* const HOPSPACK::EvalRequestType *apps_request*)

performs a function evaluation at APPS-provided x_in

Convert APPSPACK vector of variables to DAKOTA vector of variables and perform function evaluation asynchronously or not as specified in the DAKOTA input deck. If evaluation is asynchronous, map the dakota id to the APPS tag. If evaluation is synchronous, map the responses to the APPS tag.

References Model::asynch_compute_response(), Model::compute_response(), Model::continuous_variables(), Model::current_response(), Model::evaluation_id(), Response::function_values(), APPSEvalMgr::functionList, APPSEvalMgr::iteratedModel, APPSEvalMgr::modelAsynchFlag, APPSEvalMgr::numWorkersTotal, APPSEvalMgr::numWorkersUsed, APPSEvalMgr::tagList, and APPSEvalMgr::xTrial.

int recv (int & *apps_tag,* HOPSPACK::Vector & *apps_f,* HOPSPACK::Vector & *apps_cEqs,* HOPSPACK::Vector & *apps_cIneqs,* string & *apps_msg*)

returns a function value to APPS

Retrieve a set of reponse values, convert to APPS data structures, and return them to APPS. APPS tags are tied to corresponding responses using the appropriate (i.e., asynchronous or synchronous) map.

References APPSEvalMgr::blockingSynch, APPSEvalMgr::constrMapIndices, APPSEvalMgr::constrMapMultipliers, APPSEvalMgr::constrMapOffsets, APPSEvalMgr::dakotaResponseMap, APPSEvalMgr::functionList, APPSEvalMgr::iteratedModel, APPSEvalMgr::modelAsynchFlag, Model::num_nonlinear_eq_constraints(), APPSEvalMgr::numWorkersUsed, Model::primary_response_fn_sense(), Model::synchronize(), Model::synchronize_nowait(), and APPSEvalMgr::tagList.

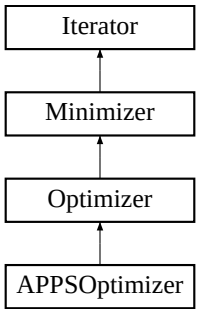
The documentation for this class was generated from the following files:

- APPSEvalMgr.hpp
- APPSEvalMgr.cpp

13.7 APPSOptimizer Class Reference

Wrapper class for HOPSPACK.

Inheritance diagram for APPSOptimizer:



Public Member Functions

- APPSOptimizer (ProblemDescDB &problem_db, Model &model)

  constructor
- APPSOptimizer (Model &model)

  alternate constructor for on-the-fly instantiation without ProblemDescDB
- ~APPSOptimizer ()

  destructor
- void find_optimum ()

  Performs the iterations to determine the optimal solution.

Protected Member Functions

- void set_apps_parameters ()

  sets options for specific methods based on user specifications
- void initialize_variables_and_constraints ()

  initializes problem variables and constraints

Protected Attributes

- HOPSPACK::ParameterList params

  Pointer to APPS parameter list.
- HOPSPACK::ParameterList * problemParams

  Pointer to APPS problem parameter sublist.
- HOPSPACK::ParameterList * linearParams

  Pointer to APPS linear constraint parameter sublist.
- HOPSPACK::ParameterList * mediatorParams

  Pointer to APPS mediator parameter sublist.
- HOPSPACK::ParameterList * citizenParams

  Pointer to APPS citizen/algorithm parameter sublist.
- APPSEvalMgr * evalMgr

  Pointer to the APPS evaluation manager object.
- std::vector< int > constraintMapIndices

  map from Dakota constraint number to APPS constraint number

- std::vector< double > constraintMapMultipliers

  multipliers for constraint transformations

- std::vector< double > constraintMapOffsets

  offsets for constraint transformations

Additional Inherited Members

13.7.1 Detailed Description

Wrapper class for HOPSPACK.

The APPSOptimizer class provides a wrapper for HOPSPACK, a Sandia-developed C++ library for generalized pattern search. HOPSPACK defaults to a coordinate pattern search but also allows for augmented search patterns. It can solve problems with bounds, linear constraints, and general nonlinear constraints. APPSOptimizer uses an APPSEvalMgr object to manage the function evaluations.

The user input mappings are as follows: `output` `max_function_evaluations`, `constraint_tol` `initial_delta`, `contraction_factor`, `threshold_delta`, `solution_target`, `synchronization`, `merit_function`, `constraint_penalty`, and `smoothing_factor` are mapped into HOPS's `"Display"`, "Maximum Evaluations", "Active Tolerance"/"Nonlinear Active Tolerance", "Initial Step", "Contraction Factor", "Step Tolerance", "Objective Target", "Synchronous Evaluations", "Penalty Function", "Penalty Parameter", and "Penalty Smoothing Value" data attributes. Refer to the HOPS web site (https://software.sandia.gov/trac/hopspack) for additional information on HOPS objects and controls.

13.7.2 Member Function Documentation

void find_optimum ( ) [virtual]

Performs the iterations to determine the optimal solution.

find_optimum redefines the Optimizer virtual function to perform the optimization using HOPS. It first sets up the problem data, then executes minimize() on the HOPS optimizer, and finally catalogues the results.

Implements Optimizer.

References Model::asynch_flag(), Iterator::bestResponseArray, Iterator::bestVariablesArray, APPSOptimizer::constraintMapIndices, APPSOptimizer::constraintMapMultipliers, APPSOptimizer::constraintMapOffsets, APPSOptimizer::evalMgr, Model::evaluation_capacity(), APPSOptimizer::initialize_variables_and_constraints(), Iterator::iteratedModel, Optimizer::localObjectiveRecast, Minimizer::numContinuousVars, Minimizer::numFunctions, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, APPSOptimizer::params, Model::primary_response_fn_sense(), APPSEvalMgr::set_asynch_flag(), and APPSEvalMgr::set_total_workers().

void set_apps_parameters ( ) [protected]

sets options for specific methods based on user specifications

Set all of the HOPS algorithmic parameters as specified in the DAKOTA input deck. This is called at construction time.

References APPSOptimizer::citizenParams, Minimizer::constraintTol, APPSOptimizer::evalMgr, ProblemDescDB::get_real(), ProblemDescDB::get_string(), ProblemDescDB::is_null(), APPSOptimizer::linearParams, Iterator::maxEvalConcurrency, Iterator::maxFunctionEvals, APPSOptimizer::mediatorParams, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, Iterator::outputLevel, APPSOptimizer::params, Iterator::probDescDB, APPSOptimizer::problemParams, and APPSEvalMgr::set_blocking_synch().

Referenced by APPSOptimizer::APPSOptimizer().

void initialize_variables_and_constraints ( ) [protected]

initializes problem variables and constraints

Set the variables and constraints as specified in the DAKOTA input deck. This is done at run time.

References Minimizer::bigRealBoundSize, APPSOptimizer::constraintMapIndices, APPSOptimizer::constraintMapMultipliers, APPSOptimizer::constraintMapOffsets, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), APPSOptimizer::evalMgr, Iterator::iteratedModel, Model::linear_eq_constraint_coeffs(), Model::linear_eq_constraint_targets(), Model::linear_ineq_constraint_coeffs(), Model::linear_ineq_constraint_lower_bounds(), Model::linear_ineq_constraint_upper_bounds(), APPSOptimizer::linearParams, Model::nonlinear_eq_constraint_targets(), Model::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Minimizer::numContinuousVars, Minimizer::numLinearEqConstraints, Minimizer::numLinearIneqConstraints, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, APPSOptimizer::problemParams, and APPSEvalMgr::set_constraint_map().

Referenced by APPSOptimizer::find_optimum().

The documentation for this class was generated from the following files:

- APPSOptimizer.hpp
- APPSOptimizer.cpp

13.8 BaseConstructor Struct Reference

Dummy struct for overloading letter-envelope constructors.

Public Member Functions

- BaseConstructor (int=0)

  C++ structs can have constructors.

13.8.1 Detailed Description

Dummy struct for overloading letter-envelope constructors.

BaseConstructor is used to overload the constructor for the base class portion of letter objects. It avoids infinite recursion (Coplien p.139) in the letter-envelope idiom by preventing the letter from instantiating another envelope. Putting this struct here avoids circular dependencies.

The documentation for this struct was generated from the following file:

- dakota_global_defs.hpp

13.9 callback_data Struct Reference

Public Attributes

- double rosen_cdv_upper_bd

  upper bound value to pass through parser to callback function

13.9.1 Detailed Description

Data structure to pass application-specific values through Dakota back to the callback function, for example to convey late updates to bounds, initial points, etc., to Dakota.

The documentation for this struct was generated from the following file:

- library_mode.cpp

13.10 COLINApplication Class Reference

Inherits Application< colin::MO_MINLP2_problem >.

Public Member Functions

- COLINApplication ()

  Default constructor. Rrequired by COLIN's ApplicationHandle creation.
- COLINApplication (Model &model)

  Constructor with Model (not presently used).
- ~COLINApplication ()

  Destructor.
- void set_problem (Model &model)

  Helper function called after default construction to extract problem information from the Model and set it for COLIN.
- void set_blocking_synch (const bool blockingSynchFlag)

  publishes whether or not COLIN is operating synchronously
- virtual utilib::Any spawn_evaluation_impl (const utilib::Any &domain, const colin::AppRequest::request_map_t &requests, utilib::seed_t &seed)

  Schedule one or more requests at specified domain point, returning a DAKOTA-specific evaluation tracking ID.
- virtual bool evaluation_available ()

  Check to see if there are any function values ready to be collected.
- virtual void perform_evaluation_impl (const utilib::Any &domain, const colin::AppRequest::request_map_t &requests, utilib::seed_t &seed, colin::AppResponse::response_map_t &colin_responses)

  Perform a function evaluation at t given point.
- virtual utilib::Any collect_evaluation_impl (colin::AppResponse::response_map_t &responses, utilib::seed_t &seed)

  Collect a completed evaluation from DAKOTA.
- virtual void colin_request_to_dakota_request (const utilib::Any &domain, const colin::AppRequest::request_map_t &requests, utilib::seed_t &seed)

  Helper function to convert evaluation request data from COLIN structures to DAKOTA structures.
- virtual void dakota_response_to_colin_response (const Response &dakota_response, colin::AppResponse::response_map_t &colin_responses)

  Gelper function to convert evaluation response data from DAKOTA structures to COLIN structures.
- virtual bool map_domain (const utilib::Any &src, utilib::Any &native, bool forward=true) const

  Map the domain point into data type desired by this application context.

Protected Attributes

- Model iteratedModel

  Shallow copy of the model on which COLIN will iterate.
- bool blockingSynch

  Flag for COLIN synchronous behavior (Pattern Search only).
- ActiveSet activeSet

  Local copy of model's active set for convenience.
- std::vector< int > requestedEvals

  Evaluations queued for asynch evaluation.
- IntResponseMap dakota_responses

  eval_id to response mapping to cache completed jobs.

13.10.1 Detailed Description

COLINApplication is a DAKOTA class that is derived from COLIN's Application hierarchy. It redefines a variety of virtual COLIN functions to use the corresponding DAKOTA functions. This is a more flexible algorithm library interfacing approach than can be obtained with the function pointer approaches used by NPSOLOptimizer and SNLLOptimizer.

13.10.2 Member Function Documentation

void set_problem (Model & *model*)

Helper function called after default construction to extract problem information from the Model and set it for COLIN.

Set variable bounds and linear and nonlinear constraints. This avoids using probDescDB, so it is called by both the standard and the on-the-fly COLINOptimizer constructors.

References Response::active_set(), COLINApplication::activeSet, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::current_response(), Model::cv(), Model::discrete_int_lower_bounds(), Model::discrete_int_sets(), Model::discrete_int_upper_bounds(), Model::discrete_set_int_values(), Model::discrete_set_real_values(), Model::div(), Model::drv(), COLINApplication::iteratedModel, Model::linear_eq_constraint_coeffs(), Model::linear_eq_constraint_targets(), Model::linear_ineq_constraint_coeffs(), Model::linear_ineq_constraint_lower_bounds(), Model::linear_ineq_constraint_upper_bounds(), Model::nonlinear_eq_constraint_targets(), Model::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Model::num_functions(), Model::num_linear_eq_constraints(), Model::num_linear_ineq_constraints(), Model::num_nonlinear_eq_constraints(), Model::num_nonlinear_ineq_constraints(), and Model::primary_response_fn_sense().

Referenced by COLINApplication::COLINApplication().

utilib::Any spawn_evaluation_impl (const utilib::Any & *domain,* const colin::AppRequest::request_map_t & *requests,* utilib::seed_t & *seed*) `[virtual]`

Schedule one or more requests at specified domain point, returning a DAKOTA-specific evaluation tracking ID.

Schedule one or more requests at specified domain point, returning a DAKOTA-specific evaluation tracking ID. This is only called by COLIN's concurrent evaluator, which is only instantiated when the Model supports asynch evals. The domain point is guaranteed to be compatible with data type specified by map_domain(...)

References Model::asynch_compute_response(), COLINApplication::colin_request_to_dakota_request(), Model::evaluation_id(), and COLINApplication::iteratedModel.

bool evaluation_available ( ) [virtual]

Check to see if there are any function values ready to be collected.

Check to see if any asynchronous evaluations have finished. This is only called by COLIN's concurrent evaluator, which is only instantiated when the Model supports asynch evals.

References COLINApplication::blockingSynch, COLINApplication::dakota_responses, COLINApplication::iteratedModel, Model::synchronize(), and Model::synchronize_nowait().

void perform_evaluation_impl (const utilib::Any & *domain,* const colin::AppRequest::request_map_t & *requests,* utilib::seed_t & *seed,* colin::AppResponse::response_map_t & *colin_responses*) [virtual]

Perform a function evaluation at t given point.

Perform an evaluation at a specified domain point. Wait for and return the response. This is only called by COLIN's serial evaluator, which is only instantiated when the Model does not support asynch evals. The domain point is guaranteed to be compatible with data type specified by map_domain(...)

References COLINApplication::colin_request_to_dakota_request(), Model::compute_response(), Model::current_response(), COLINApplication::dakota_response_to_colin_response(), and COLINApplication::iteratedModel.

utilib::Any collect_evaluation_impl (colin::AppResponse::response_map_t & *colin_responses,* utilib::seed_t & *seed*) [virtual]

Collect a completed evaluation from DAKOTA.

Collect the next completed evaluation from DAKOTA. Always returns the evalid of the response returned.

References COLINApplication::dakota_response_to_colin_response(), and COLINApplication::dakota_responses.

void colin_request_to_dakota_request (const utilib::Any & *domain,* const colin::AppRequest::request_map_t & *requests,* utilib::seed_t & *seed*) [virtual]

Helper function to convert evaluation request data from COLIN structures to DAKOTA structures.

Map COLIN info requests to DAKOTA objectives and constraints.

References Model::continuous_variables(), Model::discrete_int_sets(), Model::discrete_int_variable(), Model::discrete_real_variable(), Model::discrete_set_int_values(), Model::discrete_set_real_values(), Model::div(), Model::drv(), COLINApplication::iteratedModel, Model::num_functions(), and Dakota::set_index_to_value().

Referenced by COLINApplication::perform_evaluation_impl(), and COLINApplication::spawn_evaluation_impl().

void dakota_response_to_colin_response (const Response & *dakota_response,* colin::AppResponse::response_map_t & *colin_responses*) [virtual]

Gelper function to convert evaluation response data from DAKOTA structures to COLIN structures.

Map DAKOTA objective and constraint values to COLIN response.

References Response::active_set_request_vector(), and Response::function_value().

Referenced by COLINApplication::collect_evaluation_impl(), and COLINApplication::perform_evaluation_impl().

bool map_domain (const utilib::Any & *src,* utilib::Any & *native,* bool *forward* = `true`) const [virtual]

Map the domain point into data type desired by this application context.

Map the domain point into data type desired by this application context (utilib::MixedIntVars). This data type can be exposed from the Any &domain presented to spawn and collect.

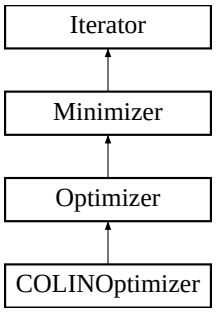
The documentation for this class was generated from the following files:

- COLINApplication.hpp
- COLINApplication.cpp

13.11 COLINOptimizer Class Reference

Wrapper class for optimizers defined using COLIN.

Inheritance diagram for COLINOptimizer:



Public Member Functions

- COLINOptimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor
- COLINOptimizer (const String &method_name, Model &model, int seed, int max_iter, int max_eval)

  alternate constructor for on-the-fly instantiations
- COLINOptimizer (const String &method_name, Model &model)

  alternate constructor for Iterator instantiations by name
- ~COLINOptimizer ()

  destructor
- void reset ()

  clears internal optimizer state
- void find_optimum ()

  iterates the COLIN solver to determine the optimal solution
- bool returns_multiple_points () const

  some COLIN methods can return multiple points

Protected Member Functions

- void solver_setup (unsigned short method_name)

  convenience function for setting up the particular COLIN solver and appropriate Application
- void set_rng (int seed)

  sets up the random number generator for stochastic methods
- void set_solver_parameters ()

sets construct-time options for specific methods based on user specifications, including calling method-specific set functions

- void post_run (std::ostream &s)

  Get the final set of points from the solver Look up responses and sort, first according to constraint violation, then according to function value.
- std::pair< bool, bool > colin_cache_lookup (const colin::AppResponse &colinResponse, Response &tmpResponseHolder)

  Retrieve response from Colin AppResponse, return pair indicating success for <objective, constraints>
- double constraint_violation (const Response &tmpResponseHolder)

  Compute constraint violation, based on nonlinear constraints in iteratedModel and provided Response data.

Protected Attributes

- short solverType

  COLIN solver sub-type as enumerated in COLINOptimizer.cpp.
- colin::SolverHandle colinSolver

  handle to the COLIN solver
- std::pair
  < colin::ApplicationHandle,
  COLINApplication ∗ > colinProblem

  handle and pointer to the COLINApplication object
- colin::EvaluationManager_Base ∗ colinEvalMgr

  pointer to the COLIN evalutaion manager object
- utilib::RNG ∗ rng

  random number generator pointer
- bool blockingSynch

  the `synchronization` setting: true if `blocking`, false if `nonblocking`
- Real constraint_penalty

  Buffer to hold problem constraint_penalty parameter.
- bool constant_penalty

  Buffer to hold problem constant_penalty parameter.

Additional Inherited Members

13.11.1 Detailed Description

Wrapper class for optimizers defined using COLIN.

The COLINOptimizer class wraps COLIN, a Sandia-developed C++ optimization interface library. A variety of COLIN optimizers are defined in COLIN and its associated libraries, including SCOLIB which contains the optimization components from the old COLINY (formerly SGOPT) library. COLIN contains optimizers such as genetic algorithms, pattern search methods, and other nongradient-based techniques. COLINOptimizer uses a COLINApplication object to perform the function evaluations.

The user input mappings are as follows: `max_iterations`, `max_function_evaluations`, `convergence_tolerance`, and `solution_accuracy` are mapped into COLIN's `max_iterations`, `max_function_evaluations_this_trial`, `function_value_tolerance`, `sufficient_objective_value` properties. An `outputLevel` is mapped to COLIN's `output_level` property and a setting of `debug` activates output of method initialization and sets the COLIN `debug` attribute to 10000 for the DEBUG output level. Refer to [Hart, W.E., 2006] for additional information on COLIN objects and controls.

13.11.2 Constructor & Destructor Documentation

COLINOptimizer (ProblemDescDB & *problem_db,* Model & *model*)

standard constructor

Standard constructor.

References ProblemDescDB::get_int(), ProblemDescDB::get_ushort(), Iterator::probDescDB, COLINOptimizer::set_rng(), COLINOptimizer::set_solver_parameters(), and COLINOptimizer::solver_setup().

COLINOptimizer (const String & *method_name,* Model & *model,* int *seed,* int *max_iter,* int *max_eval*)

alternate constructor for on-the-fly instantiations

Alternate constructor for on-the-fly instantiations.

References Iterator::maxFunctionEvals, Iterator::maxIterations, Iterator::method_string_to_enum(), COLINOptimizer::set_rng(), COLINOptimizer::set_solver_parameters(), and COLINOptimizer::solver_setup().

COLINOptimizer (const String & *method_name,* Model & *model*)

alternate constructor for Iterator instantiations by name

Alternate constructor for Iterator instantiations by name.

References Iterator::method_string_to_enum(), COLINOptimizer::set_solver_parameters(), and COLINOptimizer::solver_setup().

13.11.3 Member Function Documentation

void find_optimum ( ) `[virtual]`

iterates the COLIN solver to determine the optimal solution

find_optimum redefines the Optimizer virtual function to perform the optimization using COLIN. It first sets up the problem data, then executes optimize() on the COLIN solver and finally catalogues the results.

Implements Optimizer.

References Dakota::_NPOS, Dakota::abort_handler(), Model::asynch_flag(), COLINOptimizer::blockingSynch, COLINOptimizer::colinEvalMgr, COLINOptimizer::colinProblem, COLINOptimizer::colinSolver, COLINOptimizer::constant_penalty, COLINOptimizer::constraint_penalty, Model::continuous_variables(), Model::discrete_int_sets(), Model::discrete_int_variables(), Model::discrete_real_variables(), Model::discrete_set_int_values(), Model::discrete_set_real_values(), Model::evaluation_capacity(), Iterator::iteratedModel, Minimizer::numDiscreteIntVars, Minimizer::numDiscreteRealVars, Iterator::outputLevel, Dakota::set_value_to_index(), and COLINOptimizer::solverType.

bool returns_multiple_points ( ) const `[virtual]`

some COLIN methods can return multiple points

Designate which solvers can return multiple final points.

Reimplemented from Iterator.

References COLINOptimizer::solverType.

void solver_setup (unsigned short *method_name*) `[protected]`

convenience function for setting up the particular COLIN solver and appropriate Application

This convenience function is called by the constructors in order to instantiate the solver.

References COLINOptimizer::colinProblem, COLINOptimizer::colinSolver, COLINOptimizer::constant_penalty, COLINOptimizer::constraint_penalty, ProblemDescDB::get_string(), Iterator::method_enum_to_string(), Iterator::probDescDB, and COLINOptimizer::solverType.

Referenced by COLINOptimizer::COLINOptimizer().

void set_rng (int *seed*) [protected]

sets up the random number generator for stochastic methods

Instantiate random number generator (RNG).

References COLINOptimizer::colinSolver, and COLINOptimizer::rng.

Referenced by COLINOptimizer::COLINOptimizer().

void set_solver_parameters ( ) [protected]

sets construct-time options for specific methods based on user specifications, including calling method-specific set functions

Sets solver properties based on user specifications. Called at construction time.

References Model::asynch_flag(), COLINOptimizer::blockingSynch, COLINOptimizer::colinSolver, COLINOptimizer::constant_penalty, COLINOptimizer::constraint_penalty, Iterator::convergenceTol, ProblemDescDB::get_bool(), ProblemDescDB::get_int(), ProblemDescDB::get_real(), ProblemDescDB::get_sa(), ProblemDescDB::get_string(), ProblemDescDB::is_null(), Iterator::iteratedModel, Iterator::maxEvalConcurrency, Iterator::maxFunctionEvals, Iterator::maxIterations, Minimizer::numContinuousVars, Iterator::outputLevel, Iterator::probDescDB, and COLINOptimizer::solverType.

Referenced by COLINOptimizer::COLINOptimizer().

void post_run (std::ostream & *s*) [protected], [virtual]

Get the final set of points from the solver Look up responses and sort, first according to constraint violation, then according to function value.

Supplement Optimizer::post_run to first retrieve points from the Colin cache (or possibly the Dakota DB) and rank them. When complete, this function will populate bestVariablesArray and bestResponsesArray with iterator-space data, that is, in the context of the solver, leaving any further untransformation to Optimizer.

Reimplemented from Iterator.

References Iterator::bestResponseArray, Iterator::bestVariablesArray, COLINOptimizer::colin_cache_lookup(), COLINOptimizer::colinProblem, COLINOptimizer::colinSolver, COLINOptimizer::constraint_violation(), Variables::continuous_variables(), Response::copy(), Variables::copy(), Model::current_response(), Model::current_variables(), Model::discrete_int_sets(), Variables::discrete_int_variable(), Variables::discrete_real_variable(), Model::discrete_set_int_values(), Model::discrete_set_real_values(), Response::function_values(), Iterator::iteratedModel, Optimizer::localObjectiveRecast, Minimizer::numDiscreteIntVars, Minimizer::numDiscreteRealVars, Iterator::numFinalSolutions, Optimizer::numObjectiveFns, Minimizer::objective(), Optimizer::post_run(), Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), Minimizer::resize_best_resp_array(), Minimizer::resize_best_vars_array(), Dakota::set_index_to_value(), and Model::subordinate_model().

std::pair< bool, bool > colin_cache_lookup (const colin::AppResponse & *colinResponse,* Response & *tmpResponseHolder*) [protected]

Retrieve response from Colin AppResponse, return pair indicating success for <objective, constraints>

Encapsulated Colin Cache response extraction, which will ultimately become the default lookup. Might want to return separate vectors of function values and constraints for use in the sort, but not for now (least change). Return true if not needed or successful lookup.

References Response::function_value(), Minimizer::numNonlinearConstraints, and Optimizer::numObjectiveFns.

Referenced by COLINOptimizer::post_run().

double constraint_violation (const Response & *tmpResponseHolder*) [protected]

Compute constraint violation, based on nonlinear constraints in iteratedModel and provided Response data.

BMA TODO: incorporate constraint tolerance, possibly via elevating SurrBasedMinimizer::constraint_violation(). Always use iteratedModel to get the constraints; they are in the right space.

References Response::function_values(), Iterator::iteratedModel, Model::nonlinear_eq_constraint_targets(), Model::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Model::num_nonlinear_eq_constraints(), Model::num_nonlinear_ineq_constraints(), and Minimizer::numIterPrimaryFns.

Referenced by COLINOptimizer::post_run().

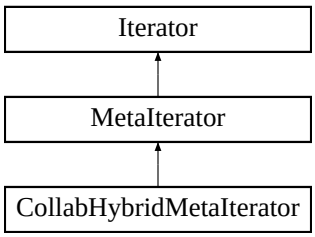
The documentation for this class was generated from the following files:

- COLINOptimizer.hpp
- COLINOptimizer.cpp

13.12 CollabHybridMetaIterator Class Reference

Meta-iterator for hybrid iteration using multiple collaborating optimization and nonlinear least squares methods.

Inheritance diagram for CollabHybridMetaIterator:



Public Member Functions

- CollabHybridMetaIterator (ProblemDescDB &problem_db)

  standard constructor
- CollabHybridMetaIterator (ProblemDescDB &problem_db, Model &model)

  alternate constructor
- ~CollabHybridMetaIterator ()

  destructor

Protected Member Functions

- void core_run ()

  Performs the collaborative hybrid iteration.
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance

- const Variables & variables_results () const

  return the final solution from the collaborative iteration (variables)
- const Response & response_results () const

  return the final solution from the collaborative iteration (response)

Private Attributes

- String hybridCollabType

  abo or hops
- StringArray methodList

  the list of method name identifiers
- bool lightwtCtor

  use of lightweight Iterator construction by name
- IteratorArray selectedIterators

  the set of iterators, one for each entry in methodList
- ModelArray selectedModels

  the set of models, one for each iterator
- Variables bestVariables

  best variables found in collaborative iteration
- Response bestResponse

  best response found in collaborative iteration

Additional Inherited Members

13.12.1 Detailed Description

Meta-iterator for hybrid iteration using multiple collaborating optimization and nonlinear least squares methods.

This meta-iterator has two approaches to hybrid iteration: (1) agent-based using the ABO framework; (2) nonagent-based using the HOPSPACK framework.

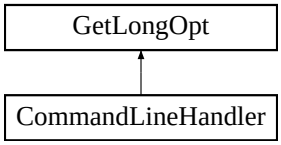
The documentation for this class was generated from the following files:

- CollabHybridMetaIterator.hpp
- CollabHybridMetaIterator.cpp

13.13 CommandLineHandler Class Reference

Utility class for managing command line inputs to DAKOTA.

Inheritance diagram for CommandLineHandler:



Public Member Functions

- CommandLineHandler ()

  default constructor, requires check_usage() call for parsing
- CommandLineHandler (int argc, char **argv, int world_rank)

  constructor with parsing
- ~CommandLineHandler ()

  destructor
- void check_usage (int argc, char **argv)

  Verifies that DAKOTA is called with the correct command usage. Prints a descriptive message and exits the program if incorrect.
- int read_restart_evals () const

  Returns the number of evaluations to be read from the restart file (as specified on the DAKOTA command line) as an integer instead of a const char.*
- void usage (std::ostream &outfile=Cout) const

  Print usage information to outfile, conditionally on rank.

Private Member Functions

- void initialize_options ()

  enrolls the supported command line inputs.
- void output_helper (const std::string &message, std::ostream &os) const

  output only on Dakota worldRank 0 if possible

Private Attributes

- int worldRank

  Rank of this process within Dakota's allocation; manages conditional output.

Additional Inherited Members

13.13.1 Detailed Description

Utility class for managing command line inputs to DAKOTA.

CommandLineHandler provides additional functionality that is specific to DAKOTA's needs for the definition and parsing of command line options. Inheritance is used to allow the class to have all the functionality of the base class, GetLongOpt.

13.13.2 Member Function Documentation

void output_helper (const std::string & *message*, std::ostream & *os*) const `[private]`

output only on Dakota worldRank 0 if possible

When there is a valid ParallelLibrary, output only on rank 0

References CommandLineHandler::worldRank.

Referenced by CommandLineHandler::check_usage().

The documentation for this class was generated from the following files:

- CommandLineHandler.hpp
- CommandLineHandler.cpp

13.14 CommandShell Class Reference

Utility class which defines convenience operators for spawning processes with system calls.

Public Member Functions

- CommandShell ()

  constructor
- ~CommandShell ()

  destructor
- CommandShell & operator<< (const char *cmd)

  appends cmd to sysCommand
- CommandShell & operator<< (const std::string &cmd)

  convenient operator: appends string to the commandString to be executed
- CommandShell & operator<< (CommandShell &(*f)(CommandShell &))

  allows passing of the flush function to the shell using <<
- CommandShell & flush ()

  "flushes" the shell; i.e. executes the sysCommand
- void asynch_flag (const bool flag)

  set the asynchFlag
- bool asynch_flag () const

  get the asynchFlag
- void suppress_output_flag (const bool flag)

  set the suppressOutputFlag
- bool suppress_output_flag () const

  get the suppressOutputFlag

Private Attributes

- std::string sysCommand

  The command string that is constructed through one or more << insertions and then executed by flush.
- bool asynchFlag

  flags nonblocking operation (background system calls)
- bool suppressOutputFlag

  flags suppression of shell output (no command echo)

13.14.1 Detailed Description

Utility class which defines convenience operators for spawning processes with system calls.

The CommandShell class wraps the C system() utility and defines convenience operators for building a command string and then passing it to the shell.

13.14.2 Member Function Documentation

CommandShell & operator<< (const char * *cmd*) [inline]

appends cmd to sysCommand

convenient operator: appends string to the commandString to be executed

References CommandShell::sysCommand.

CommandShell & operator<< (CommandShell &(*)(CommandShell &) *f*) [inline]

allows passing of the flush function to the shell using <<

convenience operator: allows passing of the flush func to the shell via <<

CommandShell & flush ( )

"flushes" the shell; i.e. executes the sysCommand

Executes the sysCommand by passing it to system(). Appends an "&" if asynchFlag is set (background system call) and echos the sysCommand to Cout if suppressOutputFlag is not set.

References Dakota::abort_handler(), CommandShell::asynchFlag, CommandShell::suppressOutputFlag, and CommandShell::sysCommand.

Referenced by Dakota::flush().

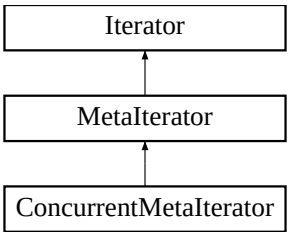
The documentation for this class was generated from the following files:

- CommandShell.hpp
- CommandShell.cpp

13.15 ConcurrentMetaIterator Class Reference

Meta-iterator for multi-start iteration or pareto set optimization.

Inheritance diagram for ConcurrentMetaIterator:



Public Member Functions

- ConcurrentMetaIterator (ProblemDescDB &problem_db)

  standard constructor
- ConcurrentMetaIterator (ProblemDescDB &problem_db, Model &model)

  alternate constructor
- ~ConcurrentMetaIterator ()

  destructor

Protected Member Functions

- void pre_run ()

  pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

- void core_run ()

  Performs the concurrent iteration by executing selectedIterator on iteratedModel multiple times in parallel for different parameter sets.

- void print_results (std::ostream &s)

  print the final iterator results

- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance

- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance

- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance

- void initialize_iterator (int job_index)

  used by IteratorScheduler to set the starting data for a run

- void pack_parameters_buffer (MPIPackBuffer &send_buffer, int job_index)

  used by IteratorScheduler to pack starting data for an iterator run

- void unpack_parameters_initialize (MPIUnpackBuffer &recv_buffer)

  used by IteratorScheduler to unpack starting data and initialize an iterator run

- void pack_results_buffer (MPIPackBuffer &send_buffer, int job_index)

  used by IteratorScheduler to pack results data from an iterator run

- void unpack_results_buffer (MPIUnpackBuffer &recv_buffer, int job_index)

  used by IteratorScheduler to unpack results data from an iterator run

- void update_local_results (int job_index)

  used by IteratorScheduler to update local results arrays

- const Model & algorithm_space_model () const

  return the result of any recasting or surrogate model recursion layered on top of iteratedModel by the derived Iterator ctor chain

Private Member Functions

- void initialize_iterator (const RealVector &param_set)

  called by unpack_parameters_initialize(MPIUnpackBuffer) and initialize_iterator(int) to update iteratedModel and selectedIterator

- void initialize_model ()

  initialize the iterated Model prior to Iterator instantiation and define param_set_len

Private Attributes

- Iterator selectedIterator

  the iterator selected for concurrent iteration
- bool lightwtCtor

  use of lightweight Iterator construction by name
- RealVector initialPt

  the initial continuous variables for restoring the starting point in the Pareto set minimization
- RealVectorArray parameterSets

  an array of parameter set vectors (either multistart variable sets or pareto multi-objective/least squares weighting sets) to be performed.
- int paramSetLen

  length of each of the parameter sets associated with an iterator job (number of continuous variables for MULTI_START, number of objective fns for PARETO_SET)
- int numRandomJobs

  number of randomly-generated parameter sets to evaluate
- int randomSeed

  seed for random number generator for random samples
- PRPArray prpResults

  1-d array of ParamResponsePair results corresponding to numIteratorJobs

Friends

- class IteratorScheduler

  protect scheduler callback functions from general access

Additional Inherited Members

13.15.1 Detailed Description

Meta-iterator for multi-start iteration or pareto set optimization.

This meta-iterator maintains two concurrent iterator capabilities. First, a general capability for running an iterator multiple times from different starting points is provided (often used for multi-start optimization, but not restricted to optimization). Second, a simple capability for mapping the "pareto frontier" (the set of optimal solutions in multiobjective formulations) is provided. This pareto set is mapped through running an optimizer multiple times for different sets of multiobjective weightings.

13.15.2 Member Function Documentation

void pre_run ( ) `[protected], [virtual]`

pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

pre-run phase, which a derived iterator may optionally reimplement; when not present, pre-run is likely integrated into the derived run function. This is a virtual function; when re-implementing, a derived class must call its nearest parent's pre_run(), if implemented, typically *before* performing its own implementation steps.

Reimplemented from Iterator.

References Analyzer::all_samples(), Iterator::all_samples(), ParallelLibrary::bcast_hs(), Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), Dakota::copy_data(), Model::estimate_message_lengths(), ConcurrentMetaIterator::initialPt, Iterator::iteratedModel, IteratorScheduler::iterator_message_lengths(), IteratorScheduler::iteratorCommRank, IteratorScheduler::iteratorScheduling, IteratorScheduler::iteratorServerId, MetaIterator::iterSched, IteratorScheduler::lead_rank(), Model::message_lengths(), Iterator::methodName, Iterator::methodPCIter, IteratorScheduler::miPLIndex, IteratorScheduler::numIteratorJobs, IteratorScheduler::numIteratorServers, ConcurrentMetaIterator::numRandomJobs, Iterator::parallelLib, ConcurrentMetaIterator::parameterSets, ConcurrentMetaIterator::paramSetLen, ConcurrentMetaIterator::prpResults, ConcurrentMetaIterator::randomSeed, and MPIPackBuffer::size().

void print_results (std::ostream & *s*) [protected], [virtual]

print the final iterator results

This virtual function provides additional iterator-specific final results outputs beyond the function evaluation summary printed in finalize_run().

Reimplemented from Iterator.

References Variables::continuous_variables(), Variables::discrete_int_variables(), Variables::discrete_real_variables(), ParamResponsePair::eval_id(), Iterator::methodName, ConcurrentMetaIterator::parameterSets, ConcurrentMetaIterator::paramSetLen, ParamResponsePair::prp_parameters(), ParamResponsePair::prp_response(), ConcurrentMetaIterator::prpResults, and Response::write_tabular().

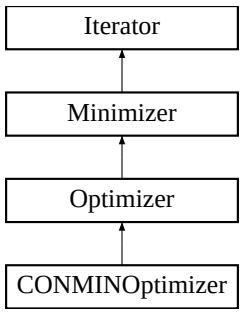
The documentation for this class was generated from the following files:

- ConcurrentMetaIterator.hpp
- ConcurrentMetaIterator.cpp

13.16 CONMINOptimizer Class Reference

Wrapper class for the CONMIN optimization library.

Inheritance diagram for CONMINOptimizer:



Public Member Functions

- CONMINOptimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor
- CONMINOptimizer (const String &method_string, Model &model)

  alternate constructor; construct without ProblemDescDB
- ~CONMINOptimizer ()

destructor

- void find_optimum ()

  Used within the optimizer branch for computing the optimal solution. Redefines the run virtual function for the optimizer branch.

Protected Member Functions

- void initialize_run ()

  performs run-time set up

Private Member Functions

- void initialize ()

  Shared constructor code.
- void allocate_workspace ()

  Allocates workspace for the optimizer.
- void deallocate_workspace ()

  Releases workspace memory.
- void allocate_constraints ()

  Allocates constraint mappings.

Private Attributes

- int conminInfo

  INFO from CONMIN manual.
- int printControl

  IPRINT from CONMIN manual (controls output verbosity)
- Real objFnValue

  value of the objective function passed to CONMIN
- RealVector constraintValues

  array of nonlinear constraint values passed to CONMIN
- int numConminNlnConstr

  total number of nonlinear constraints seen by CONMIN
- int numConminLinConstr

  total number of linear constraints seen by CONMIN
- int numConminConstr

  total number of linear and nonlinear constraints seen by CONMIN
- SizetArray constraintMappingIndices

  a container of indices for referencing the corresponding Response constraints used in computing the CONMIN constraints.
- RealArray constraintMappingMultipliers

  a container of multipliers for mapping the Response constraints to the CONMIN constraints.
- RealArray constraintMappingOffsets

  a container of offsets for mapping the Response constraints to the CONMIN constraints.
- int N1

Size variable for CONMIN arrays. See CONMIN manual.

- int N2

  Size variable for CONMIN arrays. See CONMIN manual.
- int N3

  Size variable for CONMIN arrays. See CONMIN manual.
- int N4

  Size variable for CONMIN arrays. See CONMIN manual.
- int N5

  Size variable for CONMIN arrays. See CONMIN manual.
- int NFDG

  Finite difference flag.
- int IPRINT

  Flag to control amount of output data.
- int ITMAX

  Flag to specify the maximum number of iterations.
- double FDCH

  Relative finite difference step size.
- double FDCHM

  Absolute finite difference step size.
- double CT

  Constraint thickness parameter.
- double CTMIN

  Minimum absolute value of CT used during optimization.
- double CTL

  Constraint thickness parameter for linear and side constraints.
- double CTLMIN

  Minimum value of CTL used during optimization.
- double DELFUN

  Relative convergence criterion threshold.
- double DABFUN

  Absolute convergence criterion threshold.
- double ∗ conminDesVars

  Array of design variables used by CONMIN (length N1 = numdv+2)
- double ∗ conminLowerBnds

  Array of lower bounds used by CONMIN (length N1 = numdv+2)
- double ∗ conminUpperBnds

  Array of upper bounds used by CONMIN (length N1 = numdv+2)
- double ∗ S

  Internal CONMIN array.
- double ∗ G1

  Internal CONMIN array.
- double ∗ G2

  Internal CONMIN array.

- double * B

  Internal CONMIN array.
- double * C

  Internal CONMIN array.
- int * MS1

  Internal CONMIN array.
- double * SCAL

  Internal CONMIN array.
- double * DF

  Internal CONMIN array.
- double * A

  Internal CONMIN array.
- int * ISC

  Internal CONMIN array.
- int * IC

  Internal CONMIN array.

Additional Inherited Members

13.16.1 Detailed Description

Wrapper class for the CONMIN optimization library.

The CONMINOptimizer class provides a wrapper for CONMIN, a Public-domain Fortran 77 optimization library written by Gary Vanderplaats under contract to NASA Ames Research Center. The CONMIN User's Manual is contained in NASA Technical Memorandum X-62282, 1978. CONMIN uses a reverse communication mode, which avoids the static member function issues that arise with function pointer designs (see NPSOLOptimizer and SNLLOptimizer).

The user input mappings are as follows: `max_iterations` is mapped into CONMIN's `ITMAX` parameter, `max_function_evaluations` is implemented directly in the find_optimum() loop since there is no CONMIN parameter equivalent, `convergence_tolerance` is mapped into CONMIN's `DELFUN` and `DABFUN` parameters, `output` verbosity is mapped into CONMIN's `IPRINT` parameter (verbose: `IPRINT` = 4; quiet: `IPRINT` = 2), gradient mode is mapped into CONMIN's `NFDG` parameter, and finite difference step size is mapped into CONMIN's `FDCH` and `FDCHM` parameters. Refer to [Vanderplaats, 1978] for additional information on CONMIN parameters.

13.16.2 Member Data Documentation

int conminInfo `[private]`

INFO from CONMIN manual.

Information requested by CONMIN: 1 = evaluate objective and constraints, 2 = evaluate gradients of objective and constraints.

Referenced by CONMINOptimizer::find_optimum(), and CONMINOptimizer::initialize().

int printControl [private]

IPRINT from CONMIN manual (controls output verbosity)

Values range from 0 (nothing) to 4 (most output). 0 = nothing, 1 = initial and final function information, 2 = all of #1 plus function value and design vars at each iteration, 3 = all of #2 plus constraint values and direction vectors, 4 = all of #3 plus gradients of the objective function and constraints, 5 = all of #4 plus proposed design vector, plus objective and constraint functions from the 1-D search

Referenced by CONMINOptimizer::initialize().

RealVector constraintValues [private]

array of nonlinear constraint values passed to CONMIN

This array must be of nonzero length and must contain only one-sided inequality constraints which are <= 0 (which requires a transformation from 2-sided inequalities and equalities).

Referenced by CONMINOptimizer::allocate_workspace(), and CONMINOptimizer::find_optimum().

SizetArray constraintMappingIndices [private]

a container of indices for referencing the corresponding Response constraints used in computing the CONMIN constraints.

The length of the container corresponds to the number of CONMIN constraints, and each entry in the container points to the corresponding DAKOTA constraint.

Referenced by CONMINOptimizer::allocate_constraints(), and CONMINOptimizer::find_optimum().

RealArray constraintMappingMultipliers [private]

a container of multipliers for mapping the Response constraints to the CONMIN constraints.

The length of the container corresponds to the number of CONMIN constraints, and each entry in the container stores a multiplier for the DAKOTA constraint identified with constraintMappingIndices. These multipliers are currently +1 or -1.

Referenced by CONMINOptimizer::allocate_constraints(), and CONMINOptimizer::find_optimum().

RealArray constraintMappingOffsets [private]

a container of offsets for mapping the Response constraints to the CONMIN constraints.

The length of the container corresponds to the number of CONMIN constraints, and each entry in the container stores an offset for the DAKOTA constraint identified with constraintMappingIndices. These offsets involve inequality bounds or equality targets, since CONMIN assumes constraint allowables = 0.

Referenced by CONMINOptimizer::allocate_constraints(), and CONMINOptimizer::find_optimum().

int N1 [private]

Size variable for CONMIN arrays. See CONMIN manual.

N1 = number of variables + 2

Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::find_optimum(), and CONMINOptimizer::initialize_run().

int N2 [private]

Size variable for CONMIN arrays. See CONMIN manual.

N2 = number of constraints + 2∗(number of variables)

Referenced by CONMINOptimizer::allocate_workspace(), and CONMINOptimizer::find_optimum().

int N3 [private]

Size variable for CONMIN arrays. See CONMIN manual.
N3 = Maximum possible number of active constraints.
Referenced by CONMINOptimizer::allocate_workspace(), and CONMINOptimizer::find_optimum().

int N4 [private]

Size variable for CONMIN arrays. See CONMIN manual.
N4 = Maximum(N3,number of variables)
Referenced by CONMINOptimizer::allocate_workspace(), and CONMINOptimizer::find_optimum().

int N5 [private]

Size variable for CONMIN arrays. See CONMIN manual.
N5 = 2∗(N4)
Referenced by CONMINOptimizer::allocate_workspace(), and CONMINOptimizer::find_optimum().

double CT [private]

Constraint thickness parameter.
The value of CT decreases in magnitude during optimization.
Referenced by CONMINOptimizer::find_optimum(), and CONMINOptimizer::initialize().

double∗ S [private]

Internal CONMIN array.
Move direction in N-dimensional space.
Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), and CONMINOptimizer::find_optimum().

double∗ G1 [private]

Internal CONMIN array.
Temporary storage of constraint values.
Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), and CONMINOptimizer::find_optimum().

double∗ G2 [private]

Internal CONMIN array.
Temporary storage of constraint values.
Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), and CONMINOptimizer::find_optimum().

double∗ B [private]

Internal CONMIN array.
Temporary storage for computations involving array S.
Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), and CONMINOptimizer::find_optimum().

double* C [private]

Internal CONMIN array.

Temporary storage for use with arrays B and S.

Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), and CONMINOptimizer::find_optimum().

int* MS1 [private]

Internal CONMIN array.

Temporary storage for use with arrays B and S.

Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), and CONMINOptimizer::find_optimum().

double* SCAL [private]

Internal CONMIN array.

Vector of scaling parameters for design parameter values.

Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), and CONMINOptimizer::find_optimum().

double* DF [private]

Internal CONMIN array.

Temporary storage for analytic gradient data.

Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), and CONMINOptimizer::find_optimum().

double* A [private]

Internal CONMIN array.

Temporary 2-D array for storage of constraint gradients.

Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), and CONMINOptimizer::find_optimum().

int* ISC [private]

Internal CONMIN array.

Array of flags to identify linear constraints. (not used in this implementation of CONMIN)

Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), CONMINOptimizer::find_optimum(), and CONMINOptimizer::initialize_run().

int* IC [private]

Internal CONMIN array.

Array of flags to identify active and violated constraints

Referenced by CONMINOptimizer::allocate_workspace(), CONMINOptimizer::deallocate_workspace(), CONMINOptimizer::find_optimum(), and CONMINOptimizer::initialize_run().

The documentation for this class was generated from the following files:

- CONMINOptimizer.hpp
- CONMINOptimizer.cpp

13.17 ConsoleRedirector Class Reference

Public Member Functions

- ConsoleRedirector (std::ostream *&dakota_stream, std::ostream *default_dest)

  Constructor taking a reference to the Dakota Cout/Cerr handle and a default destination to use when no redirection (or destruct)
- ~ConsoleRedirector ()

  when the redirector stack is destroyed, it will rebind the output handle to the default ostream, then destroy open files
- void push_back ()

  push back the default or repeat the last pushed file stream
- void push_back (const String &filename)

  push back a new output filestream, or repeat the last one if no filename change
- void pop_back ()

  pop the last redirection

Protected Attributes

- std::ostream *& ostreamHandle

  The handle (target ostream) through which output is sent; typically dakota_cout or dakota_cerr. Will be rebound to specific streams as they are pushed or popped.
- std::ostream * defaultOStream

  initial stream to reset to when redirections are done (typically std::cout or std::cerr)
- std::vector< boost::shared_ptr < OutputWriter > > ostreamDestinations

  stack of redirections to OutputWriters; shared pointers are used to potentially share the same ostream at multiple levels

Private Member Functions

- ConsoleRedirector ()

  default constructor is disallowed
- ConsoleRedirector (const ConsoleRedirector &)

  copy constructor is disallowed due
- const ConsoleRedirector & operator= (const ConsoleRedirector &)

  assignment is disallowed

13.17.1 Detailed Description

Component to manage a set of output or error redirections. Push operations may present a new filename, or none in order to preserve current binding to cout/cerr or file, but place an entry on the stack. Cout/Cerr are rebound as needed when a stream is destroyed on pop.

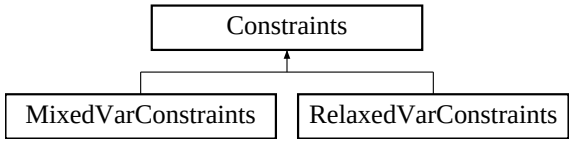
The documentation for this class was generated from the following files:

- OutputManager.hpp
- OutputManager.cpp

13.18 Constraints Class Reference

Base class for the variable constraints class hierarchy.

Inheritance diagram for Constraints:



Public Member Functions

- Constraints ()

  default constructor
- Constraints (const ProblemDescDB &prob_db, const SharedVariablesData &svd)

  standard constructor
- Constraints (const SharedVariablesData &svd)

  alternate constructor for instantiations on the fly
- Constraints (const Constraints &con)

  copy constructor
- virtual ~Constraints ()

  destructor
- Constraints operator= (const Constraints &con)

  assignment operator
- virtual void write (std::ostream &s) const

  write a variable constraints object to an std::ostream
- virtual void read (std::istream &s)

  read a variable constraints object from an std::istream
- const RealVector & continuous_lower_bounds () const

  return the active continuous variable lower bounds
- void continuous_lower_bounds (const RealVector &cl_bnds)

  set the active continuous variable lower bounds
- void continuous_lower_bound (Real cl_bnd, size_t i)

  set an active continuous variable lower bound
- const RealVector & continuous_upper_bounds () const

  return the active continuous variable upper bounds
- void continuous_upper_bounds (const RealVector &cu_bnds)

  set the active continuous variable upper bounds
- void continuous_upper_bound (Real cu_bnd, size_t i)

  set an active continuous variable upper bound
- const IntVector & discrete_int_lower_bounds () const

  return the active discrete variable lower bounds
- void discrete_int_lower_bounds (const IntVector &dil_bnds)

set the active discrete variable lower bounds

- void discrete_int_lower_bound (int dil_bnd, size_t i)

  set an active discrete variable lower bound
- const IntVector & discrete_int_upper_bounds () const

  return the active discrete variable upper bounds
- void discrete_int_upper_bounds (const IntVector &diu_bnds)

  set the active discrete variable upper bounds
- void discrete_int_upper_bound (int diu_bnd, size_t i)

  set an active discrete variable upper bound
- const RealVector & discrete_real_lower_bounds () const

  return the active discrete variable lower bounds
- void discrete_real_lower_bounds (const RealVector &drl_bnds)

  set the active discrete variable lower bounds
- void discrete_real_lower_bound (Real drl_bnd, size_t i)

  set an active discrete variable lower bound
- const RealVector & discrete_real_upper_bounds () const

  return the active discrete variable upper bounds
- void discrete_real_upper_bounds (const RealVector &dru_bnds)

  set the active discrete variable upper bounds
- void discrete_real_upper_bound (Real dru_bnd, size_t i)

  set an active discrete variable upper bound
- const RealVector & inactive_continuous_lower_bounds () const

  return the inactive continuous lower bounds
- void inactive_continuous_lower_bounds (const RealVector &icl_bnds)

  set the inactive continuous lower bounds
- const RealVector & inactive_continuous_upper_bounds () const

  return the inactive continuous upper bounds
- void inactive_continuous_upper_bounds (const RealVector &icu_bnds)

  set the inactive continuous upper bounds
- const IntVector & inactive_discrete_int_lower_bounds () const

  return the inactive discrete lower bounds
- void inactive_discrete_int_lower_bounds (const IntVector &idil_bnds)

  set the inactive discrete lower bounds
- const IntVector & inactive_discrete_int_upper_bounds () const

  return the inactive discrete upper bounds
- void inactive_discrete_int_upper_bounds (const IntVector &idiu_bnds)

  set the inactive discrete upper bounds
- const RealVector & inactive_discrete_real_lower_bounds () const

  return the inactive discrete lower bounds
- void inactive_discrete_real_lower_bounds (const RealVector &idrl_bnds)

  set the inactive discrete lower bounds
- const RealVector & inactive_discrete_real_upper_bounds () const

  return the inactive discrete upper bounds

- void inactive_discrete_real_upper_bounds (const RealVector &idru_bnds)

  set the inactive discrete upper bounds
- const RealVector & all_continuous_lower_bounds () const

  returns a single array with all continuous lower bounds
- void all_continuous_lower_bounds (const RealVector &acl_bnds)

  sets all continuous lower bounds using a single array
- void all_continuous_lower_bound (Real acl_bnd, size_t i)

  set a lower bound within the all continuous lower bounds array
- const RealVector & all_continuous_upper_bounds () const

  returns a single array with all continuous upper bounds
- void all_continuous_upper_bounds (const RealVector &acu_bnds)

  sets all continuous upper bounds using a single array
- void all_continuous_upper_bound (Real acu_bnd, size_t i)

  set an upper bound within the all continuous upper bounds array
- const IntVector & all_discrete_int_lower_bounds () const

  returns a single array with all discrete lower bounds
- void all_discrete_int_lower_bounds (const IntVector &adil_bnds)

  sets all discrete lower bounds using a single array
- void all_discrete_int_lower_bound (int adil_bnd, size_t i)

  set a lower bound within the all discrete lower bounds array
- const IntVector & all_discrete_int_upper_bounds () const

  returns a single array with all discrete upper bounds
- void all_discrete_int_upper_bounds (const IntVector &adiu_bnds)

  sets all discrete upper bounds using a single array
- void all_discrete_int_upper_bound (int adiu_bnd, size_t i)

  set an upper bound within the all discrete upper bounds array
- const RealVector & all_discrete_real_lower_bounds () const

  returns a single array with all discrete lower bounds
- void all_discrete_real_lower_bounds (const RealVector &adrl_bnds)

  sets all discrete lower bounds using a single array
- void all_discrete_real_lower_bound (Real adrl_bnd, size_t i)

  set a lower bound within the all discrete lower bounds array
- const RealVector & all_discrete_real_upper_bounds () const

  returns a single array with all discrete upper bounds
- void all_discrete_real_upper_bounds (const RealVector &adru_bnds)

  sets all discrete upper bounds using a single array
- void all_discrete_real_upper_bound (Real adru_bnd, size_t i)

  set an upper bound within the all discrete upper bounds array
- size_t num_linear_ineq_constraints () const

  return the number of linear inequality constraints
- size_t num_linear_eq_constraints () const

  return the number of linear equality constraints
- const RealMatrix & linear_ineq_constraint_coeffs () const

return the linear inequality constraint coefficients

- void linear_ineq_constraint_coeffs (const RealMatrix &lin_ineq_coeffs)

  set the linear inequality constraint coefficients
- const RealVector & linear_ineq_constraint_lower_bounds () const

  return the linear inequality constraint lower bounds
- void linear_ineq_constraint_lower_bounds (const RealVector &lin_ineq_l_bnds)

  set the linear inequality constraint lower bounds
- const RealVector & linear_ineq_constraint_upper_bounds () const

  return the linear inequality constraint upper bounds
- void linear_ineq_constraint_upper_bounds (const RealVector &lin_ineq_u_bnds)

  set the linear inequality constraint upper bounds
- const RealMatrix & linear_eq_constraint_coeffs () const

  return the linear equality constraint coefficients
- void linear_eq_constraint_coeffs (const RealMatrix &lin_eq_coeffs)

  set the linear equality constraint coefficients
- const RealVector & linear_eq_constraint_targets () const

  return the linear equality constraint targets
- void linear_eq_constraint_targets (const RealVector &lin_eq_targets)

  set the linear equality constraint targets
- size_t num_nonlinear_ineq_constraints () const

  return the number of nonlinear inequality constraints
- size_t num_nonlinear_eq_constraints () const

  return the number of nonlinear equality constraints
- const RealVector & nonlinear_ineq_constraint_lower_bounds () const

  return the nonlinear inequality constraint lower bounds
- void nonlinear_ineq_constraint_lower_bounds (const RealVector &nln_ineq_l_bnds)

  set the nonlinear inequality constraint lower bounds
- const RealVector & nonlinear_ineq_constraint_upper_bounds () const

  return the nonlinear inequality constraint upper bounds
- void nonlinear_ineq_constraint_upper_bounds (const RealVector &nln_ineq_u_bnds)

  set the nonlinear inequality constraint upper bounds
- const RealVector & nonlinear_eq_constraint_targets () const

  return the nonlinear equality constraint targets
- void nonlinear_eq_constraint_targets (const RealVector &nln_eq_targets)

  set the nonlinear equality constraint targets
- Constraints copy () const

  for use when a deep copy is needed (the representation is not shared)
- void shape ()

  shape the lower/upper bound arrays based on sharedVarsData
- void reshape (size_t num_nln_ineq_cons, size_t num_nln_eq_cons, size_t num_lin_ineq_cons, size_t num_lin_eq_cons, const SharedVariablesData &svd)

  reshape the linear/nonlinear/bound constraint arrays arrays and the lower/upper bound arrays
- void reshape ()

reshape the lower/upper bound arrays based on sharedVarsData

- void reshape (size_t num_nln_ineq_cons, size_t num_nln_eq_cons, size_t num_lin_ineq_cons, size_t num_lin_eq_cons)

  reshape the linear/nonlinear constraint arrays
- void inactive_view (short view2)

  sets the inactive view based on higher level (nested) context
- bool is_null () const

  function to check constraintsRep (does this envelope contain a letter)

Protected Member Functions

- Constraints (BaseConstructor, const ProblemDescDB &problem_db, const SharedVariablesData &svd)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
- Constraints (BaseConstructor, const SharedVariablesData &svd)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
- void build_views ()

  construct active/inactive views of all variables arrays
- void build_active_views ()

  construct active views of all variables bounds arrays
- void build_inactive_views ()

  construct inactive views of all variables bounds arrays
- void manage_linear_constraints (const ProblemDescDB &problem_db)

  perform checks on user input, convert linear constraint coefficient input to matrices, and assign defaults

Protected Attributes

- SharedVariablesData sharedVarsData

  configuration data shared from a Variables instance
- RealVector allContinuousLowerBnds

  a continuous lower bounds array combining continuous design, uncertain, and continuous state variable types (all view).
- RealVector allContinuousUpperBnds

  a continuous upper bounds array combining continuous design, uncertain, and continuous state variable types (all view).
- IntVector allDiscreteIntLowerBnds

  a discrete lower bounds array combining discrete design and discrete state variable types (all view).
- IntVector allDiscreteIntUpperBnds

  a discrete upper bounds array combining discrete design and discrete state variable types (all view).
- RealVector allDiscreteRealLowerBnds

  a discrete lower bounds array combining discrete design and discrete state variable types (all view).
- RealVector allDiscreteRealUpperBnds

  a discrete upper bounds array combining discrete design and discrete state variable types (all view).
- size_t numNonlinearIneqCons

number of nonlinear inequality constraints

- size_t numNonlinearEqCons

  number of nonlinear equality constraints
- RealVector nonlinearIneqConLowerBnds

  nonlinear inequality constraint lower bounds
- RealVector nonlinearIneqConUpperBnds

  nonlinear inequality constraint upper bounds
- RealVector nonlinearEqConTargets

  nonlinear equality constraint targets
- size_t numLinearIneqCons

  number of linear inequality constraints
- size_t numLinearEqCons

  number of linear equality constraints
- RealMatrix linearIneqConCoeffs

  linear inequality constraint coefficients
- RealMatrix linearEqConCoeffs

  linear equality constraint coefficients
- RealVector linearIneqConLowerBnds

  linear inequality constraint lower bounds
- RealVector linearIneqConUpperBnds

  linear inequality constraint upper bounds
- RealVector linearEqConTargets

  linear equality constraint targets
- RealVector continuousLowerBnds

  the active continuous lower bounds array view
- RealVector continuousUpperBnds

  the active continuous upper bounds array view
- IntVector discreteIntLowerBnds

  the active discrete lower bounds array view
- IntVector discreteIntUpperBnds

  the active discrete upper bounds array view
- RealVector discreteRealLowerBnds

  the active discrete lower bounds array view
- RealVector discreteRealUpperBnds

  the active discrete upper bounds array view
- RealVector inactiveContinuousLowerBnds

  the inactive continuous lower bounds array view
- RealVector inactiveContinuousUpperBnds

  the inactive continuous upper bounds array view
- IntVector inactiveDiscreteIntLowerBnds

  the inactive discrete lower bounds array view
- IntVector inactiveDiscreteIntUpperBnds

  the inactive discrete upper bounds array view

- RealVector inactiveDiscreteRealLowerBnds

  the inactive discrete lower bounds array view
- RealVector inactiveDiscreteRealUpperBnds

  the inactive discrete upper bounds array view

Private Member Functions

- Constraints ∗ get_constraints (const ProblemDescDB &problem_db, const SharedVariablesData &svd)

  Used only by the constructor to initialize constraintsRep to the appropriate derived type.
- Constraints ∗ get_constraints (const SharedVariablesData &svd) const

  Used by copy() to initialize constraintsRep to the appropriate derived type.

Private Attributes

- Constraints ∗ constraintsRep

  pointer to the letter (initialized only for the envelope)
- int referenceCount

  number of objects sharing constraintsRep

13.18.1 Detailed Description

Base class for the variable constraints class hierarchy.

The Constraints class is the base class for the class hierarchy managing bound, linear, and nonlinear constraints. Using the variable lower and upper bounds arrays from the input specification, different derived classes define different views of this data. The linear and nonlinear constraint data is consistent in all views and is managed at the base class level. For memory efficiency and enhanced polymorphism, the variable constraints hierarchy employs the "letter/envelope idiom" (see Coplien "Advanced C++", p. 133), for which the base class (Constraints) serves as the envelope and one of the derived classes (selected in Constraints::get_constraints()) serves as the letter.

13.18.2 Constructor & Destructor Documentation

Constraints ( )

default constructor

The default constructor: constraintsRep is NULL in this case (a populated problem_db is needed to build a meaningful Constraints object). This makes it necessary to check for NULL in the copy constructor, assignment operator, and destructor.

Constraints (const ProblemDescDB & *problem_db*, const SharedVariablesData & *svd*)

standard constructor

The envelope constructor only needs to extract enough data to properly execute get_constraints, since the constructor overloaded with BaseConstructor builds the actual base class data inherited by the derived classes.

References Dakota::abort_handler(), Constraints::constraintsRep, and Constraints::get_constraints().

Constraints (const SharedVariablesData & *svd*)

alternate constructor for instantiations on the fly

Envelope constructor for instantiations on the fly. This constructor executes get_constraints(view), which invokes the default derived/base constructors, followed by a reshape() based on vars_comps.

References Dakota::abort_handler(), Constraints::constraintsRep, and Constraints::get_constraints().

Constraints (const Constraints & *con*)

copy constructor

Copy constructor manages sharing of constraintsRep and incrementing of referenceCount.

References Constraints::constraintsRep, and Constraints::referenceCount.

~Constraints ( ) `[virtual]`

destructor

Destructor decrements referenceCount and only deletes constraintsRep when referenceCount reaches zero.

References Constraints::constraintsRep, and Constraints::referenceCount.

Constraints (BaseConstructor , const ProblemDescDB & *problem_db,* const SharedVariablesData & *svd*) `[protected]`

constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor is the one which must build the base class data for all derived classes. get_constraints() instantiates a derived class letter and the derived constructor selects this base class constructor in its initialization list (to avoid recursion in the base class constructor calling get_constraints() again). Since the letter IS the representation, its rep pointer is set to NULL (an uninitialized pointer causes problems in ~Constraints).

References Constraints::build_views(), Constraints::manage_linear_constraints(), and Constraints::shape().

Constraints (BaseConstructor , const SharedVariablesData & *svd*) `[protected]`

constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor is the one which must build the base class data for all derived classes. get_constraints() instantiates a derived class letter and the derived constructor selects this base class constructor in its initialization list (to avoid recursion in the base class constructor calling get_constraints() again). Since the letter IS the representation, its rep pointer is set to NULL (an uninitialized pointer causes problems in ~Constraints).

References Constraints::build_views(), and Constraints::shape().

13.18.3 Member Function Documentation

Constraints operator= (const Constraints & *con*)

assignment operator

Assignment operator decrements referenceCount for old constraintsRep, assigns new constraintsRep, and increments referenceCount for new constraintsRep.

References Constraints::constraintsRep, and Constraints::referenceCount.

Constraints copy ( ) const

for use when a deep copy is needed (the representation is *not* shared)

Deep copies are used for history mechanisms that catalogue permanent copies (should not change as the representation within userDefinedConstraints changes).

References Constraints::allContinuousLowerBnds, Constraints::allContinuousUpperBnds, Constraints::allDiscreteIntLowerBnds, Constraints::allDiscreteIntUpperBnds, Constraints::allDiscreteRealLowerBnds, Constraints::allDiscreteRealUpperBnds, Constraints::build_views(), Constraints::constraintsRep, Constraints::get_constraints(), Constraints::linearEqConCoeffs, Constraints::linearEqConTargets, Constraints::linearIneqConCoeffs, Constraints::linearIneqConLowerBnds, Constraints::linearIneqConUpperBnds, Constraints::nonlinearEqConTargets, Constraints::nonlinearIneqConLowerBnds, Constraints::nonlinearIneqConUpperBnds, Constraints::numLinearEqCons, Constraints::numLinearIneqCons, Constraints::numNonlinearEqCons, Constraints::numNonlinearIneqCons, and Constraints::sharedVarsData.

Referenced by SurrogateModel::force_rebuild(), and RecastModel::RecastModel().

void shape ( )

shape the lower/upper bound arrays based on sharedVarsData

Resizes the derived bounds arrays.

References SharedVariablesData::all_counts(), Constraints::allContinuousLowerBnds, Constraints::allContinuousUpperBnds, Constraints::allDiscreteIntLowerBnds, Constraints::allDiscreteIntUpperBnds, Constraints::allDiscreteRealLowerBnds, Constraints::allDiscreteRealUpperBnds, Constraints::constraintsRep, Constraints::shape(), and Constraints::sharedVarsData.

Referenced by Constraints::Constraints(), and Constraints::shape().

void reshape (size_t *num_nln_ineq_cons*, size_t *num_nln_eq_cons*, size_t *num_lin_ineq_cons*, size_t *num_lin_eq_cons*)

reshape the linear/nonlinear constraint arrays

Resizes the linear and nonlinear constraint arrays at the base class. Does NOT currently resize the derived bounds arrays.

References Constraints::constraintsRep, Constraints::continuousLowerBnds, Constraints::discreteIntLowerBnds, Constraints::discreteRealLowerBnds, Constraints::linearEqConCoeffs, Constraints::linearEqConTargets, Constraints::linearIneqConCoeffs, Constraints::linearIneqConLowerBnds, Constraints::linearIneqConUpperBnds, Constraints::nonlinearEqConTargets, Constraints::nonlinearIneqConLowerBnds, Constraints::nonlinearIneqConUpperBnds, Constraints::numLinearEqCons, Constraints::numLinearIneqCons, Constraints::numNonlinearEqCons, Constraints::numNonlinearIneqCons, and Constraints::reshape().

void build_views ( ) `[inline], [protected]`

construct active/inactive views of all variables arrays

= EMPTY)

= EMPTY)

References Constraints::build_active_views(), Constraints::build_inactive_views(), Constraints::sharedVarsData, and SharedVariablesData::view().

Referenced by Constraints::Constraints(), Constraints::copy(), and Constraints::reshape().

void manage_linear_constraints (const ProblemDescDB & *problem_db*) `[protected]`

perform checks on user input, convert linear constraint coefficient input to matrices, and assign defaults

Convenience function called from derived class constructors. The number of variables active for applying linear constraints is currently defined to be the number of active continuous variables plus the number of active discrete variables (the most general case), even though very few optimizers can currently support mixed variable linear constraints.

References Dakota::abort_handler(), Constraints::continuousLowerBnds, Dakota::copy_data(), Constraints::discreteIntLowerBnds, Constraints::discreteRealLowerBnds, ProblemDescDB::get_rv(), Constraints::linearEqConCoeffs, Constraints::linearEqConTargets, Constraints::linearIneqConCoeffs, Constraints::linearIneqConLowerBnds, Constraints::linearIneqConUpperBnds, Constraints::numLinearEqCons, and Constraints::numLinearIneqCons.

Referenced by Constraints::Constraints().

Constraints * get_constraints (const ProblemDescDB & *problem_db*, const SharedVariablesData & *svd*) `[private]`

Used only by the constructor to initialize constraintsRep to the appropriate derived type.

Initializes constraintsRep to the appropriate derived type, as given by the variables view.

References SharedVariablesData::view().

Referenced by Constraints::Constraints(), and Constraints::copy().

Constraints * get_constraints (const SharedVariablesData & *svd*) const `[private]`

Used by copy() to initialize constraintsRep to the appropriate derived type.

Initializes constraintsRep to the appropriate derived type, as given by the variables view. The default derived class constructors are invoked.

References SharedVariablesData::view().

The documentation for this class was generated from the following files:

- DakotaConstraints.hpp
- DakotaConstraints.cpp

13.19 DataEnvironment Class Reference

Handle class for environment specification data.

Public Member Functions

- DataEnvironment ()

  constructor
- DataEnvironment (const DataEnvironment &)

  copy constructor
- ~DataEnvironment ()

  destructor
- DataEnvironment & operator= (const DataEnvironment &)

  assignment operator
- void write (std::ostream &s) const

  write a DataEnvironment object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataEnvironment object from a packed MPI buffer

- void write (MPIPackBuffer &s) const

  write a DataEnvironment object to a packed MPI buffer
- DataEnvironmentRep * data_rep ()

  return dataEnvRep

Private Attributes

- DataEnvironmentRep * dataEnvRep

  pointer to the body (handle-body idiom)

Friends

- class **ProblemDescDB**
- class **NIDRProblemDescDB**

13.19.1 Detailed Description

Handle class for environment specification data.

The DataEnvironment class is used to provide a memory management handle for the data in DataEnvironment-Rep. It is populated by IDRProblemDescDB::environment_kwhandler() and is queried by the ProblemDescDB-::get_<datatype>() functions. A single DataEnvironment object is maintained in ProblemDescDB::environment-Spec.

The documentation for this class was generated from the following files:

- DataEnvironment.hpp
- DataEnvironment.cpp

13.20 DataEnvironmentRep Class Reference

Body class for environment specification data.

Public Attributes

- bool checkFlag

  flag for whether to run in check only mode (default false)
- String outputFile

  file name for output redirection (overrides command-line)
- String errorFile

  file name for error redirection (overrides command-line)
- String readRestart

  file name for restart read (overrides command-line)
- int stopRestart

  record at which to stop reading restart
- String writeRestart

  file name for restart write (overrides command-line)
- bool preRunFlag

flags invocation with command line option -pre_run

- bool runFlag

  flags invocation with command line option -run
- bool postRunFlag

  flags invocation with command line option -post_run
- String preRunInput

  filename for pre_run input
- String preRunOutput

  filename for pre_run output
- String runInput

  filename for run input
- String runOutput

  filename for run output
- String postRunInput

  filename for post_run input
- String postRunOutput

  filename for post_run output
- bool graphicsFlag

  flags use of graphics by the environment (from the `graphics` *specification in EnvIndControl)*
- bool tabularDataFlag

  flags tabular data collection by the environment (from the tabular_graphics_data specification in EnvIndControl)
- String tabularDataFile

  the filename used for tabular data collection by the environment (from the tabular_graphics_file specification in EnvIndControl)
- int outputPrecision

  output precision for tabular and screen output
- bool resultsOutputFlag

  flags use of results output to default file
- String resultsOutputFile

  named file for results output
- String topMethodPointer

  method identifier for the environment (from the `top_method_pointer` *specification*

Private Member Functions

- DataEnvironmentRep ()

  constructor
- ~DataEnvironmentRep ()

  destructor
- void write (std::ostream &s) const

  write a DataEnvironmentRep object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataEnvironmentRep object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataEnvironmentRep object to a packed MPI buffer

Private Attributes

- int referenceCount

  number of handle objects sharing this dataEnvironmentRep

Friends

- class DataEnvironment

  the handle class can access attributes of the body class directly

13.20.1 Detailed Description

Body class for environment specification data.

The DataEnvironmentRep class is used to contain the data from the environment keyword specification. Default values are managed in the DataEnvironmentRep constructor. Data is public to avoid maintaining set/get functions, but is still encapsulated within ProblemDescDB since ProblemDescDB::environmentSpec is private.

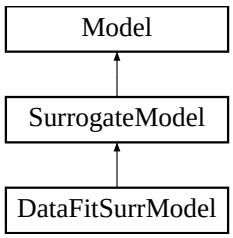
The documentation for this class was generated from the following files:

- DataEnvironment.hpp
- DataEnvironment.cpp

13.21 DataFitSurrModel Class Reference

Derived model class within the surrogate model branch for managing data fit surrogates (global and local)

Inheritance diagram for DataFitSurrModel:



Public Member Functions

- DataFitSurrModel (ProblemDescDB &problem_db)

  constructor
- DataFitSurrModel (Iterator &dace_iterator, Model &actual_model, const String &approx_type, const UShortArray &approx_order, short corr_type, short corr_order, short data_order, short output_level, const String &point_reuse, const String &export_points_file=String(), bool export_annotated=true, const String &import_points_file=String(), bool import_annotated=true, bool import_active_only=false)

  alternate constructor for instantiations on the fly
- ~DataFitSurrModel ()

  destructor
- void total_points (int points)

  set pointsTotal and pointsManagement mode

Protected Member Functions

- void derived_compute_response (const ActiveSet &set)

  portion of compute_response() specific to DataFitSurrModel
- void derived_asynch_compute_response (const ActiveSet &set)

  portion of asynch_compute_response() specific to DataFitSurrModel
- const IntResponseMap & derived_synchronize ()

  portion of synchronize() specific to DataFitSurrModel
- const IntResponseMap & derived_synchronize_nowait ()

  portion of synchronize_nowait() specific to DataFitSurrModel
- Iterator & subordinate_iterator ()

  return daceIterator
- Model & surrogate_model ()

  return this model instance
- Model & truth_model ()

  return actualModel
- void derived_subordinate_models (ModelList &ml, bool recurse_flag)

  return actualModel (and optionally its sub-models)
- void update_from_subordinate_model (bool recurse_flag=true)

  pass request to actualModel if recursing and then update from it
- Interface & derived_interface ()

  return approxInterface
- void primary_response_fn_weights (const RealVector &wts, bool recurse_flag=true)

  set the relative weightings for multiple objective functions or least squares terms and optionally recurses into actualModel
- void surrogate_response_mode (short mode)

  set responseMode and pass any bypass request on to actualModel for any lower-level surrogates.
- void surrogate_function_indices (const IntSet &surr_fn_indices)

  (re)set the surrogate index set in SurrogateModel::surrogateFnIndices and ApproximationInterface::approxFnIndices
- void build_approximation ()

  Builds the local/multipoint/global approximation using daceIterator/actualModel to generate new data points.
- bool build_approximation (const Variables &vars, const IntResponsePair &response_pr)

  Builds the local/multipoint/global approximation using daceIterator/actualModel to generate new data points that augment the vars/response anchor point.
- void update_approximation (bool rebuild_flag)

  replaces the approximation data with daceIterator results and rebuilds the approximation if requested
- void update_approximation (const Variables &vars, const IntResponsePair &response_pr, bool rebuild_flag)

  replaces the anchor point, and rebuilds the approximation if requested
- void update_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map, bool rebuild_flag)

  replaces the current points array and rebuilds the approximation if requested
- void append_approximation (bool rebuild_flag)

  appends daceIterator results to a global approximation and rebuilds it if requested

- void append_approximation (const Variables &vars, const IntResponsePair &response_pr, bool rebuild_flag)

  appends a point to a global approximation and rebuilds it if requested
- void append_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map, bool rebuild_flag)

  appends an array of points to a global approximation and rebuilds it if requested
- void pop_approximation (bool save_surr_data, bool rebuild_flag=false)

  remove approximation data added on previous append_approximation() call or a specified number of points
- void restore_approximation ()

  restore a previous approximation data state
- bool restore_available ()

  query for whether a trial increment is restorable
- void finalize_approximation ()

  finalize data fit by applying all previous trial increments
- void store_approximation ()

  store the current data fit approximation for later combination
- void combine_approximation (short corr_type)

  combine the current data fit approximation with one previously stored
- SharedApproxData & shared_approximation ()

  retrieve the SharedApproxData from approxInterface
- std::vector< Approximation > & approximations ()

  retrieve the set of Approximations from approxInterface
- const RealVectorArray & approximation_coefficients (bool normalized=false)

  return the approximation coefficients from each Approximation (request forwarded to approxInterface)
- void approximation_coefficients (const RealVectorArray &approx_coeffs, bool normalized=false)

  set the approximation coefficients within each Approximation (request forwarded to approxInterface)
- const RealVector & approximation_variances (const Variables &vars)

  return the approximation variance from each Approximation (request forwarded to approxInterface)
- const Pecos::SurrogateData & approximation_data (size_t index)

  return the approximation data from a particular Approximation (request forwarded to approxInterface)
- void component_parallel_mode (short mode)

  update component parallel mode for supporting parallelism in actualModel
- void derived_init_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set up actualModel for parallel operations
- void derived_init_serial ()

  set up actualModel for serial operations.
- void derived_set_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set active parallel configuration within actualModel
- void derived_free_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  deallocate communicator partitions for the DataFitSurrModel (request forwarded to actualModel)
- void serve_run (ParLevLIter pl_iter, int max_eval_concurrency)

  Service actualModel job requests received from the master. Completes when a termination message is received from stop_servers().
- void stop_servers ()

Executed by the master to terminate actualModel server operations when DataFitSurrModel iteration is complete.

- void inactive_view (short view, bool recurse_flag=true)

  update the Model's inactive view based on higher level (nested) context and optionally recurse into actualModel
- const String & interface_id () const

  return the approxInterface identifier
- int evaluation_id () const

  return the current evaluation id for the DataFitSurrModel
- void set_evaluation_reference ()

  set the evaluation counter reference points for the DataFitSurrModel (request forwarded to approxInterface and actualModel)
- void fine_grained_evaluation_counters ()

  request fine-grained evaluation reporting within approxInterface and actualModel
- void print_evaluation_summary (std::ostream &s, bool minimal_header=false, bool relative_count=true) const

  print the evaluation summary for the DataFitSurrModel (request forwarded to approxInterface and actualModel)
- void eval_tag_prefix (const String &eval_id_str)

  set the hierarchical eval ID tag prefix

Private Member Functions

- void import_points (bool annotated, bool active_only)

  optionally read surrogate data points from provided file
- void initialize_export ()

  initialize file stream for exporting surrogate evaluations
- void manage_data_recastings ()

  initialize manageRecasting and recastFlags for data import/export
- void export_point (int eval_id, const Variables &vars, const Response &resp)

  initialize file stream for exporting surrogate evaluations
- void derived_synchronize_approx (const IntResponseMap &approx_resp_map, IntResponseMap &approx_resp_map_rekey)

  Common code for processing of approximate response maps shared by derived_synchronize() and derived_synchronize_nowait()
- void update_global ()

  Updates fit arrays for global approximations.
- void update_local_multipoint ()

  Updates fit arrays for local or multipoint approximations.
- void build_global ()

  Builds a global approximation using daceIterator.
- void build_local_multipoint ()

  Builds a local or multipoint approximation using actualModel.
- void update_actual_model ()

  update actualModel with data from current variables/labels/bounds/targets
- void update_from_actual_model ()

  update current variables/labels/bounds/targets with data from actualModel
- bool inside (const RealVector &c_vars, const IntVector &di_vars, const RealVector &dr_vars)

  test if c_vars and d_vars are within [c_l_bnds,c_u_bnds] and [d_l_bnds,d_u_bnds]

Private Attributes

- int surrModelEvalCntr

  number of calls to derived_compute_response()/ derived_asynch_compute_response()
- int pointsTotal

  total points the user specified to construct the surrogate
- short pointsManagement

  configuration for points management in build_global()
- String pointReuse

  type of point reuse for approximation builds: `all`, `region` (default if points file), or `none` (default if no points file)
- bool manageRecasting

  flag indicating need to manage data recastings when importing build data or exporting approximate evaluations
- BoolDeque recastFlags

  a key indicating which models within a model recursion involve recasting
- String importPointsFile

  file name from `import_points_file` specification
- String exportPointsFile

  file name from `export_points_file` specification
- bool exportAnnotated

  annotation setting for file export of variables and approximate responses
- std::ofstream exportFileStream

  file name for `export_points_file` specification
- VariablesList reuseFileVars

  array of variables sets read from the `import_points_file`
- ResponseList reuseFileResponses

  array of response sets read from the `import_points_file`
- Interface approxInterface

  manages the building and subsequent evaluation of the approximations (required for both global and local)
- Model actualModel

  the truth model which provides evaluations for building the surrogate (optional for global, required for local)
- Iterator daceIterator

  selects parameter sets on which to evaluate actualModel in order to generate the necessary data for building global approximations (optional for global since restart data may also be used)
- String evalTagPrefix

  cached evalTag Prefix from parents to use at compute_response time

Additional Inherited Members

13.21.1 Detailed Description

Derived model class within the surrogate model branch for managing data fit surrogates (global and local)

The DataFitSurrModel class manages global or local approximations (surrogates that involve data fits) that are used in place of an expensive model. The class contains an approxInterface (required for both global and local) which manages the approximate function evaluations, an actualModel (optional for global, required for local) which provides truth evaluations for building the surrogate, and a daceIterator (optional for global, not used for local) which selects parameter sets on which to evaluate actualModel in order to generate the necessary data for building global approximations.

13.21.2 Constructor & Destructor Documentation

~DataFitSurrModel ( ) [inline]

destructor

Virtual destructor handles referenceCount at Strategy level.

References DataFitSurrModel::exportFileStream, and DataFitSurrModel::exportPointsFile.

13.21.3 Member Function Documentation

void derived_compute_response (const ActiveSet & *set*) [protected], [virtual]

portion of compute_response() specific to DataFitSurrModel

Compute the response synchronously using actualModel, approxInterface, or both (mixed case). For the approxInterface portion, build the approximation if needed, evaluate the approximate response, and apply correction (if active) to the results.

Reimplemented from Model.

References DiscrepancyCorrection::active(), Response::active_set(), DataFitSurrModel::actualModel, DiscrepancyCorrection::apply(), SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, SurrogateModel::asv_mapping(), DataFitSurrModel::build_approximation(), DataFitSurrModel::component_parallel_mode(), DiscrepancyCorrection::compute(), Model::compute_response(), Response::copy(), Model::current_response(), Model::currentResponse, Model::currentVariables, SurrogateModel::deltaCorr, Model::eval_tag_prefix(), DataFitSurrModel::evalTagPrefix, DataFitSurrModel::export_point(), SurrogateModel::force_rebuild(), Model::hierarchicalTagging, Interface::map(), Model::outputLevel, ActiveSet::request_vector(), SurrogateModel::response_mapping(), SurrogateModel::responseMode, DataFitSurrModel::surrModelEvalCntr, Response::update(), and DataFitSurrModel::update_actual_model().

void derived_asynch_compute_response (const ActiveSet & *set*) [protected], [virtual]

portion of asynch_compute_response() specific to DataFitSurrModel

Compute the response asynchronously using actualModel, approxInterface, or both (mixed case). For the approxInterface portion, build the approximation if needed and evaluate the approximate response in a quasi-asynchronous approach (ApproximationInterface::map() performs the map synchronously and bookkeeps the results for return in derived_synchronize() below).

Reimplemented from Model.

References DataFitSurrModel::actualModel, SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, SurrogateModel::asv_mapping(), Model::asynch_compute_response(), DataFitSurrModel::build_approximation(), Variables::copy(), Model::currentResponse, Model::currentVariables, Model::eval_tag_prefix(), DataFitSurrModel::evalTagPrefix, Interface::evaluation_id(), Model::evaluation_id(), DataFitSurrModel::exportPointsFile, SurrogateModel::force_rebuild(), Model::hierarchicalTagging, Interface::map(), SurrogateModel::rawVarsMap, ActiveSet::request_vector(), SurrogateModel::responseMode, SurrogateModel::surrIdMap, DataFitSurrModel::surrModelEvalCntr, SurrogateModel::truthIdMap, and DataFitSurrModel::update_actual_model().

const IntResponseMap & derived_synchronize ( ) [protected], [virtual]

portion of synchronize() specific to DataFitSurrModel

Blocking retrieval of asynchronous evaluations from actualModel, approxInterface, or both (mixed case). For the approxInterface portion, apply correction (if active) to each response in the array. derived_synchronize() is designed for the general case where derived_asynch_compute_response() may be inconsistent in its use of actual evaluations, approximate evaluations, or both.

Reimplemented from Model.

References DataFitSurrModel::actualModel, DataFitSurrModel::approxInterface, DataFitSurrModel::component_parallel_mode(), DiscrepancyCorrection::compute(), SurrogateModel::deltaCorr, DataFitSurrModel::derived_synchronize_approx(), Model::outputLevel, SurrogateModel::response_mapping(), SurrogateModel::responseMode, SurrogateModel::surrIdMap, SurrogateModel::surrResponseMap, Interface::synch(), Model::synchronize(), and SurrogateModel::truthIdMap.

const IntResponseMap & derived_synchronize_nowait ( ) [protected], [virtual]

portion of synchronize_nowait() specific to DataFitSurrModel

Nonblocking retrieval of asynchronous evaluations from actualModel, approxInterface, or both (mixed case). For the approxInterface portion, apply correction (if active) to each response in the map. derived_synchronize_nowait() is designed for the general case where derived_asynch_compute_response() may be inconsistent in its use of actual evals, approx evals, or both.

Reimplemented from Model.

References Dakota::abort_handler(), DataFitSurrModel::actualModel, DataFitSurrModel::approxInterface, SurrogateModel::cachedApproxRespMap, DataFitSurrModel::component_parallel_mode(), DiscrepancyCorrection::compute(), SurrogateModel::deltaCorr, DataFitSurrModel::derived_synchronize_approx(), Model::outputLevel, SurrogateModel::response_mapping(), SurrogateModel::responseMode, SurrogateModel::surrIdMap, SurrogateModel::surrResponseMap, Interface::synch_nowait(), Model::synchronize_nowait(), and SurrogateModel::truthIdMap.

void build_approximation ( ) [protected], [virtual]

Builds the local/multipoint/global approximation using daceIterator/actualModel to generate new data points.

This function constructs a new approximation, discarding any previous data. It constructs any required data for SurrogateData::{vars,resp}Data and does not define an anchor point for SurrogateData::anchor{Vars,Resp}.

Reimplemented from Model.

References DataFitSurrModel::actualModel, SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, Interface::build_approximation(), DataFitSurrModel::build_global(), DataFitSurrModel::build_local_multipoint(), Interface::clear_current(), Constraints::continuous_lower_bounds(), Model::continuous_lower_bounds(), Constraints::continuous_upper_bounds(), Model::continuous_upper_bounds(), Constraints::discrete_int_lower_bounds(), Model::discrete_int_lower_bounds(), Constraints::discrete_int_upper_bounds(), Model::discrete_int_upper_bounds(), Constraints::discrete_real_lower_bounds(), Model::discrete_real_lower_bounds(), Constraints::discrete_real_upper_bounds(), Model::discrete_real_upper_bounds(), Model::is_null(), Dakota::strbegins(), Model::surrogateType, DataFitSurrModel::update_actual_model(), DataFitSurrModel::update_global(), DataFitSurrModel::update_local_multipoint(), and Model::userDefinedConstraints.

Referenced by DataFitSurrModel::derived_asynch_compute_response(), and DataFitSurrModel::derived_compute_response().

bool build_approximation (const Variables & *vars*, const IntResponsePair & *response_pr*) [protected], [virtual]

Builds the local/multipoint/global approximation using daceIterator/actualModel to generate new data points that augment the vars/response anchor point.

This function constructs a new approximation, discarding any previous data. It uses the passed data to populate SurrogateData::anchor{Vars,Resp} and constructs any required data points for SurrogateData::{vars,resp}Data.

Reimplemented from Model.

References DataFitSurrModel::actualModel, SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, Interface::build_approximation(), DataFitSurrModel::build_global(), Interface::clear_current(), Constraints::continuous_lower_bounds(), Model::continuous_lower_bounds(), Constraints::continuous_upper_bounds(), Model::continuous_upper_bounds(), Constraints::discrete_int_lower_bounds(), Model::discrete_int_lower_bounds(), Constraints::discrete-

_int_upper_bounds(), Model::discrete_int_upper_bounds(), Constraints::discrete_real_lower_bounds(), Model::discrete_real_lower_bounds(), Constraints::discrete_real_upper_bounds(), Model::discrete_real_upper_bounds(), Model::is_null(), Dakota::strbegins(), Model::surrogateType, DataFitSurrModel::update_actual_model(), Interface::update_approximation(), DataFitSurrModel::update_global(), DataFitSurrModel::update_local_multipoint(), and Model::userDefinedConstraints.

void update_approximation (bool *rebuild_flag*) [protected], [virtual]

replaces the approximation data with daceIterator results and rebuilds the approximation if requested

This function populates/replaces SurrogateData::anchor{Vars,Resp} and rebuilds the approximation, if requested. It does not clear other data (i.e., SurrogateData::{vars,resp}Data) and does not update the actualModel with revised bounds, labels, etc. Thus, it updates data from a previous call to build_approximation(), and is not intended to be used in isolation.

Reimplemented from Model.

References Iterator::all_responses(), Iterator::all_samples(), Iterator::all_variables(), SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, Iterator::compact_mode(), DataFitSurrModel::daceIterator, Model::numFns, Interface::rebuild_approximation(), Model::surrogateType, and Interface::update_approximation().

void update_approximation (const Variables & *vars,* const IntResponsePair & *response_pr,* bool *rebuild_flag*) [protected], [virtual]

replaces the anchor point, and rebuilds the approximation if requested

This function populates/replaces SurrogateData::anchor{Vars,Resp} and rebuilds the approximation, if requested. It does not clear other data (i.e., SurrogateData::{vars,resp}Data) and does not update the actualModel with revised bounds, labels, etc. Thus, it updates data from a previous call to build_approximation(), and is not intended to be used in isolation.

Reimplemented from Model.

References SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, Model::numFns, Interface::rebuild_approximation(), Model::surrogateType, and Interface::update_approximation().

void update_approximation (const VariablesArray & *vars_array,* const IntResponseMap & *resp_map,* bool *rebuild_flag*) [protected], [virtual]

replaces the current points array and rebuilds the approximation if requested

This function populates/replaces SurrogateData::{vars,resp}Data and rebuilds the approximation, if requested. It does not clear other data (i.e., SurrogateData::anchor{Vars,Resp}) and does not update the actualModel with revised bounds, labels, etc. Thus, it updates data from a previous call to build_approximation(), and is not intended to be used in isolation.

Reimplemented from Model.

References SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, Model::numFns, Interface::rebuild_approximation(), Model::surrogateType, and Interface::update_approximation().

void append_approximation (bool *rebuild_flag*) [protected], [virtual]

appends daceIterator results to a global approximation and rebuilds it if requested

This function appends one point to SurrogateData::{vars,resp}Data and rebuilds the approximation, if requested. It does not modify other data (i.e., SurrogateData::anchor{Vars,Resp}) and does not update the actualModel with revised bounds, labels, etc. Thus, it appends to data from a previous call to build_approximation(), and is not intended to be used in isolation.

Reimplemented from Model.

References Iterator::all_responses(), Iterator::all_samples(), Iterator::all_variables(), Interface::append_approximation(), SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, Iterator::compact_mode(), DataFitSurrModel::daceIterator, Model::numFns, Interface::rebuild_approximation(), and Model::surrogateType.

void append_approximation (const Variables & *vars,* const IntResponsePair & *response_pr,* bool *rebuild_flag*) `[protected], [virtual]`

appends a point to a global approximation and rebuilds it if requested

This function appends one point to SurrogateData::{vars,resp}Data and rebuilds the approximation, if requested. It does not modify other data (i.e., SurrogateData::anchor{Vars,Resp}) and does not update the actualModel with revised bounds, labels, etc. Thus, it appends to data from a previous call to build_approximation(), and is not intended to be used in isolation.

Reimplemented from Model.

References Interface::append_approximation(), SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, Model::numFns, Interface::rebuild_approximation(), and Model::surrogateType.

void append_approximation (const VariablesArray & *vars_array,* const IntResponseMap & *resp_map,* bool *rebuild_flag*) `[protected], [virtual]`

appends an array of points to a global approximation and rebuilds it if requested

This function appends multiple points to SurrogateData::{vars,resp}Data and rebuilds the approximation, if requested. It does not modify other data (i.e., SurrogateData::anchor{Vars,Resp}) and does not update the actualModel with revised bounds, labels, etc. Thus, it appends to data from a previous call to build_approximation(), and is not intended to be used in isolation.

Reimplemented from Model.

References Interface::append_approximation(), SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, Model::numFns, Interface::rebuild_approximation(), and Model::surrogateType.

void derived_init_communicators (ParLevLIter *pl_iter,* int *max_eval_concurrency,* bool *recurse_flag* = `true`) `[protected], [virtual]`

set up actualModel for parallel operations

asynchronous flags need to be initialized for the sub-models. In addition, max_eval_concurrency is the outer level iterator concurrency, not the DACE concurrency that actualModel will see, and recomputing the message_lengths on the sub-model is probably not a bad idea either. Therefore, recompute everything on actualModel using init_communicators.

Reimplemented from Model.

References DataFitSurrModel::actualModel, DataFitSurrModel::approxInterface, DataFitSurrModel::daceIterator, Model::derivative_concurrency(), ProblemDescDB::get_db_method_node(), ProblemDescDB::get_db_model_node(), Iterator::init_communicators(), Model::init_communicators(), Iterator::is_null(), Model::is_null(), Iterator::iterated_model(), Iterator::maximum_evaluation_concurrency(), Iterator::method_id(), Interface::minimum_points(), Model::model_id(), Model::probDescDB, ProblemDescDB::set_db_list_nodes(), ProblemDescDB::set_db_method_node(), and ProblemDescDB::set_db_model_nodes().

int evaluation_id ( ) const `[inline], [protected], [virtual]`

return the current evaluation id for the DataFitSurrModel

return the DataFitSurrModel evaluation count. Due to possibly intermittent use of surrogate bypass, this is not the same as either the approxInterface or actualModel model evaluation counts. It also does not distinguish duplicate evals.

Reimplemented from Model.

References DataFitSurrModel::surrModelEvalCntr.

void import_points (bool *annotated,* bool *active_only*) [private]

optionally read surrogate data points from provided file

Constructor helper to read the points file once, if provided, and then reuse its data as appropriate within build_global(). Surrogate data imports default to active/inactive variables, but user can override to active only

References Variables::cv(), Variables::div(), Variables::drv(), Variables::dsv(), Response::is_null(), Variables::is_null(), and Variables::tv().

Referenced by DataFitSurrModel::DataFitSurrModel().

void initialize_export ( ) [private]

initialize file stream for exporting surrogate evaluations

Constructor helper to export approximation-based evaluations to a file.

Referenced by DataFitSurrModel::DataFitSurrModel().

void manage_data_recastings ( ) [private]

initialize manageRecasting and recastFlags for data import/export

Constructor helper to manage model recastings for data import/export.

Referenced by DataFitSurrModel::DataFitSurrModel().

void export_point (int *eval_id,* const Variables & *vars,* const Response & *resp*) [private]

initialize file stream for exporting surrogate evaluations

Constructor helper to export approximation-based evaluations to a file. Exports all variables, so it's clear at what values of inactive it was built at

References RecastModel::inverse_transform_response(), Model::model_rep(), and RecastModel::transform_variables().

Referenced by DataFitSurrModel::derived_compute_response().

void build_global ( ) [private]

Builds a global approximation using daceIterator.

Determine points to use in building the approximation and then evaluate them on actualModel using daceIterator. Any changes to the bounds should be performed by setting them at a higher level (e.g., SurrBasedOptStrategy).

References Dakota::abort_handler(), Iterator::active_set(), DataFitSurrModel::actualModel, Iterator::all_responses(), Iterator::all_samples(), Iterator::all_variables(), Interface::append_approximation(), Interface::approximation_data(), DataFitSurrModel::approxInterface, SurrogateModel::asv_mapping(), Iterator::compact_mode(), DataFitSurrModel::component_parallel_mode(), Variables::continuous_variables(), Model::currentVariables, Variables::cv(), Model::cv(), DataFitSurrModel::daceIterator, Dakota::data_pairs, Variables::discrete_int_variables(), Variables::discrete_real_variables(), Variables::div(), Model::div(), Variables::drv(), Model::drv(), Iterator::eval_tag_prefix(), DataFitSurrModel::evalTagPrefix, Model::hierarchicalTagging, DataFitSurrModel::importPointsFile, DataFitSurrModel::inside(), Model::interface_id(), RecastModel::inverse_transform_variables(), Iterator::is_null(), Model::is_null(),

DataFitSurrModel::manageRecasting, Interface::minimum_points(), SurrogateModel::miPLIndex, Model::model_rep(), Model::modelList, Model::modelPCIter, Iterator::num_samples(), Model::outputLevel, DataFitSurrModel::pointReuse, DataFitSurrModel::pointsManagement, DataFitSurrModel::pointsTotal, DataFitSurrModel::recastFlags, Interface::recommended_points(), ActiveSet::request_vector(), DataFitSurrModel::reuseFileResponses, DataFitSurrModel::reuseFileVars, Iterator::run(), Iterator::sampling_reset(), DataFitSurrModel::surrModelEvalCntr, and SurrogateModel::surrogateFnIndices.

Referenced by DataFitSurrModel::build_approximation().

void build_local_multipoint ( ) [private]

Builds a local or multipoint approximation using actualModel.

Evaluate the value, gradient, and possibly Hessian needed for a local or multipoint approximation using actualModel.

References Response::active_set(), DataFitSurrModel::actualModel, DataFitSurrModel::approxInterface, SurrogateModel::asv_mapping(), DataFitSurrModel::component_parallel_mode(), Model::compute_response(), Model::continuous_variable_ids(), Model::current_response(), Model::current_variables(), ActiveSet::derivative_vector(), Model::evaluation_id(), Model::hessian_type(), Model::numFns, ActiveSet::request_vector(), Dakota::strbegins(), Model::surrogateType, and Interface::update_approximation().

Referenced by DataFitSurrModel::build_approximation().

void update_actual_model ( ) [private]

update actualModel with data from current variables/labels/bounds/targets

Update variables and constraints data within actualModel using values and labels from currentVariables and bound/linear/nonlinear constraints from userDefinedConstraints.

References Dakota::abort_handler().

Referenced by DataFitSurrModel::build_approximation(), DataFitSurrModel::derived_asynch_compute_response(), and DataFitSurrModel::derived_compute_response().

void update_from_actual_model ( ) [private]

update current variables/labels/bounds/targets with data from actualModel

Update values and labels in currentVariables and bound/linear/nonlinear constraints in userDefinedConstraints from variables and constraints data within actualModel.

References Dakota::abort_handler().

Referenced by DataFitSurrModel::DataFitSurrModel(), and DataFitSurrModel::update_from_subordinate_model().

13.21.4 Member Data Documentation

Model actualModel [private]

the truth model which provides evaluations for building the surrogate (optional for global, required for local)

actualModel is unrestricted in type; arbitrary nestings are possible.

Referenced by DataFitSurrModel::build_approximation(), DataFitSurrModel::build_global(), DataFitSurrModel::build_local_multipoint(), DataFitSurrModel::DataFitSurrModel(), DataFitSurrModel::derived_asynch_compute_response(), DataFitSurrModel::derived_compute_response(), DataFitSurrModel::derived_free_communicators(), DataFitSurrModel::derived_init_communicators(), DataFitSurrModel::derived_init_serial(), DataFitSurrModel::derived_set_communicators(), DataFitSurrModel::derived_subordinate_models(), DataFitSurrModel::derived_synchronize(),

DataFitSurrModel::derived_synchronize_nowait(), DataFitSurrModel::fine_grained_evaluation_counters(), DataFitSurrModel::inactive_view(), DataFitSurrModel::inside(), DataFitSurrModel::primary_response_fn_weights(), DataFitSurrModel::print_evaluation_summary(), DataFitSurrModel::serve_run(), DataFitSurrModel::stop_servers(), DataFitSurrModel::surrogate_response_mode(), DataFitSurrModel::truth_model(), DataFitSurrModel::update_from_subordinate_model(), DataFitSurrModel::update_global(), and DataFitSurrModel::update_local_multipoint().

The documentation for this class was generated from the following files:

- DataFitSurrModel.hpp
- DataFitSurrModel.cpp

13.22 DataInterface Class Reference

Handle class for interface specification data.

Public Member Functions

- DataInterface ()

  constructor
- DataInterface (const DataInterface &)

  copy constructor
- ~DataInterface ()

  destructor
- DataInterface & operator= (const DataInterface &)

  assignment operator
- void write (std::ostream &s) const

  write a DataInterface object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataInterface object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataInterface object to a packed MPI buffer
- DataInterfaceRep * data_rep ()

  return dataIfaceRep

Static Public Member Functions

- static bool id_compare (const DataInterface &di, const std::string &id)

  compares the idInterface attribute of DataInterface objects

Private Attributes

- DataInterfaceRep * dataIfaceRep

  pointer to the body (handle-body idiom)

Friends

- class **ProblemDescDB**
- class **NIDRProblemDescDB**

13.22.1 Detailed Description

Handle class for interface specification data.

The DataInterface class is used to provide a memory management handle for the data in DataInterfaceRep. It is populated by IDRProblemDescDB::interface_kwhandler() and is queried by the ProblemDescDB::get_<datatype>() functions. A list of DataInterface objects is maintained in ProblemDescDB::dataInterfaceList, one for each interface specification in an input file.

The documentation for this class was generated from the following files:

- DataInterface.hpp
- DataInterface.cpp

13.23 DataMethod Class Reference

Handle class for method specification data.

Public Member Functions

- DataMethod ()

  constructor
- DataMethod (const DataMethod &)

  copy constructor
- ~DataMethod ()

  destructor
- DataMethod & operator= (const DataMethod &)

  assignment operator
- void write (std::ostream &s) const

  write a DataMethod object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataMethod object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataMethod object to a packed MPI buffer
- DataMethodRep * data_rep ()

  return dataMethodRep

Static Public Member Functions

- static bool id_compare (const DataMethod &dm, const std::string &id)

  compares the idMethod attribute of DataMethod objects

Private Attributes

- DataMethodRep * dataMethodRep

  pointer to the body (handle-body idiom)

Friends

- class **ProblemDescDB**
- class **NIDRProblemDescDB**

13.23.1 Detailed Description

Handle class for method specification data.

The DataMethod class is used to provide a memory management handle for the data in DataMethodRep. It is populated by IDRProblemDescDB::method_kwhandler() and is queried by the ProblemDescDB::get_<datatype>() functions. A list of DataMethod objects is maintained in ProblemDescDB::dataMethodList, one for each method specification in an input file.

The documentation for this class was generated from the following files:

- DataMethod.hpp
- DataMethod.cpp

13.24 DataMethodRep Class Reference

Body class for method specification data.

Public Attributes

- String idMethod

  string identifier for the method specification data set (from the `id_method` specification in MethodIndControl)
- String modelPointer

  string pointer to the model specification to be used by this method (from the `model_pointer` specification in MethodIndControl)
- short methodOutput

  method verbosity control: {SILENT,QUIET,NORMAL,VERBOSE,DEBUG}_OUTPUT (from the `output` specification in MethodIndControl)
- int maxIterations

  maximum number of iterations allowed for the method (from the `max_iterations` specification in MethodIndControl)
- int maxFunctionEvaluations

  maximum number of function evaluations allowed for the method (from the `max_function_evaluations` specification in MethodIndControl)
- bool speculativeFlag

  flag for use of speculative gradient approaches for maintaining parallel load balance during the line search portion of optimization algorithms (from the `speculative` specification in MethodIndControl)
- bool methodUseDerivsFlag

  flag for usage of derivative data to enhance the computation of surrogate models (PCE/SC expansions, GP models for EGO/EGRA/EGIE) based on the `use_derivatives` specification
- Real convergenceTolerance

  iteration convergence tolerance for the method (from the `convergence_tolerance` specification in MethodIndControl)
- Real constraintTolerance

tolerance for controlling the amount of infeasibility that is allowed before an active constraint is considered to be violated (from the `constraint_tolerance` *specification in MethodIndControl)*

- bool methodScaling

  flag indicating scaling status (from the `scaling` *specification in MethodIndControl)*

- size_t numFinalSolutions

  number of final solutions returned from the iterator

- RealVector linearIneqConstraintCoeffs

  coefficient matrix for the linear inequality constraints (from the `linear_inequality_constraint_matrix` *specification in MethodIndControl)*

- RealVector linearIneqLowerBnds

  lower bounds for the linear inequality constraints (from the `linear_inequality_lower_bounds` *specification in MethodIndControl)*

- RealVector linearIneqUpperBnds

  upper bounds for the linear inequality constraints (from the `linear_inequality_upper_bounds` *specification in MethodIndControl)*

- StringArray linearIneqScaleTypes

  scaling types for the linear inequality constraints (from the `linear_inequality_scale_types` *specification in MethodIndControl)*

- RealVector linearIneqScales

  scaling factors for the linear inequality constraints (from the `linear_inequality_scales` *specification in MethodIndControl)*

- RealVector linearEqConstraintCoeffs

  coefficient matrix for the linear equality constraints (from the `linear_equality_constraint_matrix` *specification in MethodIndControl)*

- RealVector linearEqTargets

  targets for the linear equality constraints (from the `linear_equality_targets` *specification in MethodIndControl)*

- StringArray linearEqScaleTypes

  scaling types for the linear equality constraints (from the `linear_equality_scale_types` *specification in MethodIndControl)*

- RealVector linearEqScales

  scaling factors for the linear equality constraints (from the `linear_equality_scales` *specification in MethodIndControl)*

- unsigned short methodName

  the method selection: one of the optimizer, least squares, nond, dace, or parameter study methods

- unsigned short subMethod

  enum value for a sub-method type

- String subMethodName

  string identifier for a sub-method name within a multi-option method specification (e.g., from meta-iterators)

- String subModelPointer

  string pointer for a sub-model specification used by a meta-iterator

- String subMethodPointer

  string pointer for a sub-method specification used by a meta-iterator

- int iteratorServers

  number of servers for concurrent iterator parallelism (from the `iterator_servers` *specification)*

- int procsPerIterator

  number of processors for each concurrent iterator partition (from the `processors_per_iterator` *specification)*
- short iteratorScheduling

  type of scheduling ({DEFAULT,MASTER,PEER}_SCHEDULING) used in concurrent iterator parallelism (from the `iterator_scheduling` *specification)*
- StringArray hybridMethodNames

  array of methods for the sequential and collaborative hybrid meta-iterators (from the `method_name_list` *specification)*
- StringArray hybridModelPointers

  array of models for the sequential and collaborative hybrid meta-iterators (from the `model_pointer_list` *specification)*
- StringArray hybridMethodPointers

  array of methods for the sequential and collaborative hybrid meta-iterators (from the `method_pointer_list` *specification)*
- String hybridGlobalMethodName

  global method name for embedded hybrids (from the `global_method_name` *specification)*
- String hybridGlobalModelPointer

  global model pointer for embedded hybrids (from the `global_model_pointer` *specification)*
- String hybridGlobalMethodPointer

  global method pointer for embedded hybrids (from the `global_method_pointer` *specification)*
- String hybridLocalMethodName

  local method name for embedded hybrids (from the `local_method_name` *specification)*
- String hybridLocalModelPointer

  local model pointer for embedded hybrids (from the `local_model_pointer` *specification)*
- String hybridLocalMethodPointer

  local method pointer for embedded hybrids (from the `local_method_pointer` *specification)*
- Real hybridLSProb

  local search probability for embedded hybrids (from the `local_search_probability` *specification)*
- int concurrentRandomJobs

  number of random jobs to perform in the pareto_set and multi_start meta-iterators (from the `random_starts` *and* `random_weight_sets` *specifications)*
- RealVector concurrentParameterSets

  user-specified (i.e., nonrandom) parameter sets to evaluate in the pareto_set and multi_start meta-iterators (from the `starting_points` *and* `weight_sets` *specifications)*
- unsigned short softConvLimit

  number of consecutive iterations with change less than convergenceTolerance required to trigger convergence
- bool surrBasedLocalLayerBypass

  flag to indicate user-specification of a bypass of any/all layerings in evaluating truth response values in SBL.
- Real surrBasedLocalTRInitSize

  initial trust region size in the surrogate-based local method (from the `initial_size` *specification in MethodSBL) note: this is a relative value, e.g., 0.1 = 10% of global bounds distance (upper bound - lower bound) for each variable*
- Real surrBasedLocalTRMinSize

minimum trust region size in the surrogate-based local method (from the `minimum_size` *specification in MethodSBL), if the trust region size falls below this threshold the SBL iterations are terminated (note: if kriging is used with SBL, the min trust region size is set to 1.0e-3 in attempt to avoid ill-conditioned matrixes that arise in kriging over small trust regions)*

- Real surrBasedLocalTRContractTrigger

  trust region minimum improvement level (ratio of actual to predicted decrease in objective fcn) in the surrogate-based local method (from the `contract_threshold` *specification in MethodSBL), the trust region shrinks or is rejected if the ratio is below this value ("eta_1" in the Conn-Gould-Toint trust region book)*

- Real surrBasedLocalTRExpandTrigger

  trust region sufficient improvement level (ratio of actual to predicted decrease in objective fn) in the surrogate-based local method (from the `expand_threshold` *specification in MethodSBL), the trust region expands if the ratio is above this value ("eta_2" in the Conn-Gould-Toint trust region book)*

- Real surrBasedLocalTRContract

  trust region contraction factor in the surrogate-based local method (from the `contraction_factor` *specification in MethodSBL)*

- Real surrBasedLocalTRExpand

  trust region expansion factor in the surrogate-based local method (from the `expansion_factor` *specification in MethodSBL)*

- short surrBasedLocalSubProbObj

  SBL approximate subproblem objective: ORIGINAL_PRIMARY, SINGLE_OBJECTIVE, LAGRANGIAN_OBJECTIVE, or AUGMENTED_LAGRANGIAN_OBJECTIVE.

- short surrBasedLocalSubProbCon

  SBL approximate subproblem constraints: NO_CONSTRAINTS, LINEARIZED_CONSTRAINTS, or ORIGINAL_CONSTRAINTS.

- short surrBasedLocalMeritFn

  SBL merit function type: BASIC_PENALTY, ADAPTIVE_PENALTY, BASIC_LAGRANGIAN, or AUGMENTED_LAGRANGIAN.

- short surrBasedLocalAcceptLogic

  SBL iterate acceptance logic: TR_RATIO or FILTER.

- short surrBasedLocalConstrRelax

  SBL constraint relaxation method: NO_RELAX or HOMOTOPY.

- bool surrBasedGlobalReplacePts

  user-specified method for adding points to the set upon which the next surrogate is based in the `surrogate_based_global` *method.*

- String dlDetails

  string of options for a dynamically linked solver

- void * dlLib

  handle to dynamically loaded library

- int verifyLevel

  the `verify_level` *specification in MethodNPSOLDC*

- Real functionPrecision

  the `function_precision` *specification in MethodNPSOLDC and the* `EPSILON` *specification in NOMAD*

- Real lineSearchTolerance

  the `linesearch_tolerance` *specification in MethodNPSOLDC*

- Real absConvTol

  absolute function convergence tolerance

- Real xConvTol

  x-convergence tolerance
- Real singConvTol

  singular convergence tolerance
- Real singRadius

  radius for singular convergence test
- Real falseConvTol

  false-convergence tolerance
- Real initTRRadius

  initial trust radius
- int covarianceType

  kind of covariance required
- bool regressDiag

  whether to print the regression diagnostic vector
- String searchMethod

  the `search_method` *specification for Newton and nonlinear interior-point methods in MethodOPTPPDC*
- Real gradientTolerance

  the `gradient_tolerance` *specification in MethodOPTPPDC*
- Real maxStep

  the `max_step` *specification in MethodOPTPPDC*
- short meritFn

  the `merit_function` *specification for nonlinear interior-point methods in MethodOPTPPDC*
- Real stepLenToBoundary

  the `steplength_to_boundary` *specification for nonlinear interior-point methods in MethodOPTPPDC*
- Real centeringParam

  the `centering_parameter` *specification for nonlinear interior-point methods in MethodOPTPPDC*
- int searchSchemeSize

  the `search_scheme_size` *specification for PDS methods in MethodOPTPPDC*
- Real initStepLength

  the `initStepLength` *choice for nonlinearly constrained APPS in MethodAPPSDC*
- Real contractStepLength

  the `contractStepLength` *choice for nonlinearly constrained APPS in MethodAPPSDC*
- Real threshStepLength

  the `threshStepLength` *choice for nonlinearly constrained APPS in MethodAPPSDC*
- String meritFunction

  the `meritFunction` *choice for nonlinearly constrained APPS in MethodAPPSDC*
- Real constrPenalty

  the `constrPenalty` *choice for nonlinearly constrained APPS in MethodAPPSDC*
- Real smoothFactor

  the initial `smoothFactor` *value for nonlinearly constrained APPS in MethodAPPSDC*
- Real constraintPenalty

  the initial `constraint_penalty` *for COLINY methods in MethodAPPS, MethodSCOLIBDIR, MethodSCOLIBPS, MethodSCOLIBSW and MethodSCOLIBEA*

- bool constantPenalty

  the `constant_penalty` flag for COLINY methods in MethodSCOLIBPS and MethodSCOLIBSW
- Real globalBalanceParam

  the `global_balance_parameter` for the DIRECT method in MethodSCOLIBDIR
- Real localBalanceParam

  the `local_balance_parameter` for the DIRECT method in MethodSCOLIBDIR
- Real maxBoxSize

  the `max_boxsize_limit` for the DIRECT method in MethodSCOLIBDIR
- Real minBoxSize

  the `min_boxsize_limit` for the DIRECT method in MethodSCOLIBDIR and MethodNCSUDC
- String boxDivision

  the `division` setting (`major_dimension` or `all_dimensions`) for the DIRECT method in MethodSCOLIBDIR
- bool mutationAdaptive

  the `non_adaptive` specification for the coliny_ea method in MethodSCOLIBEA
- bool showMiscOptions

  the `show_misc_options` specification in MethodSCOLIBDC
- StringArray miscOptions

  the `misc_options` specification in MethodSCOLIBDC
- Real solnTarget

  the `solution_target` specification in MethodSCOLIBDC
- Real crossoverRate

  the `crossover_rate` specification for EA methods in MethodSCOLIBEA
- Real mutationRate

  the `mutation_rate` specification for EA methods in MethodSCOLIBEA
- Real mutationScale

  the `mutation_scale` specification for EA methods in MethodSCOLIBEA
- Real mutationMinScale

  the `min_scale` specification for mutation in EA methods in MethodSCOLIBEA
- Real initDelta

  the `initial_delta` specification for APPS/COBYLA/PS/SW methods in MethodAPPS, MethodSCOLIBCOB, MethodSCOLIBPS, and MethodSCOLIBSW
- Real threshDelta

  the `threshold_delta` specification for APPS/COBYLA/PS/SW methods in MethodAPPS, MethodSCOLIBCOB, MethodSCOLIBPS, and MethodSCOLIBSW
- Real contractFactor

  the `contraction_factor` specification for APPS/PS/SW methods in MethodAPPS, MethodSCOLIBPS, and MethodSCOLIBSW
- int newSolnsGenerated

  the `new_solutions_generated` specification for GA/EPSA methods in MethodSCOLIBEA
- int numberRetained

  the integer assignment to random, chc, or elitist in the `replacement_type` specification for GA/EPSA methods in MethodSCOLIBEA
- bool expansionFlag

the `no_expansion` *specification for APPS/PS/SW methods in MethodAPPS, MethodSCOLIBPS, and MethodSCOLIBSW*

- int expandAfterSuccess

  the `expand_after_success` *specification for PS/SW methods in MethodSCOLIBPS and MethodSCOLIBSW*
- int contractAfterFail

  the `contract_after_failure` *specification for the SW method in MethodSCOLIBSW*
- int mutationRange

  the `mutation_range` *specification for the pga_int method in MethodSCOLIBEA*
- int totalPatternSize

  the `total_pattern_size` *specification for PS methods in MethodSCOLIBPS*
- bool randomizeOrderFlag

  the `stochastic` *specification for the PS method in MethodSCOLIBPS*
- String selectionPressure

  the `fitness_type` *specification for EA methods in MethodSCOLIBEA*
- String replacementType

  the `replacement_type` *specification for EA methods in MethodSCOLIBEA*
- String crossoverType

  the `crossover_type` *specification for EA methods in MethodSCOLIBEA*
- String mutationType

  the `mutation_type` *specification for EA methods in MethodSCOLIBEA*
- String exploratoryMoves

  the `exploratory_moves` *specification for the PS method in MethodSCOLIBPS*
- String patternBasis

  the `pattern_basis` *specification for APPS/PS methods in MethodAPPS and MethodSCOLIBPS*
- String betaSolverName

  beta solvers don't need documentation
- String evalSynchronize

  the `synchronization` *setting for parallel pattern search methods in MethodSCOLIBPS and MethodAPPS*
- size_t numCrossPoints

  The number of crossover points or multi-point schemes.
- size_t numParents

  The number of parents to use in a crossover operation.
- size_t numOffspring

  The number of children to produce in a crossover operation.
- String fitnessType

  the fitness assessment operator to use.
- String convergenceType

  The means by which this JEGA should converge.
- Real percentChange

  The minimum percent change before convergence for a fitness tracker converger.
- size_t numGenerations

  The number of generations over which a fitness tracker converger should track.
- Real fitnessLimit

The cutoff value for survival in fitness limiting selectors (e.g., below_limit selector).

- Real shrinkagePercent

  The minimum percentage of the requested number of selections that must take place on each call to the selector (0, 1).
- String nichingType

  The niching type.
- RealVector nicheVector

  The discretization percentage along each objective.
- size_t numDesigns

  The maximum number of designs to keep when using the max_designs nicher.
- String postProcessorType

  The post processor type.
- RealVector distanceVector

  The discretization percentage along each objective.
- String initializationType

  The means by which the JEGA should initialize the population.
- String flatFile

  The filename to use for initialization.
- String logFile

  The filename to use for logging.
- int populationSize

  the `population_size` *specification for GA methods in MethodSCOLIBEA*
- bool printPopFlag

  The `print_each_pop` *flag to set the printing of the population at each generation.*
- Real volBoxSize

  the `volume_boxsize_limit` *for the DIRECT method in MethodNCSUDC*
- int numSymbols

  the `symbols` *specification for DACE methods*
- bool mainEffectsFlag

  the `main_effects` *specification for sampling methods in MethodDDACE)*
- bool latinizeFlag

  the `latinize` *specification for FSU QMC and CVT methods in MethodFSUDACE*
- bool volQualityFlag

  the `quality_metrics` *specification for sampling methods (FSU QMC and CVT methods in MethodFSUDACE)*
- IntVector sequenceStart

  the `sequenceStart` *specification in MethodFSUDACE*
- IntVector sequenceLeap

  the `sequenceLeap` *specification in MethodFSUDACE*
- IntVector primeBase

  the `primeBase` *specification in MethodFSUDACE*
- int numTrials

  the `numTrials` *specification in MethodFSUDACE*
- String trialType

the `trial_type` *specification in MethodFSUDACE*

- int randomSeed

  the `seed` *specification for COLINY, NonD, & DACE methods*

- String historyFile

  the `HISTORY_FILE` *specification for NOMAD*

- String displayFormat

  the `DISPLAY_STATS` *specification for NOMAD*

- Real vns

  the `VNS` *specification for NOMAD*

- bool showAllEval

  the `DISPLAY_ALL_EVAL` *specification for NOMAD*

- int numSamples

  the `samples` *specification for NonD & DACE methods*

- bool fixedSeedFlag

  flag for fixing the value of the seed among different NonD/DACE sample sets. This results in the use of the same sampling stencil/pattern throughout an execution with repeated sampling.

- bool fixedSequenceFlag

  flag for fixing the sequence for Halton or Hammersley QMC sample sets. This results in the use of the same sampling stencil/pattern throughout an execution with repeated sampling.

- int previousSamples

  the number of previous samples when augmenting a LHS sample

- bool vbdFlag

  the `var_based_decomp` *specification for a variety of sampling methods*

- Real vbdDropTolerance

  the `var_based_decomp` *tolerance for omitting index output*

- bool backfillFlag

  the `backfill` *option allows one to augment in LHS sample by enforcing the addition of unique discrete variables to the sample*

- unsigned short vbdOrder

  a sub-specification of vbdFlag: interaction order limit for calculation/output of component VBD indices

- short covarianceControl

  restrict the calculation of a full response covariance matrix for high dimensional outputs: {DEFAULT,DIAGONAL,FULL}_COVARIANCE

- String rngName

  the `basic` *random-number generator for NonD*

- short refinementType

  refinement type for stochastic expansions from dimension refinement keyword group

- short refinementControl

  refinement control for stochastic expansions from dimension refinement keyword group

- short nestingOverride

  override for default point nesting policy: NO_NESTING_OVERRIDE, NESTED, or NON_NESTED

- short growthOverride

  override for default point growth restriction policy: NO_GROWTH_OVERRIDE, RESTRICTED, or UNRESTRICTED

- short expansionType

  enumeration for u-space type that defines u-space variable targets for probability space transformations: EXTENDED_U (default), ASKEY_U, STD_NORMAL_U, or STD_UNIFORM_U
- bool piecewiseBasis

  boolean indicating presence of `piecewise` *keyword*
- short expansionBasisType

  enumeration for type of basis in sparse grid interpolation (Pecos::{NODAL,HIERARCHICAL}_INTERPOLANT) or regression (Pecos::{TENSOR_PRODUCT,TOTAL_ORDER,ADAPTED}_BASIS).
- UShortArray expansionOrder

  the `expansion_order` *specification in MethodNonDPCE*
- SizetArray expansionSamples

  the `expansion_samples` *specification in MethodNonDPCE*
- String expansionSampleType

  allows for incremental PCE construction using the `incremental_lhs` *specification in MethodNonDPCE*
- UShortArray quadratureOrder

  the `quadrature_order` *specification in MethodNonDPCE and MethodNonDSC*
- UShortArray sparseGridLevel

  the `sparse_grid_level` *specification in MethodNonDPCE, MethodNonDSC, and other stochastic expansion-enabled methods*
- RealVector anisoDimPref

  the `dimension_preference` *specification for tensor and sparse grids and expansion orders in MethodNonDPCE and MethodNonDSC*
- unsigned short cubIntOrder

  the `cubature_integrand` *specification in MethodNonDPCE*
- SizetArray collocationPoints

  the `collocation_points` *specification in MethodNonDPCE*
- Real collocationRatio

  the `collocation_ratio` *specification in MethodNonDPCE*
- Real collocRatioTermsOrder

  order applied to the number of expansion terms when applying or computing the collocation ratio within regression PCE; based on the `ratio_order` *specification in MethodNonDPCE*
- short regressionType

  type of regression: LS, OMP, BP, BPDN, LARS, or LASSO
- short lsRegressionType

  type of least squares regression: SVD or EQ_CON_QR
- RealVector regressionNoiseTol

  noise tolerance(s) for OMP, BPDN, LARS, and LASSO
- Real regressionL2Penalty

  L2 regression penalty for a variant of LASSO known as the elastic net method (default of 0 gives standard LASSO)
- bool crossValidation

  flag indicating the use of cross-validation across expansion orders (given a prescribed maximum order) and, for some methods, noise tolerances
- unsigned short adaptedBasisAdvancements

initial grid level for the ADAPTED_BASIS_EXPANDING_FRONT approach to defining the candidate basis for sparse recovery (compressed sensing)

- bool normalizedCoeffs

  flag indicating the output of PCE coefficients corresponding to normalized basis polynomials

- String pointReuse

  allows PCE construction to reuse points from previous sample sets or data import using the `reuse_points` *specification in MethodNonDPCE*

- bool tensorGridFlag

  flag for usage of a sub-sampled set of tensor-product grid points within regression PCE; based on the `tensor_grid` *specification in MethodNonDPCE*

- UShortArray tensorGridOrder

  order of tensor-product grid points that are sub-sampled within orthogonal least interpolation PCE; based on the `tensor_grid` *specification in MethodNonDPCE*

- String expansionImportFile

  the `import_expansion_file` *specification in MethodNonDPCE*

- String expansionExportFile

  the `export_expansion_file` *specification in MethodNonDPCE*

- unsigned short sampleType

  the `sample_type` *specification in MethodNonDMC, MethodNonDPCE, and MethodNonDSC*

- unsigned short reliabilitySearchType

  the type of limit state search in MethodNonDLocalRel (`x_taylor_mean`*,* `x_taylor_mpp`*,* `x_two_point`*,* `u_taylor_mean`*,* `u_taylor_mpp`*,* `u_two_point`*, or* `no_approx`*) or MethodNonDGlobalRel (*`x_gaussian_process` *or* `u_gaussian_process`*)*

- String reliabilityIntegration

  the `first_order` *or* `second_order` *integration selection in MethodNonDLocalRel*

- unsigned short integrationRefine

  the `import`*,* `adapt_import`*, or* `mm_adapt_import` *integration refinement selection in MethodNonDLocalRel, MethodNonDPCE, and MethodNonDSC*

- int refineSamples

  the `refinement_samples` *selection in MethodNonDLocalRel, MethodNonDPCE, and MethodNonDSC*

- short distributionType

  the `distribution` `cumulative` *or* `complementary` *specification in MethodNonD*

- short responseLevelTarget

  the `compute` `probabilities`*,* `reliabilities`*, or* `gen_reliabilities` *specification in MethodNonD*

- short responseLevelTargetReduce

  the `system` `series` *or* `parallel` *specification in MethodNonD*

- RealVectorArray responseLevels

  the `response_levels` *specification in MethodNonD*

- RealVectorArray probabilityLevels

  the `probability_levels` *specification in MethodNonD*

- RealVectorArray reliabilityLevels

  the `reliability_levels` *specification in MethodNonD*

- RealVectorArray genReliabilityLevels

  the `gen_reliability_levels` *specification in MethodNonD*

- int emulatorSamples

  the number of samples to construct a GP emulator for Bayesian calibration methods (MethodNonDBayesCalib)
- int emulatorOrder

  The total order to be used in construction of a VPS surrogate.
- short emulatorType

  the `emulator` *specification in MethodNonDBayesCalib*
- String mcmcType

  the `mcmc` *type specification in MethodNonDBayesCalib*
- String rejectionType

  the `rejection` *type specification in MethodNonDBayesCalib*
- String metropolisType

  the `metropolis` *type specification in MethodNonDBayesCalib*
- RealVector proposalCovScale

  the `proposal` *covariance scale factor in MethodNonDBayesCalib*
- Real likelihoodScale

  the `likelihood` *scale factor in MethodNonDBayesCalib*
- String fitnessMetricType

  the `fitness` *metric type specification in MethodNonDAdaptive*
- String batchSelectionType

  the `batch` *selection type specification in MethodNonDAdaptive*
- int batchSize

  The size of the batch (e.g. number of supplemental points added) to be added to be added to the build points for an emulator at each iteration.
- String lipschitzType

  the `Lipschitz` *type specification in MethodNonDPOFDarts (e.g. either local or global estimation)*
- bool calibrateSigmaFlag

  flag to indicate if the sigma terms should be calibrated in MethodNonDBayesCalib
- int numChains

  number of concurrent chains
- int numCR

  number of CR-factors
- int crossoverChainPairs

  number of crossover chain pairs
- Real grThreshold

  threshold for the Gelmin-Rubin statistic
- int jumpStep

  how often to perform a long jump in generations
- RealVector finalPoint

  the `final_point` *specification in MethodPSVPS*
- RealVector stepVector

  the `step_vector` *specification in MethodPSVPS and MethodPSCPS*
- int numSteps

  the `num_steps` *specification in MethodPSVPS*

- IntVector stepsPerVariable

  the `deltas_per_variable` specification in MethodPSCPS
- RealVector listOfPoints

  the `list_of_points` specification in MethodPSLPS
- String pstudyFilename

  the `import_points_file` spec for a file-based parameter study
- bool pstudyFileAnnotated

  whether the parameter study points file is annotated
- bool pstudyFileActive

  whether to import active variables only
- UShortArray varPartitions

  the `partitions` specification for PStudy method in MethodPSMPS
- Real refinementRate

  rate of mesh refinement in Richardson extrapolation
- String approxImportFile

  the file name for point import in surrogate-based methods
- bool approxImportAnnotated

  whether the point import file is annotated (default true)
- bool approxImportActive

  whether to import active variables only
- String approxExportFile

  the file name for point export in surrogate-based methods
- bool approxExportAnnotated

  whether the point export file is annotated (default true)

Private Member Functions

- DataMethodRep ()

  constructor
- ~DataMethodRep ()

  destructor
- void write (std::ostream &s) const

  write a DataInterfaceRep object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataInterfaceRep object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataInterfaceRep object to a packed MPI buffer

Private Attributes

- int referenceCount

  number of handle objects sharing this dataMethodRep

Friends

- class DataMethod

  the handle class can access attributes of the body class directly

13.24.1 Detailed Description

Body class for method specification data.

The DataMethodRep class is used to contain the data from a method keyword specification. Default values are managed in the DataMethodRep constructor. Data is public to avoid maintaining set/get functions, but is still encapsulated within ProblemDescDB since ProblemDescDB::dataMethodList is private.

The documentation for this class was generated from the following files:

- DataMethod.hpp
- DataMethod.cpp

13.25 DataModel Class Reference

Handle class for model specification data.

Public Member Functions

- DataModel ()

  constructor
- DataModel (const DataModel &)

  copy constructor
- ~DataModel ()

  destructor
- DataModel & operator= (const DataModel &)

  assignment operator
- void write (std::ostream &s) const

  write a DataModel object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataModel object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataModel object to a packed MPI buffer
- DataModelRep * data_rep ()

  return dataModelRep

Static Public Member Functions

- static bool id_compare (const DataModel &dm, const std::string &id)

  compares the idModel attribute of DataModel objects

Private Attributes

- DataModelRep * dataModelRep

  pointer to the body (handle-body idiom)

Friends

- class **ProblemDescDB**
- class **NIDRProblemDescDB**

13.25.1 Detailed Description

Handle class for model specification data.

The DataModel class is used to provide a memory management handle for the data in DataModelRep. It is populated by IDRProblemDescDB::model_kwhandler() and is queried by the ProblemDescDB::get_<datatype>() functions. A list of DataModel objects is maintained in ProblemDescDB::dataModelList, one for each model specification in an input file.

The documentation for this class was generated from the following files:

- DataModel.hpp
- DataModel.cpp

13.26 DataModelRep Class Reference

Body class for model specification data.

Public Attributes

- String idModel

  string identifier for the model specification data set (from the `id_model` specification in ModelIndControl)
- String modelType

  model type selection: single, surrogate, or nested (from the model type specification in ModelIndControl)
- String variablesPointer

  string pointer to the variables specification to be used by this model (from the `variables_pointer` specification in ModelIndControl)
- String interfacePointer

  string pointer to the interface specification to be used by this model (from the `interface_pointer` specification in ModelSingle and the `optional_interface_pointer` specification in ModelNested)
- String responsesPointer

  string pointer to the responses specification to be used by this model (from the `responses_pointer` specification in ModelIndControl)
- bool hierarchicalTags

  whether this model and its children will add hierarchy-based tags to eval ids
- String subMethodPointer

  pointer to a sub-iterator used for global approximations (from the `dace_method_pointer` specification in ModelSurrG) or by nested models (from the `sub_method_pointer` specification in ModelNested)
- IntSet surrogateFnIndices

  array specifying the response function set that is approximated
- String surrogateType

  the selected surrogate type: local_taylor, multipoint_tana, global_(neural_network,mars,orthogonal_polynomial,gaussian, polynomial,kriging), or hierarchical
- String truthModelPointer

pointer to the model specification for constructing the truth model used in building local, multipoint, and hierarchical approximations (from the `actual_model_pointer` *specification in ModelSurrL and ModelSurrMP and the* `high_fidelity_model_pointer` *specification in ModelSurrH)*

- String lowFidelityModelPointer

  pointer to the low fidelity model specification used in hierarchical approximations (from the `low_fidelity_model_pointer` *specification in ModelSurrH)*

- int pointsTotal

  user-specified lower bound on total points with which to build the model (if reuse_points < pointsTotal, new samples will make up the difference)

- short pointsManagement

  points management configuration for DataFitSurrModel: DEFAULT_POINTS, MINIMUM_POINTS, or RECOMMENDED_POINTS

- String approxPointReuse

  sample reuse selection for building global approximations: none, all, region, or file (from the `reuse_samples` *specification in ModelSurrG)*

- String approxImportFile

  the file name from the `import_points_file` *specification in ModelSurrG*

- bool approxImportAnnotated

  whether the point import file is annotated (default true)

- bool approxImportActive

  whether to import active variables only

- String approxExportFile

  the file name from the `export_points_file` *specification in ModelSurrG*

- bool approxExportAnnotated

  whether the point export file is annotated (default true)

- String approxExportModelFile

  the file name from the `export_model_file` *specification in ModelSurrG*

- short approxCorrectionType

  correction type for global and hierarchical approximations: NO_CORRECTION, ADDITIVE_CORRECTION, MULTIPLICATIVE_CORRECTION, or COMBINED_CORRECTION (from the `correction` *specification in ModelSurrG and ModelSurrH)*

- short approxCorrectionOrder

  correction order for global and hierarchical approximations: 0, 1, or 2 (from the `correction` *specification in ModelSurrG and ModelSurrH)*

- bool modelUseDerivsFlag

  flags the use of derivatives in building global approximations (from the `use_derivatives` *specification in ModelSurrG)*

- short polynomialOrder

  scalar integer indicating the order of the polynomial approximation (1=linear, 2=quadratic, 3=cubic; from the `polynomial` *specification in ModelSurrG)*

- RealVector krigingCorrelations

  vector of correlations used in building a kriging approximation (from the `correlations` *specification in ModelSurrG)*

- String krigingOptMethod

  optimization method to use in finding optimal correlation parameters: none, sampling, local, global

- short krigingMaxTrials

  maximum number of trials in optimization of kriging correlations
- RealVector krigingMaxCorrelations

  upper bound on kriging correlation vector
- RealVector krigingMinCorrelations

  lower bound on kriging correlation vector
- Real krigingNugget

  nugget value for kriging
- short krigingFindNugget

  option to have Kriging find the best nugget value to use
- short mlsPolyOrder

  polynomial order for moving least squares approximation
- short mlsWeightFunction

  weight function for moving least squares approximation
- short rbfBases

  bases for radial basis function approximation
- short rbfMaxPts

  maximum number of points for radial basis function approximation
- short rbfMaxSubsets

  maximum number of subsets for radial basis function approximation
- short rbfMinPartition

  minimum partition for radial basis function approximation
- short marsMaxBases

  maximum number of bases for MARS approximation
- String marsInterpolation

  interpolation type for MARS approximation
- short annRandomWeight

  random weight for artificial neural network approximation
- short annNodes

  number of nodes for artificial neural network approximation
- Real annRange

  range for artificial neural network approximation
- String trendOrder

  scalar integer indicating the order of the Gaussian process mean (0= constant, 1=linear, 2=quadratic, 3=cubic); from the `gaussian_process` *specification in ModelSurrG)*
- bool pointSelection

  flag indicating the use of point selection in the Gaussian process
- StringArray diagMetrics

  List of diagnostic metrics the user requests to assess the goodness of fit for a surrogate model.
- bool crossValidateFlag

  flag indicating the use of cross validation on the metrics specified
- int numFolds

  number of folds to perform in cross validation

- Real percentFold

  percentage of data to withhold for cross validation process
- bool pressFlag

  flag indicating the use of PRESS on the metrics specified
- String approxChallengeFile

  the file name from the `challenge_points_file` *specification in ModelSurrG*
- bool approxChallengeAnnotated

  whether the challenge data file is annotated (default true)
- bool approxChallengeActive

  whether to import active variables only
- String optionalInterfRespPointer

  string pointer to the responses specification used by the optional interface in nested models (from the `optional_interface_responses_pointer` *specification in ModelNested)*
- StringArray primaryVarMaps

  the primary variable mappings used in nested models for identifying the lower level variable targets for inserting top level variable values (from the `primary_variable_mapping` *specification in ModelNested)*
- StringArray secondaryVarMaps

  the secondary variable mappings used in nested models for identifying the (distribution) parameter targets within the lower level variables for inserting top level variable values (from the `secondary_variable_mapping` *specification in ModelNested)*
- RealVector primaryRespCoeffs

  the primary response mapping matrix used in nested models for weighting contributions from the sub-iterator responses in the top level (objective) functions (from the `primary_response_mapping` *specification in ModelNested)*
- RealVector secondaryRespCoeffs

  the secondary response mapping matrix used in nested models for weighting contributions from the sub-iterator responses in the top level (constraint) functions (from the `secondary_response_mapping` *specification in ModelNested)*
- int subMethodServers

  number of servers for concurrent sub-iterator parallelism
- int subMethodProcs

  number of processors for each concurrent sub-iterator partition
- short subMethodScheduling

  scheduling approach for concurrent sub-iterator parallelism: {DEFAULT,MASTER,PEER}_SCHEDULING

Private Member Functions

- DataModelRep ()

  constructor
- ~DataModelRep ()

  destructor
- void write (std::ostream &s) const

  write a DataModelRep object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataModelRep object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataModelRep object to a packed MPI buffer

Private Attributes

- int referenceCount

  number of handle objects sharing this dataModelRep

Friends

- class DataModel

  the handle class can access attributes of the body class directly

13.26.1 Detailed Description

Body class for model specification data.

The DataModelRep class is used to contain the data from a model keyword specification. Default values are managed in the DataModelRep constructor. Data is public to avoid maintaining set/get functions, but is still encapsulated within ProblemDescDB since ProblemDescDB::dataModelList is private.

The documentation for this class was generated from the following files:

- DataModel.hpp
- DataModel.cpp

13.27 DataResponses Class Reference

Handle class for responses specification data.

Public Member Functions

- DataResponses ()

  constructor
- DataResponses (const DataResponses &)

  copy constructor
- ~DataResponses ()

  destructor
- DataResponses & operator= (const DataResponses &)

  assignment operator
- void write (std::ostream &s) const

  write a DataResponses object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataResponses object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataResponses object to a packed MPI buffer
- DataResponsesRep * data_rep ()

  return dataRespRep

Static Public Member Functions

- static bool id_compare (const DataResponses &dr, const std::string &id)

  compares the idResponses attribute of DataResponses objects

Private Attributes

- DataResponsesRep * dataRespRep

  pointer to the body (handle-body idiom)

Friends

- class **ProblemDescDB**
- class **NIDRProblemDescDB**

13.27.1 Detailed Description

Handle class for responses specification data.

The DataResponses class is used to provide a memory management handle for the data in DataResponsesRep. It is populated by IDRProblemDescDB::responses_kwhandler() and is queried by the ProblemDescDB::get_<datatype>() functions. A list of DataResponses objects is maintained in ProblemDescDB::dataResponsesList, one for each responses specification in an input file.

The documentation for this class was generated from the following files:

- DataResponses.hpp
- DataResponses.cpp

13.28 DataResponsesRep Class Reference

Body class for responses specification data.

Public Attributes

- String idResponses

  string identifier for the responses specification data set (from the `id_responses` specification in RespSetId)
- StringArray responseLabels

  the response labels array (from the `response_descriptors` specification in RespLabels)
- size_t numObjectiveFunctions

  number of objective functions (from the `num_objective_functions` specification in RespFnOpt)
- size_t numNonlinearIneqConstraints

  number of nonlinear inequality constraints (from the `num_nonlinear_inequality_constraints` specification in RespFnOpt)
- size_t numNonlinearEqConstraints

  number of nonlinear equality constraints (from the `num_nonlinear_equality_constraints` specification in RespFnOpt)
- size_t numLeastSqTerms

  number of least squares terms (from the `num_least_squares_terms` specification in RespFnLS)
- size_t numResponseFunctions

  number of generic response functions (from the `num_response_functions` specification in RespFnGen)
- size_t numScalarObjectiveFunctions

  scalar_objectives: number of objective functions which are scalar
- size_t numFieldObjectiveFunctions

field_objectives: number of objective functions which are field-valued

- size_t numScalarLeastSqTerms

  scalar_calibration_terms: number of calibration terms which are scalar

- size_t numFieldLeastSqTerms

  field_calibration_terms: number of calibration terms which are field-valued

- size_t numScalarResponseFunctions

  scalar_responses: number of response functions which are scalar

- size_t numFieldResponseFunctions

  field_responses: number of response functions which are field-valued

- StringArray primaryRespFnSense

  optimization sense for each objective function: minimize or maximize

- RealVector primaryRespFnWeights

  vector of weightings for multiobjective optimization or weighted nonlinear least squares (from the `multi_objective_weights` *specification in RespFnOpt and the* `least_squares_weights` *specification in RespFnLS)*

- RealVector nonlinearIneqLowerBnds

  vector of nonlinear inequality constraint lower bounds (from the `nonlinear_inequality_lower_bounds` *specification in RespFnOpt)*

- RealVector nonlinearIneqUpperBnds

  vector of nonlinear inequality constraint upper bounds (from the `nonlinear_inequality_upper_bounds` *specification in RespFnOpt)*

- RealVector nonlinearEqTargets

  vector of nonlinear equality constraint targets (from the `nonlinear_equality_targets` *specification in RespFnOpt)*

- StringArray primaryRespFnScaleTypes

  vector of primary response function scaling types (from the `objective_function_scale_types` *specification in RespFnOpt and the* `least_squares_term_scale_types` *specification in RespFnLS)*

- RealVector primaryRespFnScales

  vector of primary response function scaling factors (from the `objective_function_scales` *specification in RespFnOpt and the* `least_squares_term_scales` *specification in RespFnLS)*

- StringArray nonlinearIneqScaleTypes

  vector of nonlinear inequality constraint scaling types (from the `nonlinear_inequality_scale_types` *specification in RespFnOpt)*

- RealVector nonlinearIneqScales

  vector of nonlinear inequality constraint scaling factors (from the `nonlinear_inequality_scales` *specification in RespFnOpt)*

- StringArray nonlinearEqScaleTypes

  vector of nonlinear equality constraint scaling types (from the `nonlinear_equality_scale_types` *specification in RespFnOpt)*

- RealVector nonlinearEqScales

  vector of nonlinear equality constraint scaling factors (from the `nonlinear_equality_scales` *specification in RespFnOpt)*

- size_t numExperiments

  number of distinct experiments in experimental data

- size_t numExpConfigVars

number of experimental configuration vars (state variables) in each row of data

- size_t numExpStdDeviations

  whether to read num_responses standard deviations from each row of data file

- RealVector expConfigVars

  list of num_experiments x num_config_vars configuration variable values

- RealVector expObservations

  list of num_calibration_terms observation data

- RealVector expStdDeviations

  list of 1 or num_calibration_terms observation standard deviations

- String expDataFileName

  name of experimental data file containing response data (with optional state variable and sigma data) to read

- bool expDataFileAnnotated

  whether the experimental data is in annotated format

- String gradientType

  gradient type: none, numerical, analytic, or mixed (from the `no_gradients`*,* `numerical_gradients`*,* `analytic_gradients`*, and* `mixed_gradients` *specifications in RespGrad)*

- String hessianType

  Hessian type: none, numerical, quasi, analytic, or mixed (from the `no_hessians`*,* `numerical_hessians`*,* `quasi_hessians`*,* `analytic_hessians`*, and* `mixed_hessians` *specifications in RespHess)*

- bool ignoreBounds

  option to ignore bounds when doing finite differences (default is to honor bounds)

- bool centralHess

  Temporary(?) option to use old 2nd-order diffs when computing finite-difference Hessians; default is forward differences.

- String quasiHessianType

  quasi-Hessian type: bfgs, damped_bfgs, or sr1 (from the `bfgs` *and* `sr1` *specifications in RespHess)*

- String methodSource

  numerical gradient method source: dakota or vendor (from the `method_source` *specification in RespGradNum and RespGradMixed)*

- String intervalType

  numerical gradient interval type: forward or central (from the `interval_type` *specification in RespGradNum and RespGradMixed)*

- RealVector fdGradStepSize

  vector of finite difference step sizes for numerical gradients, one step size per active continuous variable, used in computing 1st-order forward or central differences (from the `fd_gradient_step_size` *specification in RespGradNum and RespGradMixed)*

- String fdGradStepType

  type of finite difference step to use for numerical gradient: relative - step length is relative to x absolute - step length is what is specified bounds - step length is relative to range of x

- RealVector fdHessStepSize

  vector of finite difference step sizes for numerical Hessians, one step size per active continuous variable, used in computing 1st-order gradient-based differences and 2nd-order function-based differences (from the `fd_hessian_step_size` *specification in RespHessNum and RespHessMixed)*

- String fdHessStepType

type of finite difference step to use for numerical Hessian: relative - step length is relative to x absolute - step length is what is specified bounds - step length is relative to range of x

- IntSet idNumericalGrads

  mixed gradient numerical identifiers (from the `id_numerical_gradients` specification in RespGradMixed)
- IntSet idAnalyticGrads

  mixed gradient analytic identifiers (from the `id_analytic_gradients` specification in RespGradMixed)
- IntSet idNumericalHessians

  mixed Hessian numerical identifiers (from the `id_numerical_hessians` specification in RespHessMixed)
- IntSet idQuasiHessians

  mixed Hessian quasi identifiers (from the `id_quasi_hessians` specification in RespHessMixed)
- IntSet idAnalyticHessians

  mixed Hessian analytic identifiers (from the `id_analytic_hessians` specification in RespHessMixed)
- IntVector fieldLengths

  number of entries in each field
- IntVector numCoordsPerField

  number of coordinates per field
- RealVector coordsPerField

  values of coordinates per field
- String coordDataFileName

  data file which contains the values of the coordinates per field
- String configDataFileName

  Field data related storage: data file which contains values of configuration variables for experiments.
- String fieldDataFileName

  Field data related storage: data file which contains values of field responses for experiments.
- String fieldCoordDataFileName

  Field data related storage: data file which contains values of coordinates of field responses for experiments.
- String sigmaDataFileName

  Field data related storage: data file which contains values of measurement error for field responses for experiments.
- StringArray sigmaType

  Array which specifies the sigma type per response (none, one constant value, one per response (vector) or a full covariance matrix.

Private Member Functions

- DataResponsesRep ()

  constructor
- ~DataResponsesRep ()

  destructor
- void write (std::ostream &s) const

  write a DataResponsesRep object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataResponsesRep object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataResponsesRep object to a packed MPI buffer

Private Attributes

- int referenceCount

  number of handle objects sharing this dataResponsesRep

Friends

- class DataResponses

  the handle class can access attributes of the body class directly

13.28.1 Detailed Description

Body class for responses specification data.

The DataResponsesRep class is used to contain the data from a responses keyword specification. Default values are managed in the DataResponsesRep constructor. Data is public to avoid maintaining set/get functions, but is still encapsulated within ProblemDescDB since ProblemDescDB::dataResponsesList is private.

The documentation for this class was generated from the following files:

- DataResponses.hpp
- DataResponses.cpp

13.29 DataVariables Class Reference

Handle class for variables specification data.

Public Member Functions

- DataVariables ()

  constructor
- DataVariables (const DataVariables &)

  copy constructor
- ~DataVariables ()

  destructor
- DataVariables operator= (const DataVariables &)

  assignment operator
- bool operator== (const DataVariables &)

  equality operator
- void write (std::ostream &s) const

  write a DataVariables object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataVariables object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataVariables object to a packed MPI buffer
- DataVariablesRep * data_rep ()

  return dataVarsRep
- size_t design ()

return total number of design variables

- size_t aleatory_uncertain ()

  return total number of aleatory uncertain variables
- size_t epistemic_uncertain ()

  return total number of epistemic uncertain variables
- size_t uncertain ()

  return total number of uncertain variables
- size_t state ()

  return total number of state variables
- size_t continuous_variables ()

  return total number of continuous variables
- size_t discrete_variables ()

  return total number of discrete variables
- size_t total_variables ()

  return total number of variables

Static Public Member Functions

- static bool id_compare (const DataVariables &dv, const std::string &id)

  compares the idVariables attribute of DataVariables objects

Private Attributes

- DataVariablesRep * dataVarsRep

  pointer to the body (handle-body idiom)

Friends

- class **ProblemDescDB**
- class **NIDRProblemDescDB**

13.29.1 Detailed Description

Handle class for variables specification data.

The DataVariables class is used to provide a memory management handle for the data in DataVariables-Rep. It is populated by IDRProblemDescDB::variables_kwhandler() and is queried by the ProblemDescDB::get_-<datatype>() functions. A list of DataVariables objects is maintained in ProblemDescDB::dataVariablesList, one for each variables specification in an input file.

The documentation for this class was generated from the following files:

- DataVariables.hpp
- DataVariables.cpp

13.30 DataVariablesRep Class Reference

Body class for variables specification data.

Public Attributes

- String idVariables

  string identifier for the variables specification data set (from the `id_variables` specification in VarSetId)
- short varsView

  user selection/override of variables view: {DEFAULT,ALL,DESIGN, UNCERTAIN,ALEATORY_UNCERTAIN,EPISTEMIC_UNCERTAIN,STATE}_VIEW
- short varsDomain

  user selection/override of variables domain: {DEFAULT,MIXED,RELAXED}_DOMAIN
- bool uncertainVarsInitPt

  flag indicating user specification of initial points (for local optimization-based UQ methods) for at least one uncertain variable type
- size_t numContinuousDesVars

  number of continuous design variables (from the `continuous_design` specification in VarDV)
- size_t numDiscreteDesRangeVars

  number of discrete design variables defined by an integer range (from the `discrete_design_range` specification in VarDV)
- size_t numDiscreteDesSetIntVars

  number of discrete design variables defined by a set of integers (from the `discrete_design_set` integer specification in VarDV)
- size_t numDiscreteDesSetStrVars

  number of discrete design variables defined by a set of strings (from the `discrete_design_set` string specification in VarDV)
- size_t numDiscreteDesSetRealVars

  number of discrete design variables defined by a set of reals (from the `discrete_design_set` real specification in VarDV)
- size_t numNormalUncVars

  number of normal uncertain variables (from the `normal_uncertain` specification in VarAUV)
- size_t numLognormalUncVars

  number of lognormal uncertain variables (from the `lognormal_uncertain` specification in VarAUV)
- size_t numUniformUncVars

  number of uniform uncertain variables (from the `uniform_uncertain` specification in VarAUV)
- size_t numLoguniformUncVars

  number of loguniform uncertain variables (from the `loguniform_uncertain` specification in VarAUV)
- size_t numTriangularUncVars

  number of triangular uncertain variables (from the `triangular_uncertain` specification in VarAUV)
- size_t numExponentialUncVars

  number of exponential uncertain variables (from the `exponential_uncertain` specification in VarAUV)
- size_t numBetaUncVars

  number of beta uncertain variables (from the `beta_uncertain` specification in VarAUV)
- size_t numGammaUncVars

  number of gamma uncertain variables (from the `gamma_uncertain` specification in VarAUV)
- size_t numGumbelUncVars

  number of gumbel uncertain variables (from the `gumbel_uncertain` specification in VarAUV)
- size_t numFrechetUncVars

number of frechet uncertain variables (from the `frechet_uncertain` *specification in VarAUV)*

- size_t numWeibullUncVars

  number of weibull uncertain variables (from the `weibull_uncertain` *specification in VarAUV)*

- size_t numHistogramBinUncVars

  number of histogram bin uncertain variables (from the `histogram_bin_uncertain` *specification in VarAUV)*

- size_t numPoissonUncVars

  number of Poisson uncertain variables (from the `poisson_uncertain` *specification in VarAUV)*

- size_t numBinomialUncVars

  number of binomial uncertain variables (from the `binomial_uncertain` *specification in VarAUV)*

- size_t numNegBinomialUncVars

  number of negative binomial uncertain variables (from the `negative_binomial_uncertain` *specification in VarAUV)*

- size_t numGeometricUncVars

  number of geometric uncertain variables (from the `geometric_uncertain` *specification in VarAUV*

- size_t numHyperGeomUncVars

  number of hypergeometric uncertain variables (from the `hypergeometric_uncertain` *specification in VarAUV))*

- size_t numHistogramPtIntUncVars

  number of integer-valued histogram point uncertain variables (from the `histogram_point_uncertain` *specification in VarAUV)*

- size_t numHistogramPtStrUncVars

  number of string-valued histogram point uncertain variables (from the `histogram_point_uncertain` *specification in VarAUV)*

- size_t numHistogramPtRealUncVars

  number of real-valued histogram point uncertain variables (from the `histogram_point_uncertain` *specification in VarAUV)*

- size_t numContinuousIntervalUncVars

  number of continuous epistemic interval uncertain variables (from the `continuous_interval_uncertain` *specification in VarEUV)*

- size_t numDiscreteIntervalUncVars

  number of discrete epistemic interval uncertain variables (from the `discrete_interval_uncertain` *specification in VarEUV)*

- size_t numDiscreteUncSetIntVars

  number of discrete epistemic uncertain integer set variables (from the `discrete_uncertain_set` *integer specification in VarEUV)*

- size_t numDiscreteUncSetStrVars

  number of discrete epistemic uncertain string set variables (from the `discrete_uncertain_set` *string specification in VarEUV)*

- size_t numDiscreteUncSetRealVars

  number of discrete epistemic uncertain real set variables (from the `discrete_uncertain_set` *real specification in VarEUV)*

- size_t numContinuousStateVars

  number of continuous state variables (from the `continuous_state` *specification in VarSV)*

- size_t numDiscreteStateRangeVars

number of discrete state variables defined by an integer range (from the `discrete_state_range` *specification in VarDV)*

- size_t numDiscreteStateSetIntVars

  number of discrete state variables defined by a set of integers (from the `discrete_state_set` *integer specification in VarDV)*

- size_t numDiscreteStateSetStrVars

  number of discrete state variables defined by a set of strings (from the `discrete_state_set` *string specification in VarDV)*

- size_t numDiscreteStateSetRealVars

  number of discrete state variables defined by a set of reals (from the `discrete_state_set` *real specification in VarDV)*

- RealVector continuousDesignVars

  initial values for the continuous design variables array (from the `continuous_design` `initial_point` *specification in VarDV)*

- RealVector continuousDesignLowerBnds

  lower bounds array for the continuous design variables (from the `continuous_design` `lower_bounds` *specification in VarDV)*

- RealVector continuousDesignUpperBnds

  upper bounds array for the continuous design variables (from the `continuous_design` `upper_bounds` *specification in VarDV)*

- StringArray continuousDesignScaleTypes

  scale types array for the continuous design variables (from the `continuous_design` `scale_types` *specification in VarDV)*

- RealVector continuousDesignScales

  scales array for the continuous design variables (from the `continuous_design` `scales` *specification in VarDV)*

- IntVector discreteDesignRangeVars

  initial values for the discrete design variables defined by an integer range (from the `discrete_design_range` `initial_point` *specification in VarDV)*

- IntVector discreteDesignRangeLowerBnds

  lower bounds array for the discrete design variables defined by an integer range (from the `discrete_design_range` `lower_bounds` *specification in VarDV)*

- IntVector discreteDesignRangeUpperBnds

  upper bounds array for the discrete design variables defined by an integer range(from the `discrete_design_range` `upper_bounds` *specification in VarDV)*

- BitArray discreteDesignRangeCat

  is each ddr var strictly categorical (true) or relaxable (false)

- IntVector discreteDesignSetIntVars

  initial values for the discrete design variables defined by an integer set (from the `discrete_design_set` *integer* `initial_point` *specification in VarDV)*

- StringArray discreteDesignSetStrVars

  initial values for the discrete design variables defined by a string set (from the `discrete_design_set` *string* `initial_point` *specification in VarDV)*

- RealVector discreteDesignSetRealVars

  initial values for the discrete design variables defined by a real set (from the `discrete_design_set` *real* `initial_point` *specification in VarDV)*

- IntSetArray discreteDesignSetInt

  complete set of admissible values for each of the discrete design variables defined by an integer set (from the `discrete_design_set` *integer* `set_values` *specification in VarDV)*
- StringSetArray discreteDesignSetStr

  complete set of admissible values for each of the discrete design variables defined by a string set (from the `discrete_design_set` *string* `set_values` *specification in VarDV)*
- RealSetArray discreteDesignSetReal

  complete set of admissible values for each of the discrete design variables defined by a real set (from the `discrete_design_set` *real* `set_values` *specification in VarDV)*
- BitArray discreteDesignSetIntCat

  is each ddsi var strictly categorical (true) or relaxable (false)
- BitArray discreteDesignSetRealCat

  is each ddsr var strictly categorical (true) or relaxable (false)
- StringArray continuousDesignLabels

  labels array for the continuous design variables (from the `continuous_design` `descriptors` *specification in VarDV)*
- StringArray discreteDesignRangeLabels

  labels array for the discrete design variables defined by an integer range (from the `discrete_design_range` `descriptors` *specification in VarDV)*
- StringArray discreteDesignSetIntLabels

  labels array for the discrete design variables defined by an integer set (from the `discrete_design` *set int* `descriptors` *specification in VarDV)*
- StringArray discreteDesignSetStrLabels

  labels array for the discrete design variables defined by a string set (from the `discrete_design_set` *string* `descriptors` *specification in VarDV)*
- StringArray discreteDesignSetRealLabels

  labels array for the discrete design variables defined by a real set (from the `discrete_design_set` *real;* `descriptors` *specification in VarDV)*
- RealVector normalUncMeans

  means of the normal uncertain variables (from the `means` *specification in VarCAUV_Normal)*
- RealVector normalUncStdDevs

  standard deviations of the normal uncertain variables (from the `std_deviations` *specification in VarCAUV_Normal)*
- RealVector normalUncLowerBnds

  distribution lower bounds for the normal uncertain variables (from the `lower_bounds` *specification in VarCAUV_Normal)*
- RealVector normalUncUpperBnds

  distribution upper bounds for the normal uncertain variables (from the `upper_bounds` *specification in VarCAUV_Normal)*
- RealVector normalUncVars

  initial values of the normal uncertain variables (from the `initial_point` *specification in VarCAUV_Normal)*
- RealVector lognormalUncLambdas

  lambdas (means of the corresponding normals) of the lognormal uncertain variables (from the `lambdas` *specification in VarCAUV_Lognormal)*
- RealVector lognormalUncZetas

zetas (standard deviations of the corresponding normals) of the lognormal uncertain variables (from the `zetas` *specification in VarCAUV_Lognormal)*

- RealVector lognormalUncMeans

  means of the lognormal uncertain variables (from the `means` *specification in VarCAUV_Lognormal)*

- RealVector lognormalUncStdDevs

  standard deviations of the lognormal uncertain variables (from the `std_deviations` *specification in VarCAUV_Lognormal)*

- RealVector lognormalUncErrFacts

  error factors for the lognormal uncertain variables (from the `error_factors` *specification in VarCAUV_Lognormal)*

- RealVector lognormalUncLowerBnds

  distribution lower bounds for the lognormal uncertain variables (from the `lower_bounds` *specification in VarCAUV_Lognormal)*

- RealVector lognormalUncUpperBnds

  distribution upper bounds for the lognormal uncertain variables (from the `upper_bounds` *specification in VarCAUV_Lognormal)*

- RealVector lognormalUncVars

  initial values of the lognormal uncertain variables (from the `initial_point` *specification in VarCAUV_Lognormal)*

- RealVector uniformUncLowerBnds

  distribution lower bounds for the uniform uncertain variables (from the `lower_bounds` *specification in VarCAUV_Uniform)*

- RealVector uniformUncUpperBnds

  distribution upper bounds for the uniform uncertain variables (from the `upper_bounds` *specification in VarCAUV_Uniform)*

- RealVector uniformUncVars

  initial values of the uniform uncertain variables (from the `initial_point` *specification in VarCAUV_Uniform)*

- RealVector loguniformUncLowerBnds

  distribution lower bounds for the loguniform uncertain variables (from the `lower_bounds` *specification in VarCAUV_Loguniform)*

- RealVector loguniformUncUpperBnds

  distribution upper bounds for the loguniform uncertain variables (from the `upper_bounds` *specification in VarCAUV_Loguniform)*

- RealVector loguniformUncVars

  initial values of the loguniform uncertain variables (from the `initial_point` *specification in VarCAUV_Loguniform)*

- RealVector triangularUncModes

  modes of the triangular uncertain variables (from the `modes` *specification in VarCAUV_Triangular)*

- RealVector triangularUncLowerBnds

  distribution lower bounds for the triangular uncertain variables (from the `lower_bounds` *specification in VarCAUV_Triangular)*

- RealVector triangularUncUpperBnds

  distribution upper bounds for the triangular uncertain variables (from the `upper_bounds` *specification in VarCAUV_Triangular)*

- RealVector triangularUncVars

  initial values of the triangular uncertain variables (from the `initial_point` *specification in VarCAUV_Triangular)*

- RealVector exponentialUncBetas

  beta factors for the exponential uncertain variables (from the `betas` *specification in VarCAUV_Exponential)*

- RealVector exponentialUncVars

  initial values of the exponential uncertain variables (from the `initial_point` specification in VarCAUV_Exponential)
- RealVector betaUncAlphas

  alpha factors for the beta uncertain variables (from the `means` specification in VarCAUV_Beta)
- RealVector betaUncBetas

  beta factors for the beta uncertain variables (from the `std_deviations` specification in VarCAUV_Beta)
- RealVector betaUncLowerBnds

  distribution lower bounds for the beta uncertain variables (from the `lower_bounds` specification in VarCAUV_Beta)
- RealVector betaUncUpperBnds

  distribution upper bounds for the beta uncertain variables (from the `upper_bounds` specification in VarCAUV_Beta)
- RealVector betaUncVars

  initial values of the beta uncertain variables (from the `initial_point` specification in VarCAUV_Beta)
- RealVector gammaUncAlphas

  alpha factors for the gamma uncertain variables (from the `alphas` specification in VarCAUV_Gamma)
- RealVector gammaUncBetas

  beta factors for the gamma uncertain variables (from the `betas` specification in VarCAUV_Gamma)
- RealVector gammaUncVars

  initial values of the gamma uncertain variables (from the `initial_point` specification in VarCAUV_Gamma)
- RealVector gumbelUncAlphas

  alpha factors for the gumbel uncertain variables (from the `alphas` specification in VarCAUV_Gumbel)
- RealVector gumbelUncBetas

  beta factors for of the gumbel uncertain variables (from the `betas` specification in VarCAUV_Gumbel)
- RealVector gumbelUncVars

  initial values of the gumbel uncertain variables (from the `initial_point` specification in VarCAUV_Gumbel)
- RealVector frechetUncAlphas

  alpha factors for the frechet uncertain variables (from the `alphas` specification in VarCAUV_Frechet)
- RealVector frechetUncBetas

  beta factors for the frechet uncertain variables (from the `betas` specification in VarCAUV_Frechet)
- RealVector frechetUncVars

  initial values of the frechet uncertain variables (from the `initial_point` specification in VarCAUV_Frechet)
- RealVector weibullUncAlphas

  alpha factors for the weibull uncertain variables (from the `alphas` specification in VarCAUV_Weibull)
- RealVector weibullUncBetas

  beta factors for the weibull uncertain variables (from the `betas` specification in VarCAUV_Weibull)
- RealVector weibullUncVars

  initial values of the weibull uncertain variables (from the `initial_point` specification in VarCAUV_Weibull)
- RealRealMapArray histogramUncBinPairs

  An array for each real-valued bin-based histogram uncertain variable. Each array entry is a map from a real value to its probability. (See continuous linear histogram in LHS manual; from the `histogram_bin_uncertain` specification in VarCAUV_Bin_Histogram). (x,y) ordinate specifications are converted to (x,c) counts within NIDR.
- RealVector histogramBinUncVars

initial values of the histogram bin uncertain variables (from the `initial_point` *specification in VarCAUV_Bin_Histogram)*

- RealVector poissonUncLambdas

  lambdas (rate parameter) for the poisson uncertain variables (from the `lambdas` *specification in VarDAUV_Poisson)*

- IntVector poissonUncVars

  initial values of the poisson uncertain variables (from the `initial_point` *specification in VarDAUV_Poisson)*

- BitArray poissonUncCat

  is each poisson var strictly categorical (true) or relaxable (false)

- RealVector binomialUncProbPerTrial

  probabilities per each trial (p) for the binomial uncertain variables from the `prob_per_trial` *specification in VarDAUV_Binomial)*

- IntVector binomialUncNumTrials

  Number of trials (N) for the binomial uncertain variables from the `num_trials` *specification in VarDAUV_Binomial)*

- IntVector binomialUncVars

  initial values of the binomial uncertain variables (from the `initial_point` *specification in VarDAUV_Binomial)*

- BitArray binomialUncCat

  is each binomial var strictly categorical (true) or relaxable (false)

- RealVector negBinomialUncProbPerTrial

  probabilities per each trial (p) for the negative binomial uncertain variables from the `prob_per_trial` *specification in VarDAUV_Negative_Binomial)*

- IntVector negBinomialUncNumTrials

  Number of trials (N) for the negative binomial uncertain variables from the `num_trials` *specification in VarDAUV_Negative_Binomial)*

- IntVector negBinomialUncVars

  initial values of the negative binomial uncertain variables (from the `initial_point` *specification in VarDAUV_Negative_Binomial)*

- BitArray negBinomialUncCat

  is each negbinomial var strictly categorical (true) or relaxable (false)

- RealVector geometricUncProbPerTrial

  probabilities per each trial (p) for the geometric uncertain variables from the `prob_per_trial` *specification in VarDAUV_Geometric)*

- IntVector geometricUncVars

  initial values of the geometric uncertain variables (from the `initial_point` *specification in VarDAUV_Geometric)*

- BitArray geometricUncCat

  is each geometric var strictly categorical (true) or relaxable (false)

- IntVector hyperGeomUncTotalPop

  Size of total populations (N) for the hypergeometric uncertain variables from the `total_population` *specification in VarDAUV_Hypergeometric)*

- IntVector hyperGeomUncSelectedPop

  Size of selected populations for the hypergeometric uncertain variables from the `selected_population` *specification in VarDAUV_Hypergeometric)*

- IntVector hyperGeomUncNumDrawn

Number failed in the selected populations for the hypergeometric variables from the `num_drawn` *specification in VarDAUV_Hypergeometric)*

- IntVector hyperGeomUncVars

  initial values of the hypergeometric uncertain variables (from the `initial_point` *specification in VarDAUV_-Hypergeometric)*

- BitArray hyperGeomUncCat

  is each hypergeom var strictly categorical (true) or relaxable (false)

- IntRealMapArray histogramUncPointIntPairs

  An array for each integer-valued point-based histogram uncertain variable. Each array entry is a map from an integer value to its probability. (See discrete histogram in LHS manual; from the `histogram_point_uncertain` *specification in VarDAUV_Point_Histogram)*

- IntVector histogramPointIntUncVars

  initial values of the real-valued histogram point uncertain variables (from the `initial_point` *specification in VarDAUV_Point_Histogram)*

- BitArray histogramUncPointIntCat

  is each hupi var strictly categorical (true) or relaxable (false)

- StringRealMapArray histogramUncPointStrPairs

  An array for each string-valued point-based histogram uncertain variable. Each array entry is a map from a string value to its probability. (See discrete histogram in LHS manual; from the `histogram_point_uncertain` *specification in VarDAUV_Point_Histogram)*

- StringArray histogramPointStrUncVars

  initial values of the real-valued histogram point uncertain variables (from the `initial_point` *specification in VarDAUV_Point_Histogram)*

- RealRealMapArray histogramUncPointRealPairs

  An array for each real-valued point-based histogram uncertain variable. Each array entry is a map from a real value to its probability. (See discrete histogram in LHS manual; from the `histogram_point_uncertain` *specification in VarDAUV_Point_Histogram)*

- RealVector histogramPointRealUncVars

  initial values of the real-valued histogram point uncertain variables (from the `initial_point` *specification in VarDAUV_Point_Histogram)*

- BitArray histogramUncPointRealCat

  is each hupr var strictly categorical (true) or relaxable (false)

- RealSymMatrix uncertainCorrelations

  correlation matrix for all uncertain variables (from the `uncertain_correlation_matrix` *specification in VarAUV_Correlations). This matrix specifies rank correlations for LHS sampling and correlation coefficients (rho-_ij = normalized covariance matrix) for other methods.*

- RealRealPairRealMapArray continuousIntervalUncBasicProbs

  Probability values per interval cell per epistemic interval uncertain variable (from the `continuous_interval-_uncertain` `interval_probs` *specification in VarCEUV_Interval)*

- RealVector continuousIntervalUncVars

  initial values of the continuous interval uncertain variables (from the `initial_point` *specification in VarCEU-V_Interval)*

- IntIntPairRealMapArray discreteIntervalUncBasicProbs

  Probability values per interval cell per epistemic interval uncertain variable (from the `discrete_interval_-uncertain` `interval_probs` *specification in VarDIUV)*

- IntVector discreteIntervalUncVars

initial values of the discrete interval uncertain variables (from the `initial_point` specification in VarDIUV)

- BitArray discreteIntervalUncCat

  is each diu var strictly categorical (true) or relaxable (false)

- IntRealMapArray discreteUncSetIntValuesProbs

  complete set of admissible values with associated basic probability assignments for each of the discrete epistemic uncertain variables defined by an integer set (from the `discrete_uncertain_set` integer `set_values` specification in VarDUSIV)

- IntVector discreteUncSetIntVars

  initial values of the discrete uncertain set integer variables (from the `initial_point` specification in VarDUSIV)

- BitArray discreteUncSetIntCat

  is each dusi var strictly categorical (true) or relaxable (false)

- StringRealMapArray discreteUncSetStrValuesProbs

  complete set of admissible values with associated basic probability assignments for each of the discrete epistemic uncertain variables defined by a string set (from the `discrete_uncertain_set` string `set_values` specification in VarDUSIV)

- StringArray discreteUncSetStrVars

  initial values of the discrete uncertain set integer variables (from the `initial_point` specification in VarDUSIV)

- RealRealMapArray discreteUncSetRealValuesProbs

  complete set of admissible values with associated basic probability assignments for each of the discrete epistemic uncertain variables defined by a real set (from the `discrete_uncertain_set_eal` `set_values` specification in VarDUSRV)

- RealVector discreteUncSetRealVars

  initial values of the discrete uncertain set real variables (from the `initial_point` specification in VarDUSRV)

- BitArray discreteUncSetRealCat

  is each dusr var strictly categorical (true) or relaxable (false)

- RealVector continuousStateVars

  initial values for the continuous state variables array (from the `continuous_state` `initial_point` specification in VarSV)

- RealVector continuousStateLowerBnds

  lower bounds array for the continuous state variables (from the `continuous_state` `lower_bounds` specification in VarSV)

- RealVector continuousStateUpperBnds

  upper bounds array for the continuous state variables (from the `continuous_state` `upper_bounds` specification in VarSV)

- IntVector discreteStateRangeVars

  initial values for the discrete state variables defined by an integer range (from the `discrete_state_range` `initial_point` specification in VarSV)

- IntVector discreteStateRangeLowerBnds

  lower bounds array for the discrete state variables defined by an integer range (from the `discrete_state_range` `lower_bounds` specification in VarSV)

- IntVector discreteStateRangeUpperBnds

  upper bounds array for the discrete state variables defined by an integer range(from the `discrete_state_range` `upper_bounds` specification in VarSV)

- BitArray discreteStateRangeCat

  is each dsr var strictly categorical (true) or relaxable (false)

- IntVector discreteStateSetIntVars

  initial values for the discrete state variables defined by an integer set (from the `discrete_state_set` *integer* `initial_point` *specification in VarSV)*
- StringArray discreteStateSetStrVars

  initial values for the discrete state variables defined by a string set (from the `discrete_state_set` *string* `initial_point` *specification in VarSV)*
- RealVector discreteStateSetRealVars

  initial values for the discrete state variables defined by a real set (from the `discrete_state_set` *real* `initial_point` *specification in VarSV)*
- IntSetArray discreteStateSetInt

  complete set of admissible values for each of the discrete state variables defined by an integer set (from the `discrete_state_set` *integer* `set_values` *specification in VarSV)*
- StringSetArray discreteStateSetStr

  complete set of admissible values for each of the discrete state variables defined by a string set (from the `discrete_state_set` *string* `set_values` *specification in VarSV)*
- RealSetArray discreteStateSetReal

  complete set of admissible values for each of the discrete state variables defined by a real set (from the `discrete_state_set` *real* `set_values` *specification in VarSV)*
- BitArray discreteStateSetIntCat

  is each dssi var strictly categorical (true) or relaxable (false)
- BitArray discreteStateSetRealCat

  is each dssr var strictly categorical (true) or relaxable (false)
- StringArray continuousStateLabels

  labels array for the continuous state variables (from the `continuous_state descriptors` *specification in VarSV)*
- StringArray discreteStateRangeLabels

  labels array for the discrete state variables defined by an integer range (from the `discrete_state_range descriptors` *specification in VarSV)*
- StringArray discreteStateSetIntLabels

  labels array for the discrete state variables defined by an integer set (from the `discrete_state_set descriptors` *specification in VarSV)*
- StringArray discreteStateSetStrLabels

  labels array for the discrete state variables defined by a string set (from the `discrete_state_set descriptors` *specification in VarSV)*
- StringArray discreteStateSetRealLabels

  labels array for the discrete state variables defined by a real set (from the `discrete_state_set descriptors` *specification in VarSV)*
- IntVector discreteDesignSetIntLowerBnds

  discrete design integer set lower bounds inferred from set values
- IntVector discreteDesignSetIntUpperBnds

  discrete design integer set upper bounds inferred from set values
- StringArray discreteDesignSetStrLowerBnds

  discrete design string set lower bounds inferred from set values
- StringArray discreteDesignSetStrUpperBnds

  discrete design stinr set upper bounds inferred from set values

- RealVector discreteDesignSetRealLowerBnds

  discrete design real set lower bounds inferred from set values
- RealVector discreteDesignSetRealUpperBnds

  discrete design real set upper bounds inferred from set values
- RealVector continuousAleatoryUncVars

  array of values for all continuous aleatory uncertain variables
- RealVector continuousAleatoryUncLowerBnds

  distribution lower bounds for all continuous aleatory uncertain variables (collected from `nuv_lower_bounds`*,* `lnuv_lower_bounds`*,* `uuv_lower_bounds`*,* `luuv_lower_bounds`*,* `tuv_lower_bounds`*, and* `buv_lower_bounds` *specifications in VarAUV, and derived for gamma, gumbel, frechet, weibull and histogram bin specifications)*
- RealVector continuousAleatoryUncUpperBnds

  distribution upper bounds for all continuous aleatory uncertain variables (collected from `nuv_upper_bounds`*,* `lnuv_upper_bounds`*,* `uuv_upper_bounds`*,* `luuv_upper_bounds`*,* `tuv_lower_bounds`*, and* `buv_upper_bounds` *specifications in VarAUV, and derived for gamma, gumbel, frechet, weibull and histogram bin specifications)*
- StringArray continuousAleatoryUncLabels

  labels for all continuous aleatory uncertain variables (collected from `nuv_descriptors`*,* `lnuv_descriptors`*,* `uuv_descriptors`*,* `luuv_descriptors`*,* `tuv_descriptors`*,* `buv_descriptors`*,* `gauv_descriptors`*,* `guuv_descriptors`*,* `fuv_descriptors`*,* `wuv_descriptors`*, and* `hbuv_descriptors` *specifications in VarAUV)*
- IntVector discreteIntAleatoryUncVars

  array of values for all discrete integer aleatory uncertain variables
- IntVector discreteIntAleatoryUncLowerBnds

  distribution lower bounds for all discrete integer aleatory uncertain variables
- IntVector discreteIntAleatoryUncUpperBnds

  distribution upper bounds for all discrete integer aleatory uncertain variables
- StringArray discreteIntAleatoryUncLabels

  labels for all discrete integer aleatory uncertain variables
- StringArray discreteStrAleatoryUncVars

  array of values for all discrete string epistemic uncertain variables
- StringArray discreteStrAleatoryUncLowerBnds

  distribution lower bounds for all discrete string epistemic uncertain variables
- StringArray discreteStrAleatoryUncUpperBnds

  distribution upper bounds for all discrete string epistemic uncertain variables
- StringArray discreteStrAleatoryUncLabels

  labels for all discrete string epistemic uncertain variables
- RealVector discreteRealAleatoryUncVars

  array of values for all discrete real aleatory uncertain variables
- RealVector discreteRealAleatoryUncLowerBnds

  distribution lower bounds for all discrete real aleatory uncertain variables
- RealVector discreteRealAleatoryUncUpperBnds

  distribution upper bounds for all discrete real aleatory uncertain variables
- StringArray discreteRealAleatoryUncLabels

  labels for all discrete real aleatory uncertain variables

- RealVector continuousEpistemicUncVars

  array of values for all continuous epistemic uncertain variables
- RealVector continuousEpistemicUncLowerBnds

  distribution lower bounds for all continuous epistemic uncertain variables
- RealVector continuousEpistemicUncUpperBnds

  distribution upper bounds for all continuous epistemic uncertain variables
- StringArray continuousEpistemicUncLabels

  labels for all continuous epistemic uncertain variables
- IntVector discreteIntEpistemicUncVars

  array of values for all discrete integer epistemic uncertain variables
- IntVector discreteIntEpistemicUncLowerBnds

  distribution lower bounds for all discrete integer epistemic uncertain variables
- IntVector discreteIntEpistemicUncUpperBnds

  distribution upper bounds for all discrete integer epistemic uncertain variables
- StringArray discreteIntEpistemicUncLabels

  labels for all discrete integer epistemic uncertain variables
- StringArray discreteStrEpistemicUncVars

  array of values for all discrete string epistemic uncertain variables
- StringArray discreteStrEpistemicUncLowerBnds

  distribution lower bounds for all discrete string epistemic uncertain variables
- StringArray discreteStrEpistemicUncUpperBnds

  distribution upper bounds for all discrete string epistemic uncertain variables
- StringArray discreteStrEpistemicUncLabels

  labels for all discrete string epistemic uncertain variables
- RealVector discreteRealEpistemicUncVars

  array of values for all discrete real epistemic uncertain variables
- RealVector discreteRealEpistemicUncLowerBnds

  distribution lower bounds for all discrete real epistemic uncertain variables
- RealVector discreteRealEpistemicUncUpperBnds

  distribution upper bounds for all discrete real epistemic uncertain variables
- StringArray discreteRealEpistemicUncLabels

  labels for all discrete real epistemic uncertain variables
- IntVector discreteStateSetIntLowerBnds

  discrete state integer set lower bounds inferred from set values
- IntVector discreteStateSetIntUpperBnds

  discrete state integer set upper bounds inferred from set values
- StringArray discreteStateSetStrLowerBnds

  discrete state string set lower bounds inferred from set values
- StringArray discreteStateSetStrUpperBnds

  discrete state string set upper bounds inferred from set values
- RealVector discreteStateSetRealLowerBnds

  discrete state real set lower bounds inferred from set values
- RealVector discreteStateSetRealUpperBnds

  discrete state real set upper bounds inferred from set values

Private Member Functions

- DataVariablesRep ()

  default constructor
- ~DataVariablesRep ()

  destructor
- void write (std::ostream &s) const

  write a DataVariablesRep object to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a DataVariablesRep object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a DataVariablesRep object to a packed MPI buffer

Private Attributes

- int referenceCount

  number of handle objects sharing dataVarsRep

Friends

- class DataVariables

  the handle class can access attributes of the body class directly

13.30.1 Detailed Description

Body class for variables specification data.

The DataVariablesRep class is used to contain the data from a variables keyword specification. Default values are managed in the DataVariablesRep constructor. Data is public to avoid maintaining set/get functions, but is still encapsulated within ProblemDescDB since ProblemDescDB::dataVariablesList is private.

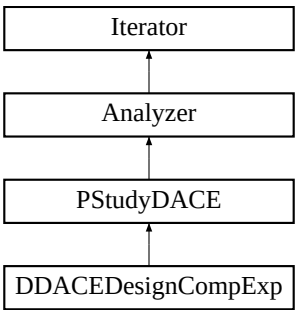
The documentation for this class was generated from the following files:

- DataVariables.hpp
- DataVariables.cpp

13.31 DDACEDesignCompExp Class Reference

Wrapper class for the DDACE design of experiments library.

Inheritance diagram for DDACEDesignCompExp:



Public Member Functions

- DDACEDesignCompExp (ProblemDescDB &problem_db, Model &model)

  primary constructor for building a standard DACE iterator

- DDACEDesignCompExp (Model &model, int samples, int symbols, int seed, unsigned short sampling_method)

  alternate constructor used for building approximations

- ~DDACEDesignCompExp ()

  destructor

- void pre_run ()

  pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

- void extract_trends ()

  Mapping of the core_run() virtual function for the PStudy/DACE branch.

- void post_input ()

  read tabular data for post-run mode

- void post_run (std::ostream &s)

  post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/Responses and perform final analysis phase in a standalone way

- int num_samples () const
- void sampling_reset (int min_samples, bool all_data_flag, bool stats_flag)

  reset sampling iterator to use at least min_samples

- unsigned short sampling_scheme () const

  return sampling name

- void vary_pattern (bool pattern_flag)

  sets varyPattern in derived classes that support it

- void get_parameter_sets (Model &model)

  Returns one block of samples (ndim ∗ num_samples)

Private Member Functions

- void compute_main_effects ()

  builds a DDaceMainEffects::OneWayANOVA if mainEffectsFlag is set

- void resolve_samples_symbols ()

  convenience function for resolving number of samples and number of symbols from input.

Static Private Member Functions

- static void copy_data (const std::vector< DDaceSamplePoint > &dspa, Real ∗ptr, const int ptr_len)

  copy DDACE point to RealVector

Private Attributes

- unsigned short daceMethod

  oas, lhs, oa_lhs, random, box_behnken, central_composite, or grid
- int samplesSpec

  initial specification of number of samples
- int symbolsSpec

  initial specification of number of symbols
- int numSamples

  current number of samples to be evaluated
- int numSymbols

  current number of symbols to be used in generating the sample set (inversely related to number of replications)
- const int seedSpec

  the user seed specification for the random number generator (allows repeatable results)
- int randomSeed

  current seed for the random number generator
- bool allDataFlag

  flag which triggers the update of allVars/allResponses for use by Iterator::all_variables() and Iterator::all_responses()
- size_t numDACERuns

  counter for number of executions for this object
- bool varyPattern

  flag for continuing the random number sequence from a previous execution (e.g., for surrogate-based optimization) so that multiple executions are repeatable but not correlated.
- bool mainEffectsFlag

  flag which specifies main effects
- std::vector< std::vector< int > > symbolMapping

  mapping of symbols for main effects calculations

Additional Inherited Members

13.31.1 Detailed Description

Wrapper class for the DDACE design of experiments library.

The DDACEDesignCompExp class provides a wrapper for DDACE, a C++ design of experiments library from the Computational Sciences and Mathematics Research (CSMR) department at Sandia's Livermore CA site. This class uses design and analysis of computer experiments (DACE) methods to sample the design space spanned by the bounds of a Model. It returns all generated samples and their corresponding responses as well as the best sample found.

13.31.2 Constructor & Destructor Documentation

DDACEDesignCompExp (ProblemDescDB & *problem_db*, Model & *model*)

primary constructor for building a standard DACE iterator

This constructor is called for a standard iterator built with data from probDescDB.

References Dakota::abort_handler(), DDACEDesignCompExp::daceMethod, DDACEDesignCompExp::mainEffectsFlag, Iterator::maxEvalConcurrency, Analyzer::numContinuousVars, and DDACEDesignCompExp::numSamples.

DDACEDesignCompExp (Model & *model,* int *samples,* int *symbols,* int *seed,* unsigned short *sampling_method*)

alternate constructor used for building approximations

This alternate constructor is used for instantiations on-the-fly, using only the incoming data. No problem description database queries are used.

References Iterator::maxEvalConcurrency, DDACEDesignCompExp::numSamples, and DDACEDesignCompExp::resolve_samples_symbols().

13.31.3 Member Function Documentation

void pre_run ( ) `[virtual]`

pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

pre-run phase, which a derived iterator may optionally reimplement; when not present, pre-run is likely integrated into the derived run function. This is a virtual function; when re-implementing, a derived class must call its nearest parent's pre_run(), if implemented, typically *before* performing its own implementation steps.

Reimplemented from Iterator.

References DDACEDesignCompExp::get_parameter_sets(), Iterator::iteratedModel, and PStudyDACE::varBasedDecompFlag.

void post_run (std::ostream & *s*) `[virtual]`

post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/Responses and perform final analysis phase in a standalone way

Post-run phase, which a derived iterator may optionally reimplement; when not present, post-run is likely integrated into run. This is a virtual function; when re-implementing, a derived class must call its nearest parent's post_run(), typically *after* performing its own implementation steps.

Reimplemented from Analyzer.

References Analyzer::allResponses, Analyzer::allSamples, SensAnalysisGlobal::compute_correlations(), DDACEDesignCompExp::compute_main_effects(), DDACEDesignCompExp::mainEffectsFlag, Analyzer::post_run(), PStudyDACE::pStudyDACESensGlobal, Iterator::subIteratorFlag, and PStudyDACE::varBasedDecompFlag.

int num_samples ( ) const `[inline], [virtual]`

Return current number of evaluation points. Since the calculation of samples, collocation points, etc. might be costly, provide a default implementation here that backs out from the maxEvalConcurrency.

Reimplemented from Analyzer.

References DDACEDesignCompExp::numSamples.

void resolve_samples_symbols ( ) `[private]`

convenience function for resolving number of samples and number of symbols from input.

This function must define a combination of samples and symbols that is acceptable for a particular sampling algorithm. Users provide requests for these quantities, but this function must enforce any restrictions imposed by the sampling algorithms.

References Dakota::abort_handler(), DDACEDesignCompExp::daceMethod, Analyzer::numContinuousVars, DDACEDesignCompExp::numSamples, and DDACEDesignCompExp::numSymbols.

Referenced by DDACEDesignCompExp::DDACEDesignCompExp(), and DDACEDesignCompExp::get_parameter_sets().

void copy_data (const std::vector< DDaceSamplePoint > & *dspa*, Real * *ptr*, const int *ptr_len*) [static], [private]

copy DDACE point to RealVector

copy DDACE point array to RealVectorArray copy DDACE point array to Real*

References Dakota::abort_handler().

Referenced by DDACEDesignCompExp::get_parameter_sets().

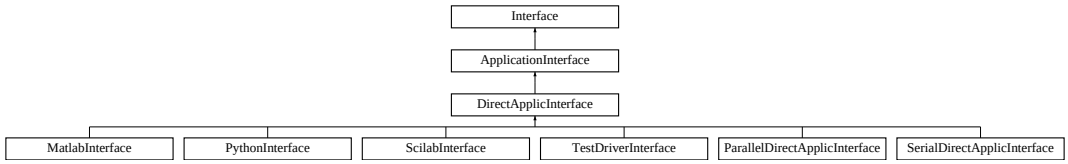
The documentation for this class was generated from the following files:

- DDACEDesignCompExp.hpp
- DDACEDesignCompExp.cpp

13.32 DirectApplicInterface Class Reference

Derived application interface class which spawns simulation codes and testers using direct procedure calls.

Inheritance diagram for DirectApplicInterface:



Public Member Functions

- DirectApplicInterface (const ProblemDescDB &problem_db)

  constructor
- ~DirectApplicInterface ()

  destructor
- void derived_map (const Variables &vars, const ActiveSet &set, Response &response, int fn_eval_id)

  Called by map() and other functions to execute the simulation in synchronous mode. The portion of performing an evaluation that is specific to a derived class.
- void derived_map_asynch (const ParamResponsePair &pair)

  Called by map() and other functions to execute the simulation in asynchronous mode. The portion of performing an asynchronous evaluation that is specific to a derived class.
- void wait_local_evaluations (PRPQueue &prp_queue)

  For asynchronous function evaluations, this method is used to detect completion of jobs and process their results. It provides the processing code that is specific to derived classes. This version waits for at least one completion.
- void test_local_evaluations (PRPQueue &prp_queue)

  For asynchronous function evaluations, this method is used to detect completion of jobs and process their results. It provides the processing code that is specific to derived classes. This version is nonblocking and will return without any completions if none are immediately available.
- int synchronous_local_analysis (int analysis_id)
- const StringArray & analysis_drivers () const

  retrieve the analysis drivers specification for application interfaces
- void init_communicators_checks (int max_eval_concurrency)
- void set_communicators_checks (int max_eval_concurrency)

Protected Member Functions

- virtual int derived_map_if (const Dakota::String &if_name)

  execute the input filter portion of a direct evaluation invocation
- virtual int derived_map_ac (const Dakota::String &ac_name)

  execute an analysis code portion of a direct evaluation invocation
- virtual int derived_map_of (const Dakota::String &of_name)

  execute the output filter portion of a direct evaluation invocation
- virtual void set_local_data (const Variables &vars, const ActiveSet &set)

  convenience function for local test simulators which sets per-evaluation variable and active set attributes; derived classes reimplementing this likely need to invoke the base class API
- virtual void set_local_data (const Response &response)

  convenience function for local test simulators which sets per-evaluation response attributes; derived classes reimplementing this likely need to invoke the base class API
- virtual void set_local_data (const Variables &vars, const ActiveSet &set, const Response &response)

  convenience function for local test simulators which sets per-evaluation variable, active set, and response attributes; derived classes reimplementing this likely need to invoke the base class API
- void overlay_response (Response &response)

  convenience function for local test simulators which overlays response contributions from multiple analyses using MPI_Reduce

Protected Attributes

- String iFilterName

  name of the direct function input filter
- String oFilterName

  name of the direct function output filter
- driver_t iFilterType

  enum type of the direct function input filter
- driver_t oFilterType

  enum type of the direct function output filter
- bool gradFlag

  signals use of fnGrads in direct simulator functions
- bool hessFlag

  signals use of fnHessians in direct simulator functions
- size_t numFns

  number of functions in fnVals
- size_t numVars

  total number of continuous and discrete variables
- size_t numACV

  total number of continuous variables
- size_t numADIV

  total number of discete integer variables
- size_t numADRV

  total number of discete real variables

- size_t numDerivVars

  number of active derivative variables
- unsigned short localDataView

  bit-wise record of which local data views are active; see enum local_data_t
- RealVector xC

  continuous variables used within direct simulator fns
- IntVector xDI

  discrete int variables used within direct simulator fns
- RealVector xDR

  discrete real variables used within direct simulator fns
- StringMultiArray xCLabels

  continuous variable labels
- StringMultiArray xDILabels

  discrete integer variable labels
- StringMultiArray xDRLabels

  discrete real variable labels
- std::map< String, var_t > varTypeMap

  map from variable label to enum
- std::map< String, driver_t > driverTypeMap

  map from driver name to enum
- std::map< var_t, Real > xCM

  map from var_t enum to continuous value
- std::map< var_t, int > xDIM

  map from var_t enum to discrete int value
- std::map< var_t, Real > xDRM

  map from var_t enum to discrete real value
- std::vector< var_t > varTypeDVV

  var_t enumerations corresponding to DVV components
- std::vector< var_t > xCMLabels

  var_t enumerations corresponding to continuous variable labels
- std::vector< var_t > xDIMLabels

  var_t enumerations corresponding to discrete integer variable labels
- std::vector< var_t > xDRMLabels

  var_t enumerations corresponding to discrete real variable labels
- ShortArray directFnASV

  class scope active set vector
- SizetArray directFnDVV

  class scope derivative variables vector
- RealVector fnVals

  response fn values within direct simulator fns
- RealMatrix fnGrads

  response fn gradients w/i direct simulator fns
- RealSymMatrixArray fnHessians

response fn Hessians within direct fns

- StringArray analysisDrivers

  the set of analyses within each function evaluation (from the analysis_drivers interface specification)

- std::vector< driver_t > analysisDriverTypes

  conversion of analysisDrivers to driver_t

- size_t analysisDriverIndex

  the index of the active analysis driver within analysisDrivers

- String2DArray analysisComponents

  the set of optional analysis components used by the analysis drivers (from the analysis_components interface specification)

13.32.1 Detailed Description

Derived application interface class which spawns simulation codes and testers using direct procedure calls.

DirectApplicInterface uses a few linkable simulation codes and several internal member functions to perform parameter to response mappings.

13.32.2 Member Function Documentation

int synchronous_local_analysis (int *analysis_id*) [inline], [virtual]

This code provides the derived function used by ApplicationInterface::serve_analyses_synch().

Reimplemented from ApplicationInterface.

References DirectApplicInterface::analysisDriverIndex, DirectApplicInterface::analysisDrivers, and DirectApplicInterface::derived_map_ac().

void init_communicators_checks (int *max_eval_concurrency*) [inline], [virtual]

Process init issues as warnings since some contexts (e.g., HierarchSurrModel) initialize more configurations than will be used and DirectApplicInterface allows override by derived plug-ins.

Reimplemented from ApplicationInterface.

References ApplicationInterface::check_asynchronous(), and ApplicationInterface::check_multiprocessor_asynchronous().

void set_communicators_checks (int *max_eval_concurrency*) [inline], [virtual]

Process run-time issues as hard errors.

Reimplemented from ApplicationInterface.

Reimplemented in SerialDirectApplicInterface, and ParallelDirectApplicInterface.

References Dakota::abort_handler(), ApplicationInterface::check_asynchronous(), and ApplicationInterface::check_multiprocessor_asynchronous().

int derived_map_ac (const Dakota::String & *ac_name*) [protected], [virtual]

execute an analysis code portion of a direct evaluation invocation

When a direct analysis/filter is a member function, the (vars,set,response) data does not need to be passed through the API. If, however, non-member analysis/filter functions are added, then pass (vars,set,response) through to the non-member fns:

```
// API declaration
int sim(const Variables& vars, const ActiveSet& set, Response& response);
// use of API within derived_map_ac()
if (ac_name == "sim")
  fail_code = sim(directFnVars, directFnActSet, directFnResponse);
```

Reimplemented in SerialDirectApplicInterface, ParallelDirectApplicInterface, MatlabInterface, PythonInterface, TestDriverInterface, and ScilabInterface.

References Dakota::abort_handler(), and ApplicationInterface::analysisServerId.

Referenced by DirectApplicInterface::derived_map(), and DirectApplicInterface::synchronous_local_analysis().

The documentation for this class was generated from the following files:

- DirectApplicInterface.hpp
- DirectApplicInterface.cpp

13.33 DiscrepancyCorrection Class Reference

Base class for discrepancy corrections.

Public Member Functions

- DiscrepancyCorrection ()

  default constructor
- DiscrepancyCorrection (Model &surr_model, const IntSet &surr_fn_indices, short corr_type, short corr_order)

  standard constructor
- DiscrepancyCorrection (const IntSet &surr_fn_indices, size_t num_fns, size_t num_vars, short corr_type, short corr_order)

  alternate constructor
- ~DiscrepancyCorrection ()

  destructor
- void initialize (Model &surr_model, const IntSet &surr_fn_indices, short corr_type, short corr_order)

  initialize the DiscrepancyCorrection data
- void initialize (const IntSet &surr_fn_indices, size_t num_fns, size_t num_vars, short corr_type, short corr_order)

  initialize the DiscrepancyCorrection data
- void compute (const Variables &vars, const Response &truth_response, const Response &approx_response, bool quiet_flag=false)

  compute the correction required to bring approx_response into agreement with truth_response and store in {add,mult}Corrections
- void compute (const Response &truth_response, const Response &approx_response, Response &discrepancy_response, bool quiet_flag=false)

  compute the correction required to bring approx_response into agreement with truth_response and store in discrepancy_response
- void apply (const Variables &vars, Response &approx_response, bool quiet_flag=false)

  apply the correction computed in compute() to approx_response
- bool active () const

  indicates an active correction via non-empty correctionType
- short correction_type () const

return correctionType

- short correction_order () const

  return correctionOrder

- short data_order () const

  return dataOrder

- bool computed () const

  return correctionComputed

Protected Attributes

- IntSet surrogateFnIndices

  for mixed response sets, this array specifies the response function subset that is approximated

- short correctionType

  approximation correction approach to be used: NO_CORRECTION, ADDITIVE_CORRECTION, MULTIPLICATIVE_CORRECTION, or COMBINED_CORRECTION.

- short correctionOrder

  approximation correction order to be used: 0, 1, or 2

- short dataOrder

  order of correction data in 3-bit format: overlay of 1 (value), 2 (gradient), and 4 (Hessian)

- bool correctionComputed

  flag indicating whether or not a correction has been computed and is available for application

- size_t numFns

  total number of response functions (of which surrogateFnIndices may define a subset)

- size_t numVars

  number of continuous variables active in the correction

Private Member Functions

- void initialize_corrections ()

  internal convenience function shared by overloaded initialize() variants

- bool check_scaling (const RealVector &truth_fns, const RealVector &approx_fns)

  define badScalingFlag

- void compute_additive (const Response &truth_response, const Response &approx_response, int index, Real &discrep_fn, RealVector &discrep_grad, RealSymMatrix &discrep_hess)

  internal convenience function for computing additive corrections between truth and approximate responses

- void compute_multiplicative (const Response &truth_response, const Response &approx_response, int index, Real &discrep_fn, RealVector &discrep_grad, RealSymMatrix &discrep_hess)

  internal convenience function for computing multiplicative corrections between truth and approximate responses

- void apply_additive (const Variables &vars, Response &approx_response)

  internal convenience function for applying additive corrections to an approximate response

- void apply_multiplicative (const Variables &vars, Response &approx_response)

  internal convenience function for applying multiplicative corrections to an approximate response

- void apply_additive (const Variables &vars, RealVector &approx_fns)

  internal convenience function for applying additive corrections to a set of response functions

- void apply_multiplicative (const Variables &vars, RealVector &approx_fns)

  internal convenience function for applying multiplicative corrections to a set of response functions
- const Response & search_db (const Variables &search_vars, const ShortArray &search_asv)

  search data_pairs for missing approximation data

Private Attributes

- bool badScalingFlag

  flag used to indicate function values near zero for multiplicative corrections; triggers an automatic switch to additive corrections
- bool computeAdditive

  flag indicating the need for additive correction calculations
- bool computeMultiplicative

  flag indicating the need for multiplicative correction calculations
- SharedApproxData sharedData

  data that is shared among all correction Approximations
- std::vector< Approximation > addCorrections

  array of additive corrections; surrogate models of a model discrepancy function (formed from model differences)
- std::vector< Approximation > multCorrections

  array of multiplicative corrections; surrogate models of a model discrepancy function (formed from model ratios)
- Model surrModel

  shallow copy of the surrogate model instance as returned by Model::surrogate_model() (the DataFitSurrModel or HierarchSurrModel::lowFidelityModel instance)
- RealVector combineFactors

  factors for combining additive and multiplicative corrections. Each factor is the weighting applied to the additive correction and 1.-factor is the weighting applied to the multiplicative correction. The factor value is determined by an additional requirement to match the high fidelity function value at the previous correction point (e.g., previous trust region center). This results in a multipoint correction instead of a strictly local correction.
- Variables correctionPrevCenterPt

  copy of center point from the previous correction cycle
- RealVector truthFnsCenter

  truth function values at the current correction point
- RealVector approxFnsCenter

  Surrogate function values at the current correction point.
- RealMatrix approxGradsCenter

  Surrogate gradient values at the current correction point.
- RealVector truthFnsPrevCenter

  copy of truth function values at center of previous correction cycle
- RealVector approxFnsPrevCenter

  copy of approximate function values at center of previous correction cycle

13.33.1 Detailed Description

Base class for discrepancy corrections.

The DiscrepancyCorrection class provides common functions for computing and applying corrections to approximations.

13.33.2 Member Function Documentation

void compute (const Variables & *vars,* const Response & *truth_response,* const Response & *approx_response,* bool *quiet_flag* = `false`)

compute the correction required to bring approx_response into agreement with truth_response and store in {add,mult}Corrections

Compute an additive or multiplicative correction that corrects the approx_response to have 0th-order consistency (matches values), 1st-order consistency (matches values and gradients), or 2nd-order consistency (matches values, gradients, and Hessians) with the truth_response at a single point (e.g., the center of a trust region). The 0th-order, 1st-order, and 2nd-order corrections use scalar values, linear scaling functions, and quadratic scaling functions, respectively, for each response function.

References Response::active_set(), DiscrepancyCorrection::addCorrections, DiscrepancyCorrection::apply(), DiscrepancyCorrection::apply_additive(), DiscrepancyCorrection::apply_multiplicative(), DiscrepancyCorrection::approxFnsCenter, DiscrepancyCorrection::approxFnsPrevCenter, DiscrepancyCorrection::approxGradsCenter, DiscrepancyCorrection::badScalingFlag, DiscrepancyCorrection::check_scaling(), DiscrepancyCorrection::combineFactors, DiscrepancyCorrection::compute_additive(), DiscrepancyCorrection::compute_multiplicative(), DiscrepancyCorrection::computeAdditive, DiscrepancyCorrection::computeMultiplicative, Variables::continuous_variables(), Response::copy(), DiscrepancyCorrection::correctionComputed, DiscrepancyCorrection::correctionOrder, DiscrepancyCorrection::correctionPrevCenterPt, DiscrepancyCorrection::correctionType, DiscrepancyCorrection::dataOrder, Variables::discrete_int_variables(), Variables::discrete_real_variables(), Response::function_gradients(), Response::function_values(), Model::is_null(), DiscrepancyCorrection::multCorrections, DiscrepancyCorrection::numFns, DiscrepancyCorrection::numVars, ActiveSet::request_values(), DiscrepancyCorrection::sharedData, DiscrepancyCorrection::surrModel, DiscrepancyCorrection::surrogateFnIndices, DiscrepancyCorrection::truthFnsCenter, and DiscrepancyCorrection::truthFnsPrevCenter.

Referenced by HierarchSurrModel::derived_asynch_compute_response(), HierarchSurrModel::derived_compute_response(), DataFitSurrModel::derived_compute_response(), HierarchSurrModel::derived_synchronize(), DataFitSurrModel::derived_synchronize(), HierarchSurrModel::derived_synchronize_nowait(), DataFitSurrModel::derived_synchronize_nowait(), and SurrBasedLocalMinimizer::minimize_surrogates().

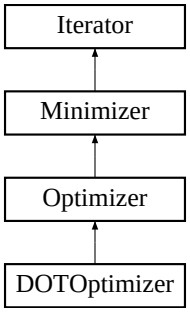
The documentation for this class was generated from the following files:

- DiscrepancyCorrection.hpp
- DiscrepancyCorrection.cpp

13.34 DOTOptimizer Class Reference

Wrapper class for the DOT optimization library.

Inheritance diagram for DOTOptimizer:



Public Member Functions

- DOTOptimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor
- DOTOptimizer (const String &method_string, Model &model)

  alternate constructor; construct without ProblemDescDB
- ~DOTOptimizer ()

  destructor
- void find_optimum ()

  Used within the optimizer branch for computing the optimal solution. Redefines the run virtual function for the optimizer branch.

Protected Member Functions

- void initialize_run ()

  performs run-time set up

Private Member Functions

- void initialize ()

  Shared constructor code.
- void allocate_workspace ()

  Allocates workspace for the optimizer.
- void allocate_constraints ()

  Allocates constraint mappings.

Private Attributes

- int dotInfo

  INFO from DOT manual.
- int dotFDSinfo

  internal DOT parameter NGOTOZ
- int dotMethod

  METHOD from DOT manual.
- int printControl

  IPRINT from DOT manual (controls output verbosity)
- RealArray realCntlParmArray

  RPRM from DOT manual.
- IntArray intCntlParmArray

  IPRM from DOT manual.
- RealVector designVars

  array of design variable values passed to DOT
- Real objFnValue

  value of the objective function passed to DOT
- RealVector constraintValues

array of nonlinear constraint values passed to DOT

- int realWorkSpaceSize

  size of realWorkSpace

- int intWorkSpaceSize

  size of intWorkSpace

- RealArray realWorkSpace

  real work space for DOT

- IntArray intWorkSpace

  int work space for DOT

- int numDotNlnConstr

  total number of nonlinear constraints seen by DOT

- int numDotLinConstr

  total number of linear constraints seen by DOT

- int numDotConstr

  total number of linear and nonlinear constraints seen by DOT

- SizetArray constraintMappingIndices

  a container of indices for referencing the corresponding Response constraints used in computing the DOT constraints.

- RealArray constraintMappingMultipliers

  a container of multipliers for mapping the Response constraints to the DOT constraints.

- RealArray constraintMappingOffsets

  a container of offsets for mapping the Response constraints to the DOT constraints.

Additional Inherited Members

13.34.1 Detailed Description

Wrapper class for the DOT optimization library.

The DOTOptimizer class provides a wrapper for DOT, a commercial Fortran 77 optimization library from Vanderplaats Research and Development. It uses a reverse communication mode, which avoids the static member function issues that arise with function pointer designs (see NPSOLOptimizer and SNLLOptimizer).

The user input mappings are as follows: `max_iterations` is mapped into DOT's `ITMAX` parameter within its `IPRM` array, `max_function_evaluations` is implemented directly in the find_optimum() loop since there is no DOT parameter equivalent, `convergence_tolerance` is mapped into DOT's `DELOBJ` parameter (the relative convergence tolerance) within its `RPRM` array, `output` verbosity is mapped into DOT's `IPRINT` parameter within its function call parameter list (verbose: `IPRINT` = 7; quiet: `IPRINT` = 3), and `optimization_type` is mapped into DOT's `MINMAX` parameter within its function call parameter list. Refer to [Vanderplaats Research and Development, 1995] for information on `IPRM`, `RPRM`, and the DOT function call parameter list.

13.34.2 Member Data Documentation

int dotInfo `[private]`

INFO from DOT manual.

Information requested by DOT: 0=optimization complete, 1=get values, 2=get gradients

Referenced by DOTOptimizer::find_optimum(), and DOTOptimizer::initialize_run().

int dotFDSinfo [private]

internal DOT parameter NGOTOZ

the DOT parameter list has been modified to pass NGOTOZ, which signals whether DOT is finite-differencing (nonzero value) or performing the line search (zero value).

Referenced by DOTOptimizer::find_optimum().

int dotMethod [private]

METHOD from DOT manual.

For nonlinear constraints: 0/1 = dot_mmfd, 2 = dot_slp, 3 = dot_sqp. For unconstrained: 0/1 = dot_bfgs, 2 = dot_frcg.

Referenced by DOTOptimizer::allocate_constraints(), DOTOptimizer::allocate_workspace(), DOTOptimizer::DOTOptimizer(), and DOTOptimizer::find_optimum().

int printControl [private]

IPRINT from DOT manual (controls output verbosity)

Values range from 0 (least output) to 7 (most output).

Referenced by DOTOptimizer::DOTOptimizer(), and DOTOptimizer::find_optimum().

RealArray realCntlParmArray [private]

RPRM from DOT manual.

Array of real control parameters.

Referenced by DOTOptimizer::find_optimum(), and DOTOptimizer::initialize().

IntArray intCntlParmArray [private]

IPRM from DOT manual.

Array of integer control parameters.

Referenced by DOTOptimizer::find_optimum(), and DOTOptimizer::initialize().

RealVector constraintValues [private]

array of nonlinear constraint values passed to DOT

This array must be of nonzero length and must contain only one-sided inequality constraints which are <= 0 (which requires a transformation from 2-sided inequalities and equalities).

Referenced by DOTOptimizer::allocate_constraints(), and DOTOptimizer::find_optimum().

SizetArray constraintMappingIndices [private]

a container of indices for referencing the corresponding Response constraints used in computing the DOT constraints.

The length of the container corresponds to the number of DOT constraints, and each entry in the container points to the corresponding DAKOTA constraint.

Referenced by DOTOptimizer::allocate_constraints(), and DOTOptimizer::find_optimum().

RealArray constraintMappingMultipliers `[private]`

a container of multipliers for mapping the Response constraints to the DOT constraints.

The length of the container corresponds to the number of DOT constraints, and each entry in the container stores a multiplier for the DAKOTA constraint identified with constraintMappingIndices. These multipliers are currently +1 or -1.

Referenced by DOTOptimizer::allocate_constraints(), and DOTOptimizer::find_optimum().

RealArray constraintMappingOffsets `[private]`

a container of offsets for mapping the Response constraints to the DOT constraints.

The length of the container corresponds to the number of DOT constraints, and each entry in the container stores an offset for the DAKOTA constraint identified with constraintMappingIndices. These offsets involve inequality bounds or equality targets, since DOT assumes constraint allowables = 0.

Referenced by DOTOptimizer::allocate_constraints(), and DOTOptimizer::find_optimum().

The documentation for this class was generated from the following files:

- DOTOptimizer.hpp
- DOTOptimizer.cpp

13.35 JEGAOptimizer::Driver Class Reference

A subclass of the JEGA front end driver that exposes the individual protected methods to execute the algorithm.

Inherits Driver.

Public Member Functions

- GeneticAlgorithm * ExtractAllData (const AlgorithmConfig &algConfig)

  Reads all required data from the problem description database stored in the supplied algorithm config.
- DesignOFSortSet PerformIterations (GeneticAlgorithm *theGA)

  Performs the required iterations on the supplied GA.
- void DestroyAlgorithm (GeneticAlgorithm *theGA)

  Deletes the supplied GA.
- Driver (const ProblemConfig &probConfig)

  Default constructs a Driver.

13.35.1 Detailed Description

A subclass of the JEGA front end driver that exposes the individual protected methods to execute the algorithm.

This is necessary because DAKOTA requires that all problem information be extracted from the problem description DB at the time of Optimizer construction and the front end does it all in the execute algorithm method which must be called in find_optimum.

13.35.2 Constructor & Destructor Documentation

Driver (const ProblemConfig & *probConfig*) `[inline]`

Default constructs a Driver.

Parameters

probConfig	The definition of the problem to be solved by this Driver whenever ExecuteAlgorithm is called.

The problem can be solved in multiple ways by multiple algorithms even using multiple different evaluators by issuing multiple calls to ExecuteAlgorithm with different AlgorithmConfigs.

13.35.3 Member Function Documentation

GeneticAlgorithm∗ ExtractAllData (const AlgorithmConfig & *algConfig*) [inline]

Reads all required data from the problem description database stored in the supplied algorithm config.

The returned GA is fully configured and ready to be run. It must also be destroyed at some later time. You MUST call DestroyAlgorithm for this purpose. Failure to do so could result in a memory leak and an eventual segmentation fault! Be sure to call DestroyAlgorithm prior to destroying the algorithm config that was used to create it!

This is just here to expose the base class method to users.

Parameters

algConfig	The fully loaded configuration object containing the database of parameters for the algorithm to be run on the known problem.

Returns

The fully configured and loaded GA ready to be run using the PerformIterations method.

Referenced by JEGAOptimizer::find_optimum().

DesignOFSortSet PerformIterations (GeneticAlgorithm ∗ *theGA*) [inline]

Performs the required iterations on the supplied GA.

This includes the calls to AlgorithmInitialize and AlgorithmFinalize and logs some information if appropriate.

This is just here to expose the base class method to users.

Parameters

theGA	The GA on which to perform iterations. This parameter must be non-null.

Returns

The final solutions reported by the supplied GA after all iterations and call to AlgorithmFinalize.

Referenced by JEGAOptimizer::find_optimum().

void DestroyAlgorithm (GeneticAlgorithm ∗ *theGA*) [inline]

Deletes the supplied GA.

Use this method to destroy a GA after all iterations have been run. This method knows if the log associated with the GA was created here and needs to be destroyed as well or not.

This is just here to expose the base class method to users.

Be sure to use this prior to destoying the algorithm config object which contains the target. The GA destructor needs the target to be in tact.

Parameters

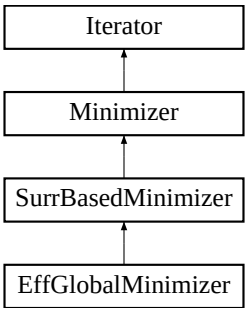
theGA	The algorithm that is no longer needed and thus must be destroyed.

Referenced by JEGAOptimizer::find_optimum().
The documentation for this class was generated from the following file:

- JEGAOptimizer.cpp

13.36 EffGlobalMinimizer Class Reference

Implementation of Efficient Global Optimization/Least Squares algorithms.
Inheritance diagram for EffGlobalMinimizer:



Public Member Functions

- EffGlobalMinimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor
- ~EffGlobalMinimizer ()

  alternate constructor for instantiations "on the fly"
- void minimize_surrogates ()

  Used for computing the optimal solution using a surrogate-based approach. Redefines the Iterator::core_run() virtual function.
- const Model & algorithm_space_model () const

Private Member Functions

- void minimize_surrogates_on_model ()

  called by minimize_surrogates for setUpType == "model"
- void get_best_sample ()

  called by minimize_surrogates for setUpType == "user_functions"
- Real expected_improvement (const RealVector &means, const RealVector &variances)

  expected improvement function for the GP
- RealVector expected_violation (const RealVector &means, const RealVector &variances)

  expected violation function for the constraint functions
- void update_penalty ()

  initialize and update the penaltyParameter

Static Private Member Functions

- static void EIF_objective_eval (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used as the objective function in the Expected Improvement (EIF) problem formulation for PMA

Private Attributes

- String setUpType

  controls iteration mode: "model" (normal usage) or "user_functions" (user-supplied functions mode for "on the fly" instantiations).
- Model fHatModel

  GP model of response, one approximation per response function.
- Model eifModel

  recast model which assimilates mean and variance to solve the max(EIF) sub-problem
- Real meritFnStar

  minimum penalized response from among true function evaluations
- RealVector truthFnStar

  true function values corresponding to the minimum penalized response
- RealVector varStar

  point that corresponds to the optimal value meritFnStar
- short dataOrder

  order of the data used for surrogate construction, in ActiveSet request vector 3-bit format; user may override responses spec

Static Private Attributes

- static EffGlobalMinimizer * effGlobalInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.36.1 Detailed Description

Implementation of Efficient Global Optimization/Least Squares algorithms.

The EffGlobalMinimizer class provides an implementation of the Efficient Global Optimization algorithm developed by Jones, Schonlau, & Welch as well as adaptation of the concept to nonlinear least squares.

13.36.2 Constructor & Destructor Documentation

~EffGlobalMinimizer ( )

alternate constructor for instantiations "on the fly"

destructor

13.36.3 Member Function Documentation

const Model & algorithm_space_model ( ) const [inline], [virtual]

default definition that gets redefined in selected derived Minimizers

Reimplemented from Minimizer.

References EffGlobalMinimizer::fHatModel.

void get_best_sample ( ) [private]

called by minimize_surrogates for setUpType == "user_functions"

determine best solution from among sample data for expected imporovement function

References Model::approximation_data(), SurrBasedMinimizer::augmented_lagrangian_merit(), Model::compute_response(), Model::continuous_variables(), Dakota::copy_data(), Model::current_response(), EffGlobalMinimizer::fHatModel, Response::function_values(), Iterator::iteratedModel, EffGlobalMinimizer::meritFnStar, Minimizer::numFunctions, SurrBasedMinimizer::origNonlinEqTargets, SurrBasedMinimizer::origNonlinIneqLowerBnds, SurrBasedMinimizer::origNonlinIneqUpperBnds, Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), EffGlobalMinimizer::truthFnStar, and EffGlobalMinimizer::varStar.

Referenced by EffGlobalMinimizer::minimize_surrogates_on_model().

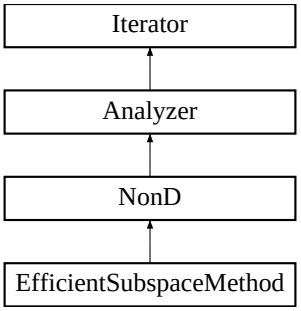
The documentation for this class was generated from the following files:

- EffGlobalMinimizer.hpp
- EffGlobalMinimizer.cpp

13.37 EfficientSubspaceMethod Class Reference

Efficient Subspace Method (ESM), as proposed by Hany S. Abdel-Khalik.

Inheritance diagram for EfficientSubspaceMethod:



Public Member Functions

- EfficientSubspaceMethod (ProblemDescDB &problem_db, Model &model)

  Standard, model-based constructor.
- ~EfficientSubspaceMethod ()

  Destructor.
- void derived_init_communicators (ParLevLIter pl_iter)

  specialization of init comms due to varied use of the original model
- void derived_set_communicators (ParLevLIter pl_iter)

specialization of init comms due to varied use of the original model

- void derived_free_communicators (ParLevLIter pl_iter)

  specialization of free comms due to varied use of the original model

- void quantify_uncertainty ()

  ESM re-implementation of the virtual UQ iterator function.

Private Member Functions

- void validate_inputs ()

  validate user-supplied input values, setting defaults, aborting on error

- void init_fullspace_sampler ()

  initialize the native problem space Monte Carlo sampler

- void expand_basis (bool &mach_svtol_met, bool &user_svtol_met)

  generate fullspace samples, append to matrix, and factor, returning whether tolerance met

- unsigned int calculate_fullspace_samples ()

  determine the number of full space samples for next iteration, based on batchSize, limiting by remaining function evaluation budget

- void generate_fullspace_samples (unsigned int diff_samples)

  sample the derivative at diff_samples points and leave temporary in dace_iterator

- void append_sample_matrices (unsigned int diff_samples)

  append the fullSpaceSampler samples to the derivative and vars matrices

- void compute_svd (bool &mach_svtol_met, bool &user_svtol_met)

  factor the derivative matrix and analyze singular values, assessing convergence and rank, returning whether tolerance met

- void print_svd_stats ()

  print inner iteration stats after SVD

- void assess_reconstruction (bool &recon_tol_met)

  determine if the reduced basis yields acceptable reconstruction error, based on sampling in the orthogonal complement of the reduced basis

- void reduced_space_uq ()

  experimental method to demonstrate creating a RecastModel and perform sampling-based UQ in the reduced space

- void uncertain_vars_to_subspace (Model &native_model, Model &vars_transform_model)

  translate the characterization of uncertain variables in the native_model to the reduced space of the transformed model

Static Private Member Functions

- static void map_xi_to_x (const Variables &recast_xi_vars, Variables &sub_model_x_vars)

  map the active continuous recast variables to the active submodel variables (linear transformation)

Private Attributes

- int seedSpec

  seed controlling all samplers
- int initialSamples

  initial number of samples at which to query the truth model
- int batchSize

  number of points to add at each iteration
- int subspaceSamples

  number of UQ samples to perform in the reduced space
- unsigned int currIter

  current iteration
- unsigned int totalSamples

  total construction samples evaluated so far
- unsigned int totalEvals

  total evaluations of model (accounting for UQ phase)
- double userSVTol

  user-specified tolerance on singular value ratio
- double nullspaceTol

  user-specified tolerance on nullspace
- double svRatio

  current singular value ratio (sigma_k/sigma_0)
- unsigned int reducedRank

  current approximation of system rank
- RealMatrix reducedBasis

  basis for the reduced subspace
- RealMatrix derivativeMatrix

  matrix of derivative data with numFunctions columns per fullspace sample; each column contains the gradient of one function at one sample point, so total matrix size is numContinuousVars ∗ (numFunctions ∗ numSamples) [D1 | D2 | ... | Dnum_samples] [dy1/dx(k=1) | dy2/dx(k=1) | ... | dyM/dx(k=1) | k=2 | ... | k=n_s]
- RealMatrix varsMatrix

  matrix of fullspace variable points samples size numContinuousVars ∗ (numSamples)
- size_t miPLIndex

  index for the active ParallelLevel within ParallelConfiguration::miPLIters
- Iterator fullSpaceSampler

  Monte Carlo sampler for the full parameter space.

Static Private Attributes

- static EfficientSubspaceMethod ∗ esmInstance

  instance of this class for use in static member functions

Additional Inherited Members

13.37.1 Detailed Description

Efficient Subspace Method (ESM), as proposed by Hany S. Abdel-Khalik.

ESM uses random sampling to construct a low-dimensional subspace of the full dimensional parameter space, then performs UQ in the reduced space

13.37.2 Member Function Documentation

void derived_init_communicators (ParLevLIter *pl_iter*) [virtual]

specialization of init comms due to varied use of the original model

This specialization is because the model is used in multiple contexts in this iterator, depending on build phase. Note that this overrides the default behavior at Iterator which recurses into any submodels.

Reimplemented from Iterator.

References EfficientSubspaceMethod::batchSize, Model::init_communicators(), Iterator::iteratedModel, and Iterator::maxEvalConcurrency.

void assess_reconstruction (bool & *recon_tol_met*) [private]

determine if the reduced basis yields acceptable reconstruction error, based on sampling in the orthogonal complement of the reduced basis

This function is experimental and needs to be carefully reviewed and cleaned up

References Iterator::activeSet, Model::aleatory_distribution_parameters(), Model::compute_response(), Model::continuous_variables(), Model::current_response(), Response::function_values(), Iterator::iteratedModel, Iterator::maxFunctionEvals, EfficientSubspaceMethod::nullspaceTol, Analyzer::numContinuousVars, Analyzer::numFunctions, Iterator::outputLevel, EfficientSubspaceMethod::reducedBasis, EfficientSubspaceMethod::reducedRank, ActiveSet::request_values(), EfficientSubspaceMethod::totalEvals, EfficientSubspaceMethod::totalSamples, and EfficientSubspaceMethod::varsMatrix.

Referenced by EfficientSubspaceMethod::quantify_uncertainty().

void reduced_space_uq ( ) [private]

experimental method to demonstrate creating a RecastModel and perform sampling-based UQ in the reduced space

This function is experimental and needs to be reviewed and cleaned up. In particular the translation of the correlations from full to reduced space is likely wrong. Transformation may be correct for covariance, but likely not correlations.

References Model::assign_rep(), NonD::construct_lhs(), Model::free_communicators(), NonD::generate_system_seed(), Model::init_communicators(), Iterator::iteratedModel, EfficientSubspaceMethod::map_xi_to_x(), Iterator::methodPCIter, EfficientSubspaceMethod::miPLIndex, Analyzer::numContinuousVars, Analyzer::numFunctions, Iterator::print_results(), EfficientSubspaceMethod::reducedRank, Iterator::run(), Iterator::sampling_reset(), EfficientSubspaceMethod::seedSpec, Iterator::sub_iterator_flag(), EfficientSubspaceMethod::subspaceSamples, EfficientSubspaceMethod::uncertain_vars_to_subspace(), and Analyzer::vary_pattern().

Referenced by EfficientSubspaceMethod::quantify_uncertainty().

void uncertain_vars_to_subspace (Model & *native_model*, Model & *vars_transform_model*) [private]

translate the characterization of uncertain variables in the native_model to the reduced space of the transformed model

transform and set the distribution parameters in the reduced model

Convert the user-specified normal random variables to the appropriate reduced space variables, based on the orthogonal transformation.

TODO: Generalize to convert other random variable types

References Dakota::abort_handler(), Model::aleatory_distribution_parameters(), Analyzer::numContinuousVars, Iterator::outputLevel, EfficientSubspaceMethod::reducedBasis, and EfficientSubspaceMethod::reducedRank.

Referenced by EfficientSubspaceMethod::reduced_space_uq().

void map_xi_to_x (const Variables & *recast_xi_vars*, Variables & *sub_model_x_vars*) `[static], [private]`

map the active continuous recast variables to the active submodel variables (linear transformation)

Perform the variables mapping from recast reduced dimension variables xi to original model x variables via linear transformation. Maps only continuous variables.

References Variables::continuous_variables(), Dakota::copy_data(), EfficientSubspaceMethod::esmInstance, Iterator::outputLevel, and EfficientSubspaceMethod::reducedBasis.

Referenced by EfficientSubspaceMethod::reduced_space_uq().

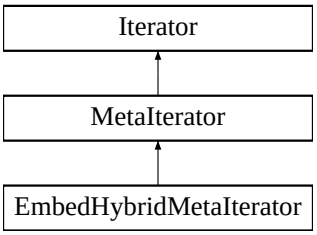
The documentation for this class was generated from the following files:

- EfficientSubspaceMethod.hpp
- EfficientSubspaceMethod.cpp

13.38 EmbedHybridMetaIterator Class Reference

Meta-iterator for closely-coupled hybrid iteration, typically involving the embedding of local search methods within global search methods.

Inheritance diagram for EmbedHybridMetaIterator:



Public Member Functions

- EmbedHybridMetaIterator (ProblemDescDB &problem_db)

  standard constructor
- EmbedHybridMetaIterator (ProblemDescDB &problem_db, Model &model)

  alternate constructor
- ~EmbedHybridMetaIterator ()

  destructor

Protected Member Functions

- void core_run ()

  Performs the hybrid iteration by executing global and local iterators, using a set of models that may vary in fidelity.
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- const Variables & variables_results () const

  return the final solution from the embedded hybrid (variables)
- const Response & response_results () const

  return the final solution from the embedded hybrid (response)

Private Attributes

- Iterator globalIterator

  the top-level outer iterator (e.g., global minimizer)
- Model globalModel

  the model employed by the top-level outer iterator
- Iterator localIterator

  the inner iterator (e.g., local minimizer)
- Model localModel

  the model employed by the inner iterator
- Real localSearchProb

  the probability of running a local search refinement within phases of the global minimization for tightly-coupled hybrids

Additional Inherited Members

13.38.1 Detailed Description

Meta-iterator for closely-coupled hybrid iteration, typically involving the embedding of local search methods within global search methods.

This meta-iterator uses multiple methods in close coordination, generally using a local search minimizer repeatedly within a global minimizer (the local search minimizer refines candidate minima which are fed back to the global minimizer).

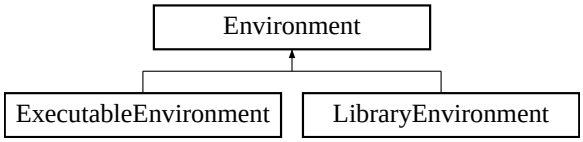
The documentation for this class was generated from the following files:

- EmbedHybridMetaIterator.hpp
- EmbedHybridMetaIterator.cpp

13.39 Environment Class Reference

Base class for the environment class hierarchy.

Inheritance diagram for Environment:



Public Member Functions

- Environment ()

  default constructor: empty envelope
- Environment (int argc, char *argv[ ])

  envelope constructor for ExecutableEnvironment letter
- Environment (ProgramOptions prog_opts)
- Environment (MPI_Comm dakota_mpi_comm, ProgramOptions prog_opts=ProgramOptions())
- Environment (const String &env_type)

  envelope constructor for letter type identifed by String
- Environment (const Environment &env)

  copy constructor
- virtual ~Environment ()

  destructor
- Environment operator= (const Environment &env)

  assignment operator
- virtual void execute ()

  the run function for the environment: invoke the iterator(s) on the model(s). Called from main.cpp.
- bool check () const

  Print status of check and return true if in a "check" mode, including version and help. Return false if proceeding to a run mode.
- MPIManager & mpi_manager ()

  return mpiManager
- ProgramOptions & program_options ()

  return programOptions
- OutputManager & output_manager ()

  return outputManager
- ParallelLibrary & parallel_library ()

  return parallelLib
- ProblemDescDB & problem_description_db ()

  return probDescDB
- const Variables & variables_results () const

  return the final environment solution (variables)

- const Response & response_results () const

  return the final environment solution (response)
- void exit_mode (const String &mode="exit")

  allow environment clients to set Dakota exit behavior (throw vs. exit)

Protected Member Functions

- Environment (BaseConstructor)

  constructor initializes the base class part of default-constructed letters
- Environment (BaseConstructor, int argc, char *argv[ ])

  constructor initializes the base class part of executable letter classes
- Environment (BaseConstructor, ProgramOptions prog_opts, MPI_Comm dakota_mpi_comm=MPI_COMM_WORLD)

  constructor initializes the base class part of library letter classes
- void parse (bool check_bcast_database=true, DbCallbackFunctionPtr callback=NULL, void *callback_data=NULL)

  parse inputs, callbacks, and optionally check and broadcast
- void construct ()

  Instantiate topLevelIterator.
- void destruct ()

  Deallocate parallel partitioning for topLevelIterator.

Protected Attributes

- MPIManager mpiManager

  the MPI manager instance
- ProgramOptions programOptions

  the command line options manager
- OutputManager outputManager

  (tagged) output stream manager
- ParallelLibrary parallelLib

  the parallel library instance
- ProblemDescDB probDescDB

  the parser database instance
- Iterator topLevelIterator

  the top level (meta-)iterator
- UsageTracker usageTracker

  tool for Dakota usage tracking (this is a thin wrapper class)

Private Member Functions

- Environment * get_environment (const String &env_type)

  Used by the envelope to instantiate the correct letter class.

Private Attributes

- Environment * environmentRep

  pointer to the letter (initialized only for the envelope)
- int referenceCount

  number of objects sharing environmentRep

13.39.1 Detailed Description

Base class for the environment class hierarchy.

The Environment class is the base class for the class hierarchy providing the top level control in DAKOTA. The environment is responsible for creating and managing iterators and models. For memory efficiency and enhanced polymorphism, the environment hierarchy employs the "letter/envelope idiom" (see Coplien "Advanced C++", p. 133), for which the base class (Environment) serves as the envelope and one of the derived classes (selected in Environment::get_environment()) serves as the letter.

13.39.2 Constructor & Destructor Documentation

Environment ( )

default constructor: empty envelope

Default envelope constructor. environmentRep is NULL in this case, which makes it necessary to check for NULL in the copy constructor, assignment operator, and destructor.

Environment (int *argc*, char * *argv[ ]*)

envelope constructor for ExecutableEnvironment letter

Envelope constructor for ExecutableEnvironment. Selection of derived type by get_environment() is not necessary in this case.

References Dakota::abort_handler(), and Environment::environmentRep.

Environment (ProgramOptions *prog_opts*)

Envelope constructor for LibraryEnvironment. Selection of derived type by get_environment() is not necessary in this case.

References Dakota::abort_handler(), and Environment::environmentRep.

Environment (MPI_Comm *dakota_mpi_comm*, ProgramOptions *prog_opts* = ProgramOptions *()*)

Envelope constructor for LibraryEnvironment. Selection of derived type by get_environment() is not necessary in this case.

References Dakota::abort_handler(), and Environment::environmentRep.

Environment (const String & *env_type*)

envelope constructor for letter type identifed by String

Alternate construction by String. Envelope constructor invokes get_environment() which instantiates a derived class letter; the derived constructor selects a BaseConstructor constructor in its initialization list to avoid the recursion of a base class constructor calling get_environment() again.

References Dakota::abort_handler(), Environment::environmentRep, and Environment::get_environment().

Environment (const Environment & *env*)

copy constructor

Copy constructor manages sharing of environmentRep and incrementing of referenceCount.

References Environment::environmentRep, and Environment::referenceCount.

∼Environment ( ) `[virtual]`

destructor

Destructor decrements referenceCount and only deletes environmentRep when referenceCount reaches zero.

References Environment::destruct(), Environment::environmentRep, and Environment::referenceCount.

Environment (BaseConstructor) `[protected]`

constructor initializes the base class part of default-constructed letters

This letter constructor initializes base class data for inherited environments that are default constructed. Since the letter IS the representation, its representation pointer is set to NULL (an uninitialized pointer causes problems in ∼Environment).

Use cases: library with no options, no MPI comm

References ProgramOptions::exit_mode(), Environment::exit_mode(), WorkdirHelper::initialize(), and Environment::programOptions.

Environment (BaseConstructor , int *argc*, char ∗ *argv[ ]*) `[protected]`

constructor initializes the base class part of executable letter classes

This letter constructor initializes base class data for inherited environments: instantiate/initialize the environment, options, parallel library, and problem description database objects. Since the letter IS the representation, its representation pointer is set to NULL (an uninitialized pointer causes problems in ∼Environment).

Use cases: executable with command-line args

References ProgramOptions::exit_mode(), Environment::exit_mode(), WorkdirHelper::initialize(), and Environment::programOptions.

Environment (BaseConstructor , ProgramOptions *prog_opts*, MPI_Comm *dakota_mpi_comm* = *MPI_COMM_WORLD*) `[protected]`

constructor initializes the base class part of library letter classes

This letter constructor initializes base class data for inherited environments. Since the letter IS the representation, its representation pointer is set to NULL (an uninitialized pointer causes problems in ∼Environment).

Use cases: library with program options library with program options and MPI comm

References ProgramOptions::exit_mode(), Environment::exit_mode(), WorkdirHelper::initialize(), and Environment::programOptions.

13.39.3 Member Function Documentation

Environment operator= (const Environment & *env*)

assignment operator

Assignment operator decrements referenceCount for old environmentRep, assigns new environmentRep, and increments referenceCount for new environmentRep.

References Environment::environmentRep, and Environment::referenceCount.

void exit_mode (const String & *mode* = `"exit"`)

allow environment clients to set Dakota exit behavior (throw vs. exit)

Set the global variable controlling Dakota's exit behavior. Call with no arguments to reset to default behavior.

References Dakota::abort_handler(), and Dakota::abort_mode.

Referenced by Environment::Environment(), and run_dakota_data().

void parse (bool *check_bcast_database* = `true`, DbCallbackFunctionPtr *callback* = `NULL`, void * *callback_data* = `NULL`) `[protected]`

parse inputs, callbacks, and optionally check and broadcast

Parse input file and invoked any callbacks, then optionally check and sync database if check_bcast_database = true

References ProblemDescDB::check_and_broadcast(), ProgramOptions::input_file(), ProgramOptions::input_string(), ProblemDescDB::parse_inputs(), Environment::probDescDB, and Environment::programOptions.

Referenced by ExecutableEnvironment::ExecutableEnvironment(), and LibraryEnvironment::LibraryEnvironment().

Environment * get_environment (const String & *env_type*) `[private]`

Used by the envelope to instantiate the correct letter class.

Used only by the envelope constructor to initialize environmentRep to the appropriate derived type, as given by the environmentName attribute.

Referenced by Environment::Environment().

The documentation for this class was generated from the following files:

- DakotaEnvironment.hpp
- DakotaEnvironment.cpp

13.40 NomadOptimizer::Evaluator Class Reference

NOMAD-based Evaluator class.

Inherits Evaluator.

Public Member Functions

- Evaluator (const NOMAD::Parameters &p, Model &model)

  Constructor.

- ~Evaluator (void)

  Destructor.

- bool eval_x (NOMAD::Eval_Point &x, const NOMAD::Double &h_max, bool &count_eval) const

  Main Evaluation Method.

- void set_constraint_map (int numNomadNonlinearIneqConstraints, int numNomadNonlinearEqConstraints, std::vector< int > constraintMapIndices, std::vector< double > constraintMapMultipliers, std::vector< double > constraintMapOffsets)

  publishes constraint transformation

Private Attributes

- Model & **_model**
- int **n_cont**
- int **n_disc_int**
- int **n_disc_real**
- int numNomadNonlinearIneqConstr

  Number of nonlinear constraints after put into Nomad format.
- int **numNomadNonlinearEqConstr**
- std::vector< int > constrMapIndices

  map from Dakota constraint number to Nomad constraint number
- std::vector< double > constrMapMultipliers

  multipliers for constraint transformations
- std::vector< double > constrMapOffsets

  offsets for constraint transformations

13.40.1 Detailed Description

NOMAD-based Evaluator class.

The NOMAD process requires an evaluation step, which calls the Simulation program. In the simplest version of this call, NOMAD executes the black box executable, which proceeds to write a file in a NOMAD-compatible format, which NOMAD reads to continue the process.

Because DAKOTA files are different form NOMAD files, and the simulations processed by DAKOTA already produce DAKOTA-compatible files, we cannot use this method for NOMAD. Instead, we implement the `Nomad-Evaluator` class, which takes the NOMAD inputs and passes them to DAKOTA's Interface for processing. The evaluator then passes the evaluation Responses into the NOMAD objects for further analysis.

13.40.2 Constructor & Destructor Documentation

Evaluator (const NOMAD::Parameters & *p*, Model & *model*)

Constructor.

```
NOMAD Evaluator Constructor
```

Parameters

p	NOMAD Parameters object
model	DAKOTA Model object

13.40.3 Member Function Documentation

bool eval_x (NOMAD::Eval_Point & *x*, const NOMAD::Double & *h_max*, bool & *count_eval*) const

Main Evaluation Method.

```
Method that handles the communication between
```

the NOMAD search process and the Black Box Evaluation managed by DAKOTA's Interface.

Parameters

x	Object that contains the points that need to evaluated. Once the evaluation is completed, this object also stores the output back to be read by NOMAD.
h_max	Current value of the barrier parameter. Not used in this implementation.
count_eval	Flag that indicates whether this evaluation counts towards the max number of evaluations, often set to `false` when the evaluation does not meet certain costs during expensive evaluations. Not used in this implementation.

Returns

`true` if the evaluation was successful; `false` otherwise.

References Dakota::set_index_to_value().
The documentation for this class was generated from the following files:

- NomadOptimizer.hpp
- NomadOptimizer.cpp

13.41 JEGAOptimizer::Evaluator Class Reference

An evaluator specialization that knows how to interact with Dakota.
Inherits GeneticAlgorithmEvaluator.

Public Member Functions

- virtual bool Evaluate (DesignGroup &group)

  Does evaluation of each design in group.
- virtual bool Evaluate (Design &des)

  This method cannot be used!!
- virtual std::string GetName () const

  Returns the proper name of this operator.
- virtual std::string GetDescription () const

  Returns a full description of what this operator does and how.
- virtual GeneticAlgorithmOperator ∗ Clone (GeneticAlgorithm &algorithm) const

  Creates and returns a pointer to an exact duplicate of this operator.
- Evaluator (GeneticAlgorithm &algorithm, Model &model)

  Constructs a Evaluator for use by algorithm.
- Evaluator (const Evaluator &copy)

  Copy constructs a Evaluator.
- Evaluator (const Evaluator &copy, GeneticAlgorithm &algorithm, Model &model)

  Copy constructs a Evaluator for use by algorithm.

Static Public Member Functions

- static const std::string & Name ()

  Returns the proper name of this operator.
- static const std::string & Description ()

  Returns a full description of what this operator does and how.

Protected Member Functions

- void SeparateVariables (const Design &from, RealVector &intoCont, IntVector &intoDiscInt, RealVector &intoDiscReal) const

  This method fills intoCont, intoDiscInt and intoDiscReal appropriately using the values of from.
- void RecordResponses (const RealVector &from, Design &into) const

  Records the computed objective and constraint function values into into.
- std::size_t GetNumberNonLinearConstraints () const

  Returns the number of non-linear constraints for the problem.
- std::size_t GetNumberLinearConstraints () const

  Returns the number of linear constraints for the problem.

Private Member Functions

- Evaluator (GeneticAlgorithm &algorithm)

  This constructor has no implementation and cannot be used.

Private Attributes

- Model & _model

  The Model known by this evaluator.

13.41.1 Detailed Description

An evaluator specialization that knows how to interact with Dakota.

This evaluator knows how to use the model to do evaluations both in synchronous and asynchronous modes.

13.41.2 Constructor & Destructor Documentation

Evaluator (GeneticAlgorithm & *algorithm*, Model & *model*) `[inline]`

Constructs a Evaluator for use by *algorithm*.

The optimizer is needed for purposes of variable scaling.

Parameters

algorithm	The GA for which the new evaluator is to be used.
model	The model through which evaluations will be done.

Evaluator (const Evaluator & *copy*) `[inline]`

Copy constructs a Evaluator.

Parameters

copy	The evaluator from which properties are to be duplicated into this.

Evaluator (const Evaluator & *copy*, GeneticAlgorithm & *algorithm*, Model & *model*) `[inline]`

Copy constructs a Evaluator for use by *algorithm*.

The optimizer is needed for purposes of variable scaling.

Parameters

copy	The existing Evaluator from which to retrieve properties.
algorithm	The GA for which the new evaluator is to be used.
model	The model through which evaluations will be done.

Evaluator (GeneticAlgorithm & *algorithm*) `[private]`

This constructor has no implementation and cannot be used.

This constructor can never be used. It is provided so that this operator can still be registered in an operator registry even though it can never be instantiated from there.

Parameters

algorithm	The GA for which the new evaluator is to be used.

13.41.3 Member Function Documentation

static const std::string& Name ( ) `[inline]`, `[static]`

Returns the proper name of this operator.

Returns

The string "DAKOTA JEGA Evaluator".

static const std::string& Description ( ) `[inline]`, `[static]`

Returns a full description of what this operator does and how.

The returned text is:

```
This evaluator uses Sandia's DAKOTA optimization
software to evaluate the passed in Designs.  This
makes it possible to take advantage of the fact that
DAKOTA is designed to run on massively parallel machines.
```

.

Returns

A description of the operation of this operator.

void SeparateVariables (const Design & *from*, RealVector & *intoCont*, IntVector & *intoDiscInt*, RealVector & *intoDiscReal*) const `[protected]`

This method fills *intoCont*, *intoDiscInt* and *intoDiscReal* appropriately using the values of *from*.

The discrete integer design variable values are placed in *intoDiscInt*, the discrete real design variable values are placed in *intoDiscReal*, and the continuum are placed into *intoCont*. The values are written into the vectors from the beginning so any previous contents of the vectors will be overwritten.

Parameters

from	The Design class object from which to extract the discrete design variable values.
intoDiscInt	The vector into which to place the extracted discrete integer values.
intoDiscReal	The vector into which to place the extracted discrete real values.
intoCont	The vector into which to place the extracted continuous values.

References JEGAOptimizer::Evaluator::_model, Model::cv(), Model::discrete_int_sets(), Model::div(), and Model::drv().

void RecordResponses (const RealVector & *from*, Design & *into*) const `[protected]`

Records the computed objective and constraint function values into *into*.

This method takes the response values stored in *from* and properly transfers them into the *into* design.

The response vector *from* is expected to contain values for each objective function followed by values for each non-linear constraint in the order in which the info objects were loaded into the target by the optimizer class.

Parameters

from	The vector of responses to install into *into*.
into	The Design to which the responses belong and into which they must be written.

std::size_t GetNumberNonLinearConstraints ( ) const `[inline], [protected]`

Returns the number of non-linear constraints for the problem.

This is computed by adding the number of non-linear equality constraints to the number of non-linear inequality constraints. These values are obtained from the model.

Returns

The total number of non-linear constraints.

std::size_t GetNumberLinearConstraints ( ) const `[inline], [protected]`

Returns the number of linear constraints for the problem.

This is computed by adding the number of linear equality constraints to the number of linear inequality constraints. These values are obtained from the model.

Returns

The total number of linear constraints.

bool Evaluate (DesignGroup & *group*) `[virtual]`

Does evaluation of each design in *group*.

This method uses the Model known by this class to get Designs evaluated. It properly formats the Design class information in a way that Dakota will understand and then interprets the Dakota results and puts them back into the Design class object. It respects the asynchronous flag in the Model so evaluations may occur synchronously or asynchronously.

Prior to evaluating a Design, this class checks to see if it is marked as already evaluated. If it is, then the evaluation of that Design is not carried out. This is not strictly necessary because Dakota keeps track of evaluated designs and does not re-evaluate. An exception is the case of a population read in from a file complete with responses where Dakota is unaware of the evaluations.

Parameters

group	The group of Design class objects to be evaluated.

Returns

true if all evaluations completed and false otherwise.

virtual bool Evaluate (Design & *des*) `[inline], [virtual]`

This method cannot be used!!

This method does nothing and cannot be called. This is because in the case of asynchronous evaluation, this method would be unable to conform. It would require that each evaluation be done in a synchronous fashion.

Parameters

des	A Design that would be evaluated if this method worked.

Returns

Would return true if the Design were evaluated and false otherwise. Never actually returns here. Issues a fatal error. Otherwise, it would always return false.

virtual std::string GetName ( ) const `[inline], [virtual]`

Returns the proper name of this operator.

Returns

See Name().

virtual std::string GetDescription ( ) const `[inline], [virtual]`

Returns a full description of what this operator does and how.

Returns

See Description().

virtual GeneticAlgorithmOperator* Clone (GeneticAlgorithm & *algorithm*) const `[inline], [virtual]`

Creates and returns a pointer to an exact duplicate of this operator.

Parameters

algorithm	The GA for which the clone is being created.

Returns

A clone of this operator.

13.41.4 Member Data Documentation

Model& _model [private]

The Model known by this evaluator.

It is through this model that evaluations will take place.

Referenced by JEGAOptimizer::Evaluator::SeparateVariables().

The documentation for this class was generated from the following file:

- JEGAOptimizer.cpp

13.42 JEGAOptimizer::EvaluatorCreator Class Reference

A specialization of the JEGA::FrontEnd::EvaluatorCreator that creates a new instance of a Evaluator.

Inherits EvaluatorCreator.

Public Member Functions

- virtual GeneticAlgorithmEvaluator * CreateEvaluator (GeneticAlgorithm &alg)

  Overriden to return a newly created Evaluator.
- EvaluatorCreator (Model &theModel)

  Constructs an EvaluatorCreator using the supplied model.

Private Attributes

- Model & _theModel

  The user defined model to be passed to the constructor of the Evaluator.

13.42.1 Detailed Description

A specialization of the JEGA::FrontEnd::EvaluatorCreator that creates a new instance of a Evaluator.

13.42.2 Constructor & Destructor Documentation

EvaluatorCreator (Model & *theModel*) [inline]

Constructs an EvaluatorCreator using the supplied model.

Parameters

theModel	The Dakota::Model this creator will pass to the created evaluator.

13.42.3 Member Function Documentation

virtual GeneticAlgorithmEvaluator* CreateEvaluator (GeneticAlgorithm & *alg*) [inline], [virtual]

Overriden to return a newly created Evaluator.

The GA will assume ownership of the evaluator so we needn't worry about keeping track of it for destruction. The additional parameters needed by the Evaluator are stored as members of this class at construction time.

Parameters

alg	The GA for which the evaluator is to be created.

Returns

A pointer to a newly created Evaluator.

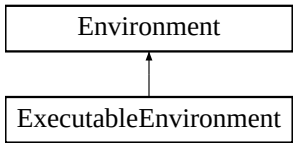
The documentation for this class was generated from the following file:

- JEGAOptimizer.cpp

13.43 ExecutableEnvironment Class Reference

Environment corresponding to execution as a stand-alone application.

Inheritance diagram for ExecutableEnvironment:



Public Member Functions

- ExecutableEnvironment ()

  default constructor
- ExecutableEnvironment (int argc, char ∗argv[ ])

  constructor
- ∼ExecutableEnvironment ()

  destructor
- void execute ()

  the run function for the environment: invoke the iterator(s) on the model(s). Called from main.cpp.

Additional Inherited Members

13.43.1 Detailed Description

Environment corresponding to execution as a stand-alone application.

This environment corresponds to a stand-alone executable program, e.g., main.cpp. It sets up the ParallelLibrary, ProgramOptions, and ProblemDescDB objects based on access to command line arguments.

The documentation for this class was generated from the following files:

- ExecutableEnvironment.hpp
- ExecutableEnvironment.cpp

13.44 ExperimentData Class Reference

Public Member Functions

- void load_scalar (const std::string &expDataFilename, const std::string &context_message, size_t numExperiments, size_t numExpConfigVars, size_t numFunctions, size_t numExpStdDeviationsRead, bool expDataFileAnnotated, bool calc_sigma_from_data, short verbosity)

  constructor

- const RealVector & config_vars (size_t response, size_t experiment)

  retrieve the vector of configuration variables for the given response and experiment number

- Real scalar_data (size_t response, size_t experiment)

  retrieve the data value for the given response, for the given experiment

- Real scalar_sigma (size_t response, size_t experiment)

  retrieve the standard deviation value for the given response, for the given experiment

Public Attributes

- std::vector< ExpDataPerResponse > allExperiments

  At the outer level, ExperimentData will just be a vector of ExpDataPerResponse;.

13.44.1 Detailed Description

The ExperimentData class is used to read and populate data (currently from user-specified files and/or the input spec) relating to experimental (physical observations) data for the purposes of calibration. Such data may include (for example): number of experiments, configuration variables, type of data (scalar vs. functional), treatment of sigma (experimental uncertainties). This class also provides an interpolation capability to interpolate between simulation or experimental data so that the differencing between simulation and experimental data may be performed properly.

13.44.2 Member Function Documentation

void load_scalar (const std::string & *expDataFilename,* const std::string & *context_message,* size_t *numExperiments,* size_t *numExpConfigVars,* size_t *numFunctions,* size_t *numExpStdDeviationsRead,* bool *expDataFileAnnotated,* bool *calc_sigma_from_data,* short *verbosity*)

constructor

Constructor from legacy file format

References Dakota::read_historical_data().

Referenced by Minimizer::data_transform_model(), NonDQUESOBayesCalibration::quantify_uncertainty(), NonDDREAMBayesCalibration::quantify_uncertainty(), and NonDGPMSABayesCalibration::quantify_uncertainty().

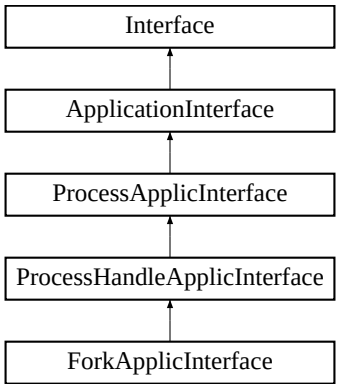
The documentation for this class was generated from the following files:

- ExperimentData.hpp
- ExperimentData.cpp

13.45 ForkApplicInterface Class Reference

Derived application interface class which spawns simulation codes using fork/execvp/waitpid.

Inheritance diagram for ForkApplicInterface:



Public Member Functions

- ForkApplicInterface (const ProblemDescDB &problem_db)

  constructor
- ~ForkApplicInterface ()

  destructor

Protected Member Functions

- void wait_local_evaluations (PRPQueue &prp_queue)

  For asynchronous function evaluations, this method is used to detect completion of jobs and process their results. It provides the processing code that is specific to derived classes. This version waits for at least one completion.
- void test_local_evaluations (PRPQueue &prp_queue)

  For asynchronous function evaluations, this method is used to detect completion of jobs and process their results. It provides the processing code that is specific to derived classes. This version is nonblocking and will return without any completions if none are immediately available.
- pid_t create_analysis_process (bool block_flag, bool new_group)

  spawn a child process for an analysis component within an evaluation using fork()/execvp() and wait for completion using waitpid() if block_flag is true
- size_t wait_local_analyses ()

  wait for asynchronous analyses on the local processor, completing at least one job
- size_t test_local_analyses_send (int analysis_id)

  test for asynchronous analysis completions on the local processor and return results for any completions by sending messages
- void join_evaluation_process_group (bool new_group)

  create (if new_group) and join the process group for asynch evaluations
- void join_analysis_process_group (bool new_group)

  create (if new_group) and join the process group for asynch analyses

- void evaluation_process_group_id (pid_t pgid)

  set evalProcGroupId
- pid_t evaluation_process_group_id () const

  return evalProcGroupId
- void analysis_process_group_id (pid_t pgid)

  set analysisProcGroupId
- pid_t analysis_process_group_id () const

  return analysisProcGroupId
- pid_t wait_evaluation (bool block_flag)

  process all available completions within the evaluation process group; if block_flag = true, wait for at least one completion
- pid_t wait_analysis (bool block_flag)

  process all available completions within the analysis process group; if block_flag = true, wait for at least one completion
- void check_group (int err, pid_t proc_group_id)

  check the exit status of setpgid and abort if an error code was returned

Private Member Functions

- pid_t wait (pid_t proc_group_id, std::map< pid_t, int > &process_id_map, bool block_flag)

  core code used by wait_{evaluation,analysis}()
- void join_process_group (pid_t &process_group_id, bool new_group)

  core code used by join_{evaluation,analysis}_process_group()

Private Attributes

- pid_t evalProcGroupId

  the process group id used to identify a set of child evaluation processes used by this interface instance (to distinguish from other interface instances that could be running at the same time)
- pid_t analysisProcGroupId

  the process group id used to identify a set of child analysis processes used by this interface instance (to distinguish from other interface instances that could be running at the same time)

Additional Inherited Members

13.45.1 Detailed Description

Derived application interface class which spawns simulation codes using fork/execvp/waitpid.

ForkApplicInterface is used on Unix systems and is a peer to SpawnApplicInterface for Windows systems.

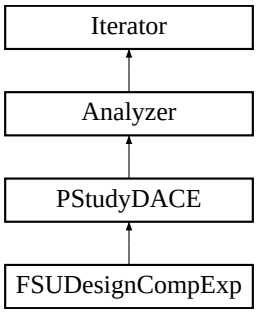
The documentation for this class was generated from the following files:

- ForkApplicInterface.hpp
- ForkApplicInterface.cpp

13.46 FSUDesignCompExp Class Reference

Wrapper class for the FSUDace QMC/CVT library.

Inheritance diagram for FSUDesignCompExp:



Public Member Functions

- FSUDesignCompExp (ProblemDescDB &problem_db, Model &model)

  primary constructor for building a standard DACE iterator
- FSUDesignCompExp (Model &model, int samples, int seed, unsigned short sampling_method)

  alternate constructor for building a DACE iterator on-the-fly
- ~FSUDesignCompExp ()

  destructor
- void pre_run ()

  pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori
- void extract_trends ()

  Mapping of the core_run() virtual function for the PStudy/DACE branch.
- void post_input ()

  read tabular data for post-run mode
- void post_run (std::ostream &s)

  post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way
- int num_samples () const
- void sampling_reset (int min_samples, bool all_data_flag, bool stats_flag)

  reset sampling iterator to use at least min_samples
- unsigned short sampling_scheme () const

  return sampling name
- void vary_pattern (bool pattern_flag)

  sets varyPattern in derived classes that support it
- void get_parameter_sets (Model &model)

  Returns one block of samples (ndim ∗ num_samples)

Private Member Functions

- void enforce_input_rules ()

  enforce sanity checks/modifications for the user input specification

Private Attributes

- int samplesSpec

  initial specification of number of samples
- int numSamples

  current number of samples to be evaluated
- bool allDataFlag

  flag which triggers the update of allVars/allResponses for use by Iterator::all_variables() and Iterator::all_responses()
- size_t numDACERuns

  counter for number of executions for this object
- bool latinizeFlag

  flag which specifies latinization of QMC or CVT sample sets
- IntVector sequenceStart

  Integer vector defining a starting index into the sequence for random variable sampled. Default is 0 0 0 (e.g. for three random variables).
- IntVector sequenceLeap

  Integer vector defining the leap number for each sequence being generated. Default is 1 1 1 (e.g. for three random vars.)
- IntVector primeBase

  Integer vector defining the prime base for each sequence being generated. Default is 2 3 5 (e.g., for three random vars.)
- int seedSpec

  the user seed specification for the random number generator (allows repeatable results)
- int randomSeed

  current seed for the random number generator
- bool varyPattern

  flag for continuing the random number or QMC sequence from a previous execution (e.g., for surrogate-based optimization) so that multiple executions are repeatable but not identical.
- int numCVTTrials

  specifies the number of sample points taken at internal CVT iteration
- int trialType

  Trial type in CVT. Specifies where the points are placed for consideration relative to the centroids. Choices are grid (2), halton (1), uniform (0), or random (-1). Default is random.

Additional Inherited Members

13.46.1 Detailed Description

Wrapper class for the FSUDace QMC/CVT library.

The FSUDesignCompExp class provides a wrapper for FSUDace, a C++ design of experiments library from Florida State University. This class uses quasi Monte Carlo (QMC) and Centroidal Voronoi Tesselation (CVT) methods to uniformly sample the parameter space spanned by the active bounds of the current Model. It returns all generated samples and their corresponding responses as well as the best sample found.

13.46.2 Constructor & Destructor Documentation

FSUDesignCompExp (ProblemDescDB & *problem_db*, Model & *model*)

primary constructor for building a standard DACE iterator

This constructor is called for a standard iterator built with data from probDescDB.

References Dakota::abort_handler(), ProblemDescDB::get_bool(), ProblemDescDB::get_int(), ProblemDescDB::get_iv(), ProblemDescDB::get_string(), Iterator::maxEvalConcurrency, Iterator::methodName, Analyzer::numContinuousVars, FSUDesignCompExp::numCVTTrials, FSUDesignCompExp::numSamples, FSUDesignCompExp::primeBase, Iterator::probDescDB, FSUDesignCompExp::randomSeed, FSUDesignCompExp::seedSpec, FSUDesignCompExp::sequenceLeap, FSUDesignCompExp::sequenceStart, FSUDesignCompExp::trialType, and FSUDesignCompExp::varyPattern.

FSUDesignCompExp (Model & *model*, int *samples*, int *seed*, unsigned short *sampling_method*)

alternate constructor for building a DACE iterator on-the-fly

This alternate constructor is used for instantiations on-the-fly, using only the incoming data. No problem description database queries are used.

References Dakota::abort_handler(), Iterator::maxEvalConcurrency, Iterator::methodName, Analyzer::numContinuousVars, FSUDesignCompExp::numCVTTrials, FSUDesignCompExp::numSamples, FSUDesignCompExp::primeBase, FSUDesignCompExp::randomSeed, FSUDesignCompExp::seedSpec, FSUDesignCompExp::sequenceLeap, FSUDesignCompExp::sequenceStart, and FSUDesignCompExp::trialType.

13.46.3 Member Function Documentation

void pre_run ( ) `[virtual]`

pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

pre-run phase, which a derived iterator may optionally reimplement; when not present, pre-run is likely integrated into the derived run function. This is a virtual function; when re-implementing, a derived class must call its nearest parent's pre_run(), if implemented, typically *before* performing its own implementation steps.

Reimplemented from Iterator.

References FSUDesignCompExp::get_parameter_sets(), Iterator::iteratedModel, and PStudyDACE::varBasedDecompFlag.

void post_run (std::ostream & *s*) `[virtual]`

post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/Responses and perform final analysis phase in a standalone way

Post-run phase, which a derived iterator may optionally reimplement; when not present, post-run is likely integrated into run. This is a virtual function; when re-implementing, a derived class must call its nearest parent's post_run(), typically *after* performing its own implementation steps.

Reimplemented from Analyzer.

References Analyzer::allResponses, Analyzer::allSamples, SensAnalysisGlobal::compute_correlations(), Analyzer::post_run(), PStudyDACE::pStudyDACESensGlobal, Iterator::subIteratorFlag, and PStudyDACE::varBasedDecompFlag.

int num_samples ( ) const `[inline], [virtual]`

Return current number of evaluation points. Since the calculation of samples, collocation points, etc. might be costly, provide a default implementation here that backs out from the maxEvalConcurrency.

Reimplemented from Analyzer.
References FSUDesignCompExp::numSamples.

void enforce_input_rules ( ) [private]

enforce sanity checks/modifications for the user input specification

Users may input a variety of quantities, but this function must enforce any restrictions imposed by the sampling algorithms.

References Dakota::abort_handler(), Iterator::methodName, Analyzer::numContinuousVars, FSUDesignCompExp::numSamples, and FSUDesignCompExp::primeBase.

Referenced by FSUDesignCompExp::get_parameter_sets().

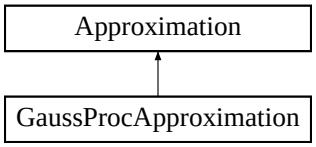
The documentation for this class was generated from the following files:

- FSUDesignCompExp.hpp
- FSUDesignCompExp.cpp

13.47 GaussProcApproximation Class Reference

Derived approximation class for Gaussian Process implementation.

Inheritance diagram for GaussProcApproximation:



Public Member Functions

- GaussProcApproximation ()

  default constructor
- GaussProcApproximation (const SharedApproxData &shared_data)

  alternate constructor
- GaussProcApproximation (const ProblemDescDB &problem_db, const SharedApproxData &shared_data)

  standard constructor
- ~GaussProcApproximation ()

  destructor

Protected Member Functions

- int min_coefficients () const

  return the minimum number of samples (unknowns) required to build the derived class approximation type in numVars dimensions
- int num_constraints () const

  return the number of constraints to be enforced via an anchor point
- void build ()

  find the covariance parameters governing the Gaussian process response

- Real value (const Variables &vars)

  retrieve the function value for a given parameter set
- const RealVector & gradient (const Variables &vars)

  retrieve the function gradient at the predicted value for a given parameter set
- Real prediction_variance (const Variables &vars)

  retrieve the variance of the predicted value for a given parameter set

Private Member Functions

- void GPmodel_build ()

  Function to compute hyperparameters governing the GP.
- void GPmodel_apply (const RealVector &new_x, bool variance_flag, bool gradients_flag)

  Function returns a response value using the GP surface.
- void normalize_training_data ()

  Normalizes the initial inputs upon which the GP surface is based.
- void get_trend ()

  Gets the trend (basis) functions for the calculation of the mean of the GP If the order = 0, the trend is a constant, if the order = 1, trend is linear, if order = 2, trend is quadratic.
- void get_beta_coefficients ()

  Gets the beta coefficients for the calculation of the mean of the GP.
- int get_cholesky_factor ()

  Gets the Cholesky factorization of the covariance matrix, with error checking.
- void get_process_variance ()

  Gets the estimate of the process variance given the values of beta and the correlation lengthscales.
- void get_cov_matrix ()

  calculates the covariance matrix for a given set of input points
- void get_cov_vector ()

  calculates the covariance vector between a new point x and the set of inputs upon which the GP is based
- void optimize_theta_global ()

  sets up and performs the optimization of the negative log likelihood to determine the optimal values of the covariance parameters using NCSUDirect
- void optimize_theta_multipoint ()

  sets up and performs the optimization of the negative log likelihood to determine the optimal values of the covariance parameters using a gradient-based solver and multiple starting points
- void predict (bool variance_flag, bool gradients_flag)

  Calculates the predicted new response value for x in normalized space.
- Real calc_nll ()

  calculates the negative log likelihood function (based on covariance matrix)
- void calc_grad_nll ()

  Gets the gradient of the negative log likelihood function with respect to the correlation lengthscales, theta.
- void get_grad_cov_vector ()

  Calculate the derivatives of the covariance vector, with respect to each componeent of x.
- void run_point_selection ()

Runs the point selection algorithm, which will choose a subset of the training set with which to construct the GP model, and estimate the necessary parameters.

- void initialize_point_selection ()

  Initializes the point selection routine by choosing a small initial subset of the training points.

- void pointsel_get_errors (RealArray &delta)

  Uses the current GP model to compute predictions at all of the training points and find the errors.

- int addpoint (int, IntArray &added_index)

  Adds a point to the effective training set. Returns 1 on success.

- int pointsel_add_sel (const RealArray &delta)

  Accepts a vector of unsorted prediction errors, determines which points should be added to the effective training set, and adds them.

- Real maxval (const RealArray &) const

  Return the maximum value of the elements in a vector.

- void pointsel_write_points ()

  Writes out the training set before and after point selection.

- void lhood_2d_grid_eval ()

  For problems with 2D input, evaluates the negative log likelihood on a grid.

- void writex (const char[ ])

  Writes out the current training set (in original units) to a specified file.

- void writeCovMat (char[ ])

  Writes out the covariance matrix to a specified file.

Static Private Member Functions

- static void negloglik (int mode, int n, const Teuchos::SerialDenseVector< int, double > &X, Real &fx, Teuchos::SerialDenseVector< int, double > &grad_x, int &result_mode)

  static function used by OPT++ as the objective function to optimize the hyperparameters in the covariance of the GP by minimizing the negative log likelihood

- static void constraint_eval (int mode, int n, const Teuchos::SerialDenseVector< int, double > &X, Teuchos::SerialDenseVector< int, double > &g, Teuchos::SerialDenseMatrix< int, double > &gradC, int &result_mode)

  static function used by OPT++ as the constraint function in the optimization of the negative log likelihood. Currently this function is empty: it is an unconstrained optimization.

- static double negloglikNCSU (const RealVector &x)

  function used by NCSUOptimizer to optimize negloglik objective

Private Attributes

- Real approxValue

  value of the approximation returned by value()

- Real approxVariance

  value of the approximation returned by prediction_variance()

- RealMatrix trainPoints

  A 2-D array (num sample sites = rows, num vars = columns) used to create the Gaussian process.

- RealMatrix trainValues

An array of response values; one response value per sample site.

- RealVector trainMeans

  The mean of the input columns of trainPoints.

- RealVector trainStdvs

  The standard deviation of the input columns of trainPoints.

- RealMatrix normTrainPoints

  Current working set of normalized points upon which the GP is based.

- RealMatrix trendFunction

  matrix to hold the trend function

- RealMatrix betaCoeffs

  matrix to hold the beta coefficients for the trend function

- RealSymMatrix covMatrix

  The covariance matrix where each element (i,j) is the covariance between points Xi and Xj in the initial set of samples.

- RealMatrix covVector

  The covariance vector where each element (j,0) is the covariance between a new point X and point Xj from the initial set of samples.

- RealMatrix approxPoint

  Point at which a prediction is requested. This is currently a single point, but it could be generalized to be a vector of points.

- RealMatrix gradNegLogLikTheta

  matrix to hold the gradient of the negative log likelihood with respect to the theta correlation terms

- Teuchos::SerialSpdDenseSolver
  < int, Real > covSlvr

  The global solver for all computations involving the inverse of the covariance matrix.

- RealMatrix gradCovVector

  A matrix, where each column is the derivative of the covVector with respect to a particular componenet of X.

- RealMatrix normTrainPointsAll

  Set of all original samples available.

- RealMatrix trainValuesAll

  All original samples available.

- RealMatrix trendFunctionAll

  Trend function values corresponding to all original samples.

- RealMatrix Rinv_YFb

  Matrix for storing inverse of correlation matrix Rinv∗(Y-FB)

- size_t numObs

  The number of observations on which the GP surface is built.

- size_t numObsAll

  The original number of observations.

- short trendOrder

  The number of variables in each X variable (number of dimensions of the problem).

- RealVector thetaParams

Theta is the vector of covariance parameters for the GP. We determine the values of theta by optimization Currently, the covariance function is theta[0]∗exp(-0.5∗sume)+delta∗pow(sige,2). sume is the sum squared of weighted distances; it involves a sum of theta[1](Xi(1)-Xj(1))^2 + theta[2](Xi(2)-Xj(2))^2 + ... where Xi(1) is the first dimension value of multi-dimensional variable Xi. delta∗pow(sige,2) is a jitter term used to improve matrix computations. delta is zero for the covariance between different points and 1 for the covariance between the same point. sige is the underlying process error.

- Real procVar

  The process variance, the multiplier of the correlation matrix.

- IntArray pointsAddedIndex

  Used by the point selection algorithm, this vector keeps track all points which have been added.

- int cholFlag

  A global indicator for success of the Cholesky factorization.

- bool usePointSelection

  a flag to indicate the use of point selection

Static Private Attributes

- static GaussProcApproximation ∗ GPinstance

  pointer to the active object instance used within the static evaluator

Additional Inherited Members

13.47.1 Detailed Description

Derived approximation class for Gaussian Process implementation.

The GaussProcApproximation class provides a global approximation (surrogate) based on a Gaussian process. The Gaussian process is built after normalizing the function values, with zero mean. Opt++ is used to determine the optimal values of the covariance parameters, those which minimize the negative log likelihood function.

13.47.2 Constructor & Destructor Documentation

GaussProcApproximation ( ) `[inline]`

default constructor

alternate constructor used by EffGlobalOptimization and NonDGlobalReliability that does not use a problem database defaults here are no point selectinn and quadratic trend function.

13.47.3 Member Function Documentation

void GPmodel_apply (const RealVector & *new_x*, bool *variance_flag*, bool *gradients_flag*) `[private]`

Function returns a response value using the GP surface.

The response value is computed at the design point specified by the RealVector function argument.

References Dakota::abort_handler(), GaussProcApproximation::approxPoint, GaussProcApproximation::get_cov_vector(), SharedApproxData::numVars, GaussProcApproximation::predict(), Approximation::sharedDataRep, GaussProcApproximation::trainMeans, and GaussProcApproximation::trainStdvs.

Referenced by GaussProcApproximation::gradient(), GaussProcApproximation::pointsel_get_errors(), GaussProcApproximation::prediction_variance(), and GaussProcApproximation::value().

13.47.4 Member Data Documentation

short trendOrder `[private]`

The number of variables in each X variable (number of dimensions of the problem).

The order of the basis function for the mean of the GP If the order = 0, the trend is a constant, if the order = 1, trend is linear, if order = 2, trend is quadratic.

Referenced by GaussProcApproximation::GaussProcApproximation(), GaussProcApproximation::get_beta_coefficients(), GaussProcApproximation::get_trend(), GaussProcApproximation::GPmodel_build(), and GaussProcApproximation::predict().

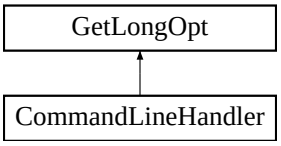
The documentation for this class was generated from the following files:

- GaussProcApproximation.hpp
- GaussProcApproximation.cpp

13.48 GetLongOpt Class Reference

GetLongOpt is a general command line utility from S. Manoharan (Advanced Computer Research Institute, Lyon, France).

Inheritance diagram for GetLongOpt:



Public Types

- enum OptType { Valueless, OptionalValue, MandatoryValue }

  enum for different types of values associated with command line options.

Public Member Functions

- GetLongOpt (const char optmark= '-')

  Constructor.
- ~GetLongOpt ()

  Destructor.
- int parse (int argc, char *const *argv)

  parse the command line args (argc, argv).
- int parse (char *const str, char *const p)

  parse a string of options (typically given from the environment).
- int enroll (const char *const opt, const OptType t, const char *const desc, const char *const val)

  Add an option to the list of valid command options.
- const char * retrieve (const char *const opt) const

  Retrieve value of option.
- void usage (std::ostream &outfile=Cout) const

Print usage information to outfile.

- void usage (const char *str)

  Change header of usage output to str.
- void store (const char *name, const char *value)

  Store a specified option value.

Private Member Functions

- char * basename (char *const p) const

  extract the base name from a string as delimited by '/'
- int setcell (Cell *c, char *valtoken, char *nexttoken, const char *p)

  internal convenience function for setting Cell::value

Private Attributes

- Cell * table

  option table
- const char * ustring

  usage message
- char * pname

  program basename
- char optmarker

  option marker
- int enroll_done

  finished enrolling
- Cell * last

  last entry in option table

13.48.1 Detailed Description

GetLongOpt is a general command line utility from S. Manoharan (Advanced Computer Research Institute, Lyon, France).

GetLongOpt manages the definition and parsing of "long options." Command line options can be abbreviated as long as there is no ambiguity. If an option requires a value, the value should be separated from the option either by whitespace or an "=".

13.48.2 Member Enumeration Documentation

enum OptType

enum for different types of values associated with command line options.

Enumerator

Valueless option that may never have a value

OptionalValue option with optional value

MandatoryValue option with required value

13.48.3 Constructor & Destructor Documentation

GetLongOpt (const char *optmark* = '-')

Constructor.

Constructor for GetLongOpt takes an optional argument: the option marker. If unspecified, this defaults to '-', the standard (?) Unix option marker.

References GetLongOpt::enroll_done, GetLongOpt::last, GetLongOpt::optmarker, GetLongOpt::table, and GetLongOpt::ustring.

13.48.4 Member Function Documentation

int parse (int *argc*, char *const * *argv*)

parse the command line args (argc, argv).

A return value < 1 represents a parse error. Appropriate error messages are printed when errors are seen. parse returns the the optind (see getopt(3)) if parsing is successful.

References GetLongOpt::basename(), GetLongOpt::enroll_done, GetLongOpt::optmarker, GetLongOpt::pname, GetLongOpt::setcell(), and GetLongOpt::table.

Referenced by CommandLineHandler::check_usage().

int parse (char *const *str*, char *const *p*)

parse a string of options (typically given from the environment).

A return value < 1 represents a parse error. Appropriate error messages are printed when errors are seen. parse takes two strings: the first one is the string to be parsed and the second one is a string to be prefixed to the parse errors.

References GetLongOpt::enroll_done, GetLongOpt::optmarker, GetLongOpt::setcell(), and GetLongOpt::table.

int enroll (const char *const *opt*, const OptType *t*, const char *const *desc*, const char *const *val*)

Add an option to the list of valid command options.

enroll adds option specifications to its internal database. The first argument is the option sting. The second is an enum saying if the option is a flag (Valueless), if it requires a mandatory value (MandatoryValue) or if it takes an optional value (OptionalValue). The third argument is a string giving a brief description of the option. This description will be used by GetLongOpt::usage. GetLongOpt, for usage-printing, uses {$val} to represent values needed by the options. {<$val>} is a mandatory value and {[$val]} is an optional value. The final argument to enroll is the default string to be returned if the option is not specified. For flags (options with Valueless), use "" (empty string, or in fact any arbitrary string) for specifying TRUE and 0 (null pointer) to specify FALSE.

References GetLongOpt::enroll_done, GetLongOpt::last, and GetLongOpt::table.

Referenced by CommandLineHandler::initialize_options().

const char * retrieve (const char *const *opt*) const

Retrieve value of option.

The values of the options that are enrolled in the database can be retrieved using retrieve. This returns a string and this string should be converted to whatever type you want. See atoi, atof, atol, etc. If a "parse" is not done before retrieving all you will get are the default values you gave while enrolling! Ambiguities while retrieving (may happen when options are abbreviated) are resolved by taking the matching option that was enrolled last. For example, -{v} will expand to {-verify}. If you try to retrieve something you didn't enroll, you will get a warning message.

References GetLongOpt::optmarker, and GetLongOpt::table.

Referenced by CommandLineHandler::check_usage(), ProgramOptions::manage_run_modes(), ProgramOptions::ProgramOptions(), and CommandLineHandler::read_restart_evals().

void usage (const char * *str*) [inline]

Change header of usage output to str.

GetLongOpt::usage is overloaded. If passed a string "str", it sets the internal usage string to "str". Otherwise it simply prints the command usage.

References GetLongOpt::ustring.

The documentation for this class was generated from the following files:

- CommandLineHandler.hpp
- CommandLineHandler.cpp

13.49 Graphics Class Reference

The Graphics class provides a single interface to 2D (motif) and 3D (PLPLOT) graphics; there is only one instance of this OutputManager::dakotaGraphics.

Public Member Functions

- Graphics ()

  constructor
- ~Graphics ()

  destructor
- void create_plots_2d (const Variables &vars, const Response &response)

  creates the 2d graphics window and initializes the plots
- void add_datapoint (int graphics_cntr, const Variables &vars, const Response &response)

  adds data to each window in the 2d graphics based on the results of a model evaluation
- void add_datapoint (int i, double x, double y)

  adds data to a single window in the 2d graphics
- void new_dataset (int i)

  creates a separate line graphic for subsequent data points for a single window in the 2d graphics
- void close ()

  close graphics windows
- void set_x_labels2d (const char *x_label)

  set x label for each plot equal to x_label
- void set_y_labels2d (const char *y_label)

  set y label for each plot equal to y_label
- void set_x_label2d (int i, const char *x_label)

  set x label for ith plot equal to x_label
- void set_y_label2d (int i, const char *y_label)

  set y label for ith plot equal to y_label

Private Attributes

- Graphics2D * graphics2D

  pointer to the 2D graphics object
- bool win2dOn

  flag to indicate if 2D graphics window is active

13.49.1 Detailed Description

The Graphics class provides a single interface to 2D (motif) and 3D (PLPLOT) graphics; there is only one instance of this OutputManager::dakotaGraphics.

13.49.2 Member Function Documentation

void create_plots_2d (const Variables & *vars,* const Response & *response*)

creates the 2d graphics window and initializes the plots

Sets up a single event loop for duration of the dakotaGraphics object, continuously adding data to a single window. There is no reset. To start over with a new data set, you need a new object (delete old and instantiate new).

References Variables::continuous_variable_labels(), Variables::cv(), Variables::discrete_int_variable_labels(), Variables::discrete_real_variable_labels(), Variables::div(), Variables::drv(), Response::function_labels(), Graphics::graphics2D, Response::num_functions(), Dakota::re_match(), and Graphics::win2dOn.

Referenced by NonDReliability::initialize_graphics(), SurrBasedMinimizer::initialize_graphics(), and Iterator::initialize_graphics().

void add_datapoint (int *graphics_cntr,* const Variables & *vars,* const Response & *response*)

adds data to each window in the 2d graphics based on the results of a model evaluation

Adds data to each 2d plot and each tabular data column (one for each active variable and for each response function). graphicsCntr is used for the x axis in the graphics and the first column in the tabular data.

References Response::active_set_request_vector(), Variables::continuous_variables(), Variables::discrete_int_variables(), Variables::discrete_real_variables(), Response::function_values(), Graphics::graphics2D, and Graphics::win2dOn.

Referenced by OutputManager::add_datapoint(), NonDLocalReliability::mean_value(), and NonDLocalReliability::update_level_data().

void add_datapoint (int *i,* double *x,* double *y*)

adds data to a single window in the 2d graphics

Adds data to a single 2d plot. Allows complete flexibility in defining other kinds of x-y plotting in the 2D graphics.

References Graphics::graphics2D, and Graphics::win2dOn.

void new_dataset (int *i*)

creates a separate line graphic for subsequent data points for a single window in the 2d graphics

Used for displaying multiple data sets within the same plot.

References Graphics::graphics2D, and Graphics::win2dOn.

Referenced by NonDLocalReliability::update_level_data().

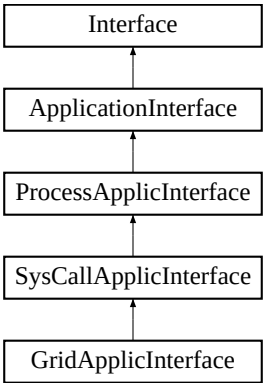
The documentation for this class was generated from the following files:

- DakotaGraphics.hpp
- DakotaGraphics.cpp

13.50 GridApplicInterface Class Reference

Derived application interface class which spawns simulation codes using grid services such as Condor or Globus.

Inheritance diagram for GridApplicInterface:



Public Member Functions

- GridApplicInterface (const ProblemDescDB &problem_db)

  constructor
- ~GridApplicInterface ()

  destructor
- void derived_map (const Variables &vars, const ActiveSet &set, Response &response, int fn_eval_id)

  Called by map() and other functions to execute the simulation in synchronous mode. The portion of performing an evaluation that is specific to a derived class.
- void derived_map_asynch (const ParamResponsePair &pair)

  Called by map() and other functions to execute the simulation in asynchronous mode. The portion of performing an asynchronous evaluation that is specific to a derived class.
- void wait_local_evaluations (PRPQueue &prp_queue)

  For asynchronous function evaluations, this method is used to detect completion of jobs and process their results. It provides the processing code that is specific to derived classes. This version waits for at least one completion.
- void test_local_evaluations (PRPQueue &prp_queue)

  Convenience function for common code between wait and nowait case.
- int synchronous_local_analysis (int analysis_id)

Protected Member Functions

- bool grid_file_test (const String &root_file)

  test file(s) for existence based on root_file name

Protected Attributes

- IntSet idSet

  Set of function evaluation id's for active asynchronous system call evaluations.
- IntShortMap failCountMap

  map linking function evaluation id's to number of response read failures
- start_grid_computing_t start_grid_computing

  handle to dynamically linked start_grid_computing function
- perform_analysis_t perform_analysis

  handle to dynamically linked perform_analysis grid function
- get_jobs_completed_t get_jobs_completed

  handle to dynamically linked get_jobs_completed grid function
- stop_grid_computing_t stop_grid_computing

  handle to dynamically linked stop_grid_computing function

13.50.1 Detailed Description

Derived application interface class which spawns simulation codes using grid services such as Condor or Globus.

This class is currently a modified copy of SysCallApplicInterface adapted for use with an external grid dervices library which was dynamically linked using dlopen() services.

13.50.2 Member Function Documentation

int synchronous_local_analysis (int *analysis_id*) `[inline], [virtual]`

This code provides the derived function used by ApplicationInterface::serve_analyses_synch(). TODO - allow local analyses?????

Reimplemented from ApplicationInterface.

References SysCallApplicInterface::spawn_analysis_to_shell().

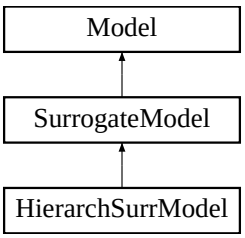
The documentation for this class was generated from the following files:

- GridApplicInterface.hpp
- GridApplicInterface.cpp

13.51 HierarchSurrModel Class Reference

Derived model class within the surrogate model branch for managing hierarchical surrogates (models of varying fidelity).

Inheritance diagram for HierarchSurrModel:



Public Member Functions

- HierarchSurrModel (ProblemDescDB &problem_db)

  constructor
- ~HierarchSurrModel ()

  destructor

Protected Member Functions

- void derived_compute_response (const ActiveSet &set)

  portion of compute_response() specific to HierarchSurrModel
- void derived_asynch_compute_response (const ActiveSet &set)

  portion of asynch_compute_response() specific to HierarchSurrModel
- const IntResponseMap & derived_synchronize ()

  portion of synchronize() specific to HierarchSurrModel
- const IntResponseMap & derived_synchronize_nowait ()

  portion of synchronize_nowait() specific to HierarchSurrModel
- Model & surrogate_model ()

  return lowFidelityModel
- Model & truth_model ()

  return highFidelityModel
- void derived_subordinate_models (ModelList &ml, bool recurse_flag)

  return lowFidelityModel and highFidelityModel
- void primary_response_fn_weights (const RealVector &wts, bool recurse_flag=true)

  set the relative weightings for multiple objective functions or least squares terms and optionally recurses into LF/HF models
- void surrogate_response_mode (short mode)

  set responseMode and pass any bypass request on to highFidelityModel for any lower-level surrogate recursions.
- void surrogate_function_indices (const IntSet &surr_fn_indices)

  (re)set the surrogate index set in SurrogateModel::surrogateFnIndices
- void build_approximation ()

  use highFidelityModel to compute the truth values needed for correction of lowFidelityModel results
- void component_parallel_mode (short mode)

  update component parallel mode for supporting parallelism in lowFidelityModel and highFidelityModel
- void derived_init_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set up lowFidelityModel and highFidelityModel for parallel operations
- void derived_init_serial ()

  set up lowFidelityModel and highFidelityModel for serial operations.
- void derived_set_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set active parallel configuration within lowFidelityModel and highFidelityModel
- void derived_free_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  deallocate communicator partitions for the HierarchSurrModel (request forwarded to lowFidelityModel and highFidelityModel)
- void serve_run (ParLevLIter pl_iter, int max_eval_concurrency)

Service lowFidelityModel and highFidelityModel job requests received from the master. Completes when a termination message is received from stop_servers().

- void stop_servers ()

  Executed by the master to terminate lowFidelityModel and highFidelityModel server operations when iteration on the HierarchSurrModel is complete.

- void inactive_view (short view, bool recurse_flag=true)

  update the Model's inactive view based on higher level (nested) context and optionally recurse into

- int evaluation_id () const

  Return the current evaluation id for the HierarchSurrModel.

- void set_evaluation_reference ()

  set the evaluation counter reference points for the HierarchSurrModel (request forwarded to lowFidelityModel and highFidelityModel)

- void fine_grained_evaluation_counters ()

  request fine-grained evaluation reporting within lowFidelityModel and highFidelityModel

- void print_evaluation_summary (std::ostream &s, bool minimal_header=false, bool relative_count=true) const

  print the evaluation summary for the HierarchSurrModel (request forwarded to lowFidelityModel and highFidelityModel)

- void eval_tag_prefix (const String &eval_id_str)

  set the hierarchical eval ID tag prefix

Private Member Functions

- void update_model (Model &model)

  update the incoming model (lowFidelityModel or highFidelityModel) with current variable values/bounds/labels

Private Attributes

- int hierModelEvalCntr

  number of calls to derived_compute_response()/ derived_asynch_compute_response()

- IntResponseMap cachedTruthRespMap

  map of high-fidelity responses retrieved in derived_synchronize_nowait() that could not be returned since corresponding low-fidelity response portions were still pending.

- Model lowFidelityModel

  provides approximate low fidelity function evaluations. Model is of arbitrary type and supports recursions (e.g., lowFidelityModel can be a data fit surrogate on a low fidelity model).

- Model highFidelityModel

  provides truth evaluations for computing corrections to the low fidelity results. Model is of arbitrary type and supports recursions.

- Response highFidRefResponse

  the reference high fidelity response computed in build_approximation() and used for calculating corrections.

- String evalTagPrefix

  cached evalTag Prefix from parents to use at compute_response time

Additional Inherited Members

13.51.1 Detailed Description

Derived model class within the surrogate model branch for managing hierarchical surrogates (models of varying fidelity).

The HierarchSurrModel class manages hierarchical models of varying fidelity. In particular, it uses a low fidelity model as a surrogate for a high fidelity model. The class contains a lowFidelityModel which performs the approximate low fidelity function evaluations and a highFidelityModel which provides truth evaluations for computing corrections to the low fidelity results.

13.51.2 Member Function Documentation

void derived_compute_response (const ActiveSet & *set*) [protected], [virtual]

portion of compute_response() specific to HierarchSurrModel

Compute the response synchronously using lowFidelityModel, highFidelityModel, or both (mixed case). For the lowFidelityModel portion, compute the high fidelity response if needed with build_approximation(), and, if correction is active, correct the low fidelity results.

Reimplemented from Model.

References Response::active_set(), DiscrepancyCorrection::apply(), SurrogateModel::approxBuilds, SurrogateModel::asv_mapping(), HierarchSurrModel::build_approximation(), HierarchSurrModel::component_parallel_mode(), DiscrepancyCorrection::compute(), Model::compute_response(), DiscrepancyCorrection::computed(), Response::copy(), Model::current_response(), Model::currentResponse, Model::currentVariables, SurrogateModel::deltaCorr, Model::eval_tag_prefix(), HierarchSurrModel::evalTagPrefix, SurrogateModel::force_rebuild(), Model::hierarchicalTagging, HierarchSurrModel::hierModelEvalCntr, HierarchSurrModel::highFidelityModel, HierarchSurrModel::highFidRefResponse, HierarchSurrModel::lowFidelityModel, Model::outputLevel, ActiveSet::request_vector(), SurrogateModel::response_mapping(), SurrogateModel::responseMode, Response::update(), and HierarchSurrModel::update_model().

void derived_asynch_compute_response (const ActiveSet & *set*) [protected], [virtual]

portion of asynch_compute_response() specific to HierarchSurrModel

Compute the response asynchronously using lowFidelityModel, highFidelityModel, or both (mixed case). For the lowFidelityModel portion, compute the high fidelity response with build_approximation() (for correcting the low fidelity results in derived_synchronize() and derived_synchronize_nowait()) if not performed previously.

Reimplemented from Model.

References DiscrepancyCorrection::apply(), SurrogateModel::approxBuilds, SurrogateModel::asv_mapping(), Model::asynch_compute_response(), Model::asynch_flag(), HierarchSurrModel::build_approximation(), SurrogateModel::cachedApproxRespMap, HierarchSurrModel::cachedTruthRespMap, HierarchSurrModel::component_parallel_mode(), DiscrepancyCorrection::compute(), Model::compute_response(), DiscrepancyCorrection::computed(), Response::copy(), Variables::copy(), Model::current_response(), Model::currentVariables, SurrogateModel::deltaCorr, ActiveSet::derivative_vector(), Model::eval_tag_prefix(), HierarchSurrModel::evalTagPrefix, Model::evaluation_id(), SurrogateModel::force_rebuild(), Model::hierarchicalTagging, HierarchSurrModel::hierModelEvalCntr, HierarchSurrModel::highFidelityModel, HierarchSurrModel::highFidRefResponse, HierarchSurrModel::lowFidelityModel, Model::outputLevel, SurrogateModel::rawVarsMap, ActiveSet::request_vector(), SurrogateModel::responseMode, SurrogateModel::surrIdMap, SurrogateModel::truthIdMap, and HierarchSurrModel::update_model().

const IntResponseMap & derived_synchronize ( ) [protected], [virtual]

portion of synchronize() specific to HierarchSurrModel

Blocking retrieval of asynchronous evaluations from lowFidelityModel, highFidelityModel, or both (mixed case). For the lowFidelityModel portion, apply correction (if active) to each response in the array. derived_synchronize() is designed for the general case where derived_asynch_compute_response() may be inconsistent in its use of low fidelity evaluations, high fidelity evaluations, or both.

Reimplemented from Model.

References DiscrepancyCorrection::apply(), SurrogateModel::cachedApproxRespMap, HierarchSurrModel::cachedTruthRespMap, HierarchSurrModel::component_parallel_mode(), DiscrepancyCorrection::compute(), DiscrepancyCorrection::computed(), SurrogateModel::deltaCorr, HierarchSurrModel::derived_synchronize_nowait(), HierarchSurrModel::highFidelityModel, HierarchSurrModel::highFidRefResponse, HierarchSurrModel::lowFidelityModel, Model::outputLevel, SurrogateModel::rawVarsMap, SurrogateModel::response_mapping(), SurrogateModel::responseMode, SurrogateModel::surrIdMap, SurrogateModel::surrResponseMap, Model::synchronize(), and SurrogateModel::truthIdMap.

const IntResponseMap & derived_synchronize_nowait ( ) `[protected], [virtual]`

portion of synchronize_nowait() specific to HierarchSurrModel

Nonblocking retrieval of asynchronous evaluations from lowFidelityModel, highFidelityModel, or both (mixed case). For the lowFidelityModel portion, apply correction (if active) to each response in the map. derived_synchronize_nowait() is designed for the general case where derived_asynch_compute_response() may be inconsistent in its use of actual evals, approx evals, or both.

Reimplemented from Model.

References DiscrepancyCorrection::apply(), SurrogateModel::cachedApproxRespMap, HierarchSurrModel::cachedTruthRespMap, HierarchSurrModel::component_parallel_mode(), DiscrepancyCorrection::compute(), DiscrepancyCorrection::computed(), SurrogateModel::deltaCorr, HierarchSurrModel::highFidelityModel, HierarchSurrModel::highFidRefResponse, HierarchSurrModel::lowFidelityModel, Model::outputLevel, SurrogateModel::rawVarsMap, SurrogateModel::response_mapping(), SurrogateModel::responseMode, SurrogateModel::surrIdMap, SurrogateModel::surrResponseMap, Model::synchronize_nowait(), and SurrogateModel::truthIdMap.

Referenced by HierarchSurrModel::derived_synchronize().

int evaluation_id ( ) const `[inline], [protected], [virtual]`

Return the current evaluation id for the HierarchSurrModel.

return the hierarchical model evaluation count. Due to possibly intermittent use of surrogate bypass, this is not the same as either the loFi or hiFi model evaluation counts. It also does not distinguish duplicate evals.

Reimplemented from Model.

References HierarchSurrModel::hierModelEvalCntr.

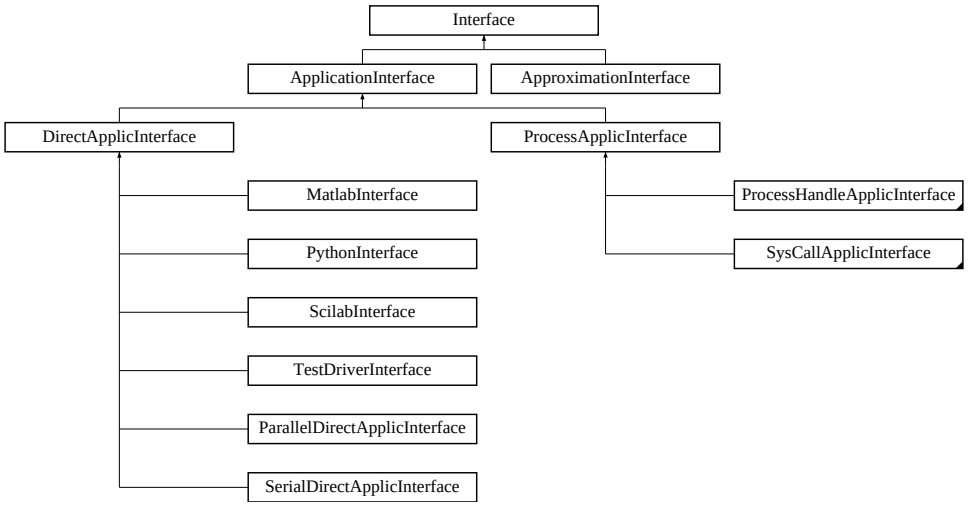
The documentation for this class was generated from the following files:

- HierarchSurrModel.hpp
- HierarchSurrModel.cpp

13.52 Interface Class Reference

Base class for the interface class hierarchy.

Inheritance diagram for Interface:



Public Member Functions

- Interface ()

  default constructor
- Interface (ProblemDescDB &problem_db)

  standard constructor for envelope
- Interface (const Interface &interface_in)

  copy constructor
- virtual ~Interface ()

  destructor
- Interface operator= (const Interface &interface_in)

  assignment operator
- virtual void map (const Variables &vars, const ActiveSet &set, Response &response, bool asynch_flag=false)

  the function evaluator: provides a "mapping" from the variables to the responses.
- virtual const IntResponseMap & synch ()

  recovers data from a series of asynchronous evaluations (blocking)
- virtual const IntResponseMap & synch_nowait ()

  recovers data from a series of asynchronous evaluations (nonblocking)
- virtual void serve_evaluations ()

  evaluation server function for multiprocessor executions
- virtual void stop_evaluation_servers ()

  send messages from iterator rank 0 to terminate evaluation servers
- virtual void init_communicators (const IntArray &message_lengths, int max_eval_concurrency)

  allocate communicator partitions for concurrent evaluations within an iterator and concurrent multiprocessor analyses within an evaluation.
- virtual void set_communicators (const IntArray &message_lengths, int max_eval_concurrency)

  set the local parallel partition data for an interface (the partitions are already allocated in ParallelLibrary).
- virtual void free_communicators ()

deallocate communicator partitions for concurrent evaluations within an iterator and concurrent multiprocessor analyses within an evaluation.

- virtual void init_serial ()

  reset certain defaults for serial interface objects.
- virtual int asynch_local_evaluation_concurrency () const

  return the user-specified concurrency for asynch local evaluations
- virtual short interface_synchronization () const

  return the user-specified interface synchronization
- virtual int minimum_points (bool constraint_flag) const

  returns the minimum number of points required to build a particular ApproximationInterface (used by DataFitSurrModels).
- virtual int recommended_points (bool constraint_flag) const

  returns the recommended number of points required to build a particular ApproximationInterface (used by DataFitSurrModels).
- virtual void approximation_function_indices (const IntSet &approx_fn_indices)

  set the (currently active) approximation function index set
- virtual void update_approximation (const Variables &vars, const IntResponsePair &response_pr)

  updates the anchor point for an approximation
- virtual void update_approximation (const RealMatrix &samples, const IntResponseMap &resp_map)

  updates the current data points for an approximation
- virtual void update_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map)

  updates the current data points for an approximation
- virtual void append_approximation (const Variables &vars, const IntResponsePair &response_pr)

  appends a single point to an existing approximation
- virtual void append_approximation (const RealMatrix &samples, const IntResponseMap &resp_map)

  appends multiple points to an existing approximation
- virtual void append_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map)

  appends multiple points to an existing approximation
- virtual void build_approximation (const RealVector &c_l_bnds, const RealVector &c_u_bnds, const IntVector &di_l_bnds, const IntVector &di_u_bnds, const RealVector &dr_l_bnds, const RealVector &dr_u_bnds)

  builds the approximation
- virtual void rebuild_approximation (const BoolDeque &rebuild_deque)

  rebuilds the approximation after a data update
- virtual void pop_approximation (bool save_surr_data)

  removes data from last append from the approximation
- virtual void restore_approximation ()

  restores the approximation to a selected previous state
- virtual bool restore_available ()

  queries the approximation for the ability to restore a previous increment
- virtual void finalize_approximation ()

  finalizes the approximation by applying all trial increments
- virtual void store_approximation ()

  move the current approximation into storage for later combination

- virtual void combine_approximation (short corr_type)

  combine the current approximation with one previously stored
- virtual void clear_current ()

  clears current data from an approximation interface
- virtual void clear_all ()

  clears all data from an approximation interface
- virtual void clear_saved ()

  clears saved data (from pop invocations) from an approximation interface
- virtual SharedApproxData & shared_approximation ()

  retrieve the SharedApproxData within an ApproximationInterface
- virtual std::vector
  < Approximation > & approximations ()

  retrieve the Approximations within an ApproximationInterface
- virtual const
  Pecos::SurrogateData & approximation_data (size_t index)

  retrieve the approximation data from a particular Approximation within an ApproximationInterface
- virtual const RealVectorArray & approximation_coefficients (bool normalized=false)

  retrieve the approximation coefficients from each Approximation within an ApproximationInterface
- virtual void approximation_coefficients (const RealVectorArray &approx_coeffs, bool normalized=false)

  set the approximation coefficients within each Approximation within an ApproximationInterface
- virtual const RealVector & approximation_variances (const Variables &vars)

  retrieve the approximation variances from each Approximation within an ApproximationInterface
- virtual const StringArray & analysis_drivers () const

  retrieve the analysis drivers specification for application interfaces
- virtual bool evaluation_cache () const

  return flag indicating usage of the global evaluation cache
- virtual void file_cleanup () const

  clean up any interface parameter/response files when aborting
- void assign_rep (Interface *interface_rep, bool ref_count_incr=true)

  replaces existing letter with a new one
- unsigned short interface_type () const

  returns the interface type
- const String & interface_id () const

  returns the interface identifier
- int evaluation_id () const

  returns the value of the (total) evaluation id counter for the interface
- void fine_grained_evaluation_counters (size_t num_fns)

  set fineGrainEvalCounters to true and initialize counters if needed
- void init_evaluation_counters (size_t num_fns)

  initialize fine grained evaluation counters
- void set_evaluation_reference ()

  set evaluation count reference points for the interface
- void print_evaluation_summary (std::ostream &s, bool minimal_header, bool relative_count) const

print an evaluation summary for the interface

- bool multi_proc_eval () const

  returns a flag signaling the use of multiprocessor evaluation partitions
- bool iterator_eval_dedicated_master () const

  returns a flag signaling the use of a dedicated master processor at the iterator-evaluation scheduling level
- bool is_null () const

  function to check interfaceRep (does this envelope contain a letter?)
- void eval_tag_prefix (const String &eval_id_str, bool append_iface_id=true)

  set the evaluation tag prefix (does not recurse)

Protected Member Functions

- Interface (BaseConstructor, const ProblemDescDB &problem_db)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
- Interface (NoDBBaseConstructor, size_t num_fns, short output_level)

  constructor initializes the base class part of letter classes (NoDBBaseConstructor used for on the fly instantiations without a DB)
- void init_algebraic_mappings (const Variables &vars, const Response &response)

  Define algebraicACVIndices, algebraicACVIds, and algebraicFnIndices.
- void asv_mapping (const ActiveSet &total_set, ActiveSet &algebraic_set, ActiveSet &core_set)

  define the evaluation requirements for algebraic_mappings() (algebraic_set) and the core Application/Approximation mapping (core_set) from the total Interface evaluation requirements (total_set)
- void asv_mapping (const ActiveSet &algebraic_set, ActiveSet &total_set)

  map an algebraic ASV back to original total ordering for asynch recovery
- void algebraic_mappings (const Variables &vars, const ActiveSet &algebraic_set, Response &algebraic_response)

  evaluate the algebraic_response using the AMPL solver library and the data extracted from the algebraic_mappings file
- void response_mapping (const Response &algebraic_response, const Response &core_response, Response &total_response)

  combine the response from algebraic_mappings() with the response from derived_map() to create the total response
- String final_eval_id_tag (int fn_eval_id)

  form and return the final evaluation ID tag, appending iface ID if needed

Protected Attributes

- unsigned short interfaceType

  the interface type: enum for system, fork, direct, grid, or approximation
- String interfaceId

  the interface specification identifier string from the DAKOTA input file
- bool algebraicMappings

  flag for the presence of algebraic_mappings that define the subset of an Interface's parameter to response mapping that is explicit and algebraic.
- bool coreMappings

flag for the presence of non-algebraic mappings that define the core of an Interface's parameter to response mapping (using analysis_drivers for ApplicationInterface or functionSurfaces for ApproximationInterface).

- int currEvalId

  identifier for the current evaluation, which may differ from the evaluation counters in the case of evaluation scheduling; used on iterator master as well as server processors. Currently, this is set prior to all invocations of derived_map() for all processors.

- bool fineGrainEvalCounters

  controls use of fn val/grad/hess counters

- int evalIdCntr

  total interface evaluation counter

- int newEvalIdCntr

  new (non-duplicate) interface evaluation counter

- int evalIdRefPt

  iteration reference point for evalIdCntr

- int newEvalIdRefPt

  iteration reference point for newEvalIdCntr

- IntArray fnValCounter

  number of value evaluations by resp fn

- IntArray fnGradCounter

  number of gradient evaluations by resp fn

- IntArray fnHessCounter

  number of Hessian evaluations by resp fn

- IntArray newFnValCounter

  number of new value evaluations by resp fn

- IntArray newFnGradCounter

  number of new gradient evaluations by resp fn

- IntArray newFnHessCounter

  number of new Hessian evaluations by resp fn

- IntArray fnValRefPt

  iteration reference point for fnValCounter

- IntArray fnGradRefPt

  iteration reference point for fnGradCounter

- IntArray fnHessRefPt

  iteration reference point for fnHessCounter

- IntArray newFnValRefPt

  iteration reference point for newFnValCounter

- IntArray newFnGradRefPt

  iteration reference point for newFnGradCounter

- IntArray newFnHessRefPt

  iteration reference point for newFnHessCounter

- IntResponseMap rawResponseMap

  Set of responses returned after either a blocking or nonblocking schedule of asynchronous evaluations.

- StringArray fnLabels

response function descriptors from the DAKOTA input file (used in print_evaluation_summary() and derived direct interface classes)

- bool multiProcEvalFlag

  flag for multiprocessor evaluation partitions (evalComm)
- bool ieDedMasterFlag

  flag for dedicated master partitioning at the iterator level
- short outputLevel

  output verbosity level: {SILENT,QUIET,NORMAL,VERBOSE,DEBUG}_OUTPUT
- String evalTagPrefix

  set of period-delimited evaluation ID tags to use in evaluation tagging
- bool appendIfaceId

  whether to append the interface ID to the prefix during map (default true)

Private Member Functions

- Interface * get_interface (ProblemDescDB &problem_db)

  Used by the envelope to instantiate the correct letter class.
- int algebraic_function_type (String)

  Used by algebraic mappings to determine the correct AMPL function evaluation call to make.

Private Attributes

- StringArray algebraicVarTags

  set of variable tags from AMPL stub.col
- SizetArray algebraicACVIndices

  set of indices mapping AMPL algebraic variables to DAKOTA all continuous variables
- SizetArray algebraicACVIds

  set of ids mapping AMPL algebraic variables to DAKOTA all continuous variables
- StringArray algebraicFnTags

  set of function tags from AMPL stub.row
- IntArray algebraicFnTypes

  function type: > 0 = objective, < 0 = constraint |value|-1 is the objective (constraint) index when making AMPL objval (conival) calls
- SizetArray algebraicFnIndices

  set of indices mapping AMPL algebraic objective functions to DAKOTA response functions
- RealArray algebraicConstraintWeights

  set of weights for computing Hessian matrices for algebraic constraints;
- int numAlgebraicResponses

  number of algebraic responses (objectives+constraints)
- Interface * interfaceRep

  pointer to the letter (initialized only for the envelope)
- int referenceCount

  number of objects sharing interfaceRep
- ASL * asl

  pointer to an AMPL solver library (ASL) object

13.52.1 Detailed Description

Base class for the interface class hierarchy.

The Interface class hierarchy provides the part of a Model that is responsible for mapping a set of Variables into a set of Responses. The mapping is performed using either a simulation-based application interface or a surrogate-based approximation interface. For memory efficiency and enhanced polymorphism, the interface hierarchy employs the "letter/envelope idiom" (see Coplien "Advanced C++", p. 133), for which the base class (Interface) serves as the envelope and one of the derived classes (selected in Interface::get_interface()) serves as the letter.

13.52.2 Constructor & Destructor Documentation

Interface ( )

default constructor

used in Model envelope class instantiations

Interface (ProblemDescDB & *problem_db*)

standard constructor for envelope

Used in Model instantiation to build the envelope. This constructor only needs to extract enough data to properly execute get_interface, since Interface::Interface(BaseConstructor, problem_db) builds the actual base class data inherited by the derived interfaces.

References Dakota::abort_handler(), Interface::get_interface(), and Interface::interfaceRep.

Interface (const Interface & *interface_in*)

copy constructor

Copy constructor manages sharing of interfaceRep and incrementing of referenceCount.

References Interface::interfaceRep, and Interface::referenceCount.

~Interface ( ) `[virtual]`

destructor

Destructor decrements referenceCount and only deletes interfaceRep if referenceCount is zero.

References Interface::interfaceRep, and Interface::referenceCount.

Interface (BaseConstructor , const ProblemDescDB & *problem_db*) `[protected]`

constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor is the one which must build the base class data for all inherited interfaces. get_interface() instantiates a derived class letter and the derived constructor selects this base class constructor in its initialization list (to avoid the recursion of the base class constructor calling get_interface() again). Since this is the letter and the letter IS the representation, interfaceRep is set to NULL (an uninitialized pointer causes problems in ~Interface).

References Dakota::abort_handler(), Interface::algebraic_function_type(), Interface::algebraicConstraintWeights, Interface::algebraicFnTags, Interface::algebraicFnTypes, Interface::algebraicMappings, Interface::algebraicVarTags, Interface::asl, Interface::fineGrainEvalCounters, Interface::fnLabels, ProblemDescDB::get_sa(), ProblemDescDB::get_string(), Interface::init_evaluation_counters(), Interface::outputLevel, and Dakota::strends().

13.52.3 Member Function Documentation

Interface operator= (const Interface & *interface_in*)

assignment operator

Assignment operator decrements referenceCount for old interfaceRep, assigns new interfaceRep, and increments referenceCount for new interfaceRep.

References Interface::interfaceRep, and Interface::referenceCount.

void assign_rep (Interface * *interface_rep,* bool *ref_count_incr* = `true`)

replaces existing letter with a new one

Similar to the assignment operator, the assign_rep() function decrements referenceCount for the old interfaceRep and assigns the new interfaceRep. It is different in that it is used for publishing derived class letters to existing envelopes, as opposed to sharing representations among multiple envelopes (in particular, assign_rep is passed a letter object and operator= is passed an envelope object). Letter assignment supports two models as governed by ref_count_incr:

- ref_count_incr = true (default): the incoming letter belongs to another envelope. In this case, increment the reference count in the normal manner so that deallocation of the letter is handled properly.
- ref_count_incr = false: the incoming letter is instantiated on the fly and has no envelope. This case is modeled after get_interface(): a letter is dynamically allocated using new and passed into assign_rep, the letter's reference count is not incremented, and the letter is not remotely deleted (its memory management is passed over to the envelope).

References Dakota::abort_handler(), Interface::interfaceRep, and Interface::referenceCount.

Referenced by DataFitSurrModel::DataFitSurrModel(), parallel_interface_plugin(), LibraryEnvironment::plugin_interface(), and run_dakota().

void eval_tag_prefix (const String & *eval_id_str,* bool *append_iface_id* = `true`)

set the evaluation tag prefix (does not recurse)

default implementation just sets the list of eval ID tags; derived classes containing additional models or interfaces should override (currently no use cases)

References Interface::appendIfaceId, Interface::eval_tag_prefix(), Interface::evalTagPrefix, and Interface::interfaceRep.

Referenced by NestedModel::derived_compute_response(), SingleModel::eval_tag_prefix(), and Interface::eval_tag_prefix().

void response_mapping (const Response & *algebraic_response,* const Response & *core_response,* Response & *total_response*) `[protected]`

combine the response from algebraic_mappings() with the response from derived_map() to create the total response

This function will get invoked even when only algebraic mappings are active (no core mappings from derived_map), since the AMPL algebraic_response may be ordered differently from the total_response. In this case, the core_response object is unused.

References Dakota::_NPOS, Dakota::abort_handler(), Response::active_set_derivative_vector(), Response::active_set_request_vector(), Interface::algebraicACVIds, Interface::algebraicFnIndices, Interface::coreMappings, Dakota::find_index(), Response::function_gradient(), Response::function_gradient_view(), Response::function_gradients(),

Response::function_hessian(), Response::function_hessian_view(), Response::function_hessians(), Response::function_value(), Response::function_values(), Response::function_values_view(), Interface::outputLevel, Response::reset(), and Response::reset_inactive().

Referenced by ApproximationInterface::map(), ApplicationInterface::map(), ApplicationInterface::synch(), and ApplicationInterface::synch_nowait().

Interface * get_interface (ProblemDescDB & *problem_db*) [private]

Used by the envelope to instantiate the correct letter class.

used only by the envelope constructor to initialize interfaceRep to the appropriate derived type.

References ProblemDescDB::get_string(), ProblemDescDB::get_ushort(), and Interface::interface_type().

Referenced by Interface::Interface().

13.52.4 Member Data Documentation

IntResponseMap rawResponseMap [protected]

Set of responses returned after either a blocking or nonblocking schedule of asynchronous evaluations.

The map is a full/partial set of completions which are identified through their evalIdCntr key. The raw set is postprocessed (i.e., finite diff grads merged) in Model::synchronize() where it becomes responseMap.

Referenced by ApplicationInterface::asynchronous_local_evaluations(), ApplicationInterface::process_asynch_local(), ApplicationInterface::process_synch_local(), ApplicationInterface::receive_evaluation(), ApplicationInterface::synch(), ApproximationInterface::synch(), ApplicationInterface::synch_nowait(), ApproximationInterface::synch_nowait(), ApplicationInterface::test_local_backfill(), and ApplicationInterface::test_receives_backfill().

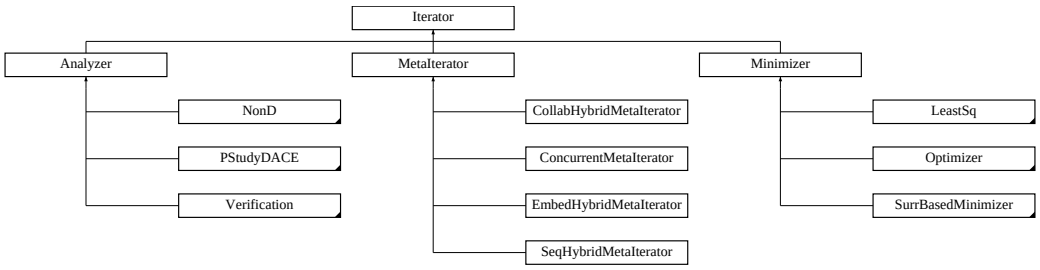
The documentation for this class was generated from the following files:

- DakotaInterface.hpp
- DakotaInterface.cpp

13.53 Iterator Class Reference

Base class for the iterator class hierarchy.

Inheritance diagram for Iterator:



Public Member Functions

- Iterator ()

  default constructor
- Iterator (ProblemDescDB &problem_db)

standard envelope constructor, which constructs its own model(s)

- Iterator (ProblemDescDB &problem_db, Model &model)

  alternate envelope constructor which uses the ProblemDescDB but accepts a model from a higher level (meta-iterator) context, instead of constructing its own

- Iterator (const String &method_string, Model &model)

  alternate envelope constructor for instantiations by name without the ProblemDescDB

- Iterator (const Iterator &iterator)

  copy constructor

- virtual ~Iterator ()

  destructor

- Iterator operator= (const Iterator &iterator)

  assignment operator

- virtual void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance

- virtual void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance

- virtual void initialize_run ()

  utility function to perform common operations prior to pre_run(); typically memory initialization; setting of instance pointers

- virtual void pre_run ()

  pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

- virtual void core_run ()

  core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post

- virtual void post_run (std::ostream &s)

  post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way

- virtual void finalize_run ()

  utility function to perform common operations following post_run(); deallocation and resetting of instance pointers

- virtual void pre_output ()

  write variables to file, following pre-run

- virtual void post_input ()

  read tabular data for post-run mode

- virtual void reset ()

  restore initial state for repeated sub-iterator executions

- virtual void initialize_iterator (int job_index)

  used by IteratorScheduler to set the starting data for a run

- virtual void pack_parameters_buffer (MPIPackBuffer &send_buffer, int job_index)

  used by IteratorScheduler to pack starting data for an iterator run

- virtual void unpack_parameters_buffer (MPIUnpackBuffer &recv_buffer)

  used by IteratorScheduler to unpack starting data for an iterator run

- virtual void unpack_parameters_initialize (MPIUnpackBuffer &recv_buffer)

  used by IteratorScheduler to unpack starting data and initialize an iterator run

- virtual void pack_results_buffer (MPIPackBuffer &send_buffer, int job_index)

used by IteratorScheduler to pack results data from an iterator run

- virtual void unpack_results_buffer (MPIUnpackBuffer &recv_buffer, int job_index)

  used by IteratorScheduler to unpack results data from an iterator run
- virtual void update_local_results (int job_index)

  used by IteratorScheduler to update local results arrays
- virtual const Variables & variables_results () const

  return a single final iterator solution (variables)
- virtual const Response & response_results () const

  return a single final iterator solution (response)
- virtual const VariablesArray & variables_array_results ()

  return multiple final iterator solutions (variables). This should only be used if returns_multiple_points() returns true.
- virtual const ResponseArray & response_array_results ()

  return multiple final iterator solutions (response). This should only be used if returns_multiple_points() returns true.
- virtual bool accepts_multiple_points () const

  indicates if this iterator accepts multiple initial points. Default return is false. Override to return true if appropriate.
- virtual bool returns_multiple_points () const

  indicates if this iterator returns multiple final points. Default return is false. Override to return true if appropriate.
- virtual void initial_points (const VariablesArray &pts)

  sets the multiple initial points for this iterator. This should only be used if accepts_multiple_points() returns true.
- virtual void response_results_active_set (const ActiveSet &set)

  set the requested data for the final iterator response results
- virtual void initialize_graphics (int iterator_server_id=1)

  initialize the 2D graphics window and the tabular graphics data
- virtual void print_results (std::ostream &s)

  print the final iterator results
- virtual const Model & algorithm_space_model () const

  return the result of any recasting or surrogate model recursion layered on top of iteratedModel by the derived Iterator ctor chain
- virtual unsigned short uses_method () const

  return name of any enabling iterator used by this iterator
- virtual void method_recourse ()

  perform a method switch, if possible, due to a detected conflict
- virtual const VariablesArray & all_variables ()

  return the complete set of evaluated variables
- virtual const RealMatrix & all_samples ()

  return the complete set of evaluated samples
- virtual const IntResponseMap & all_responses () const

  return the complete set of computed responses
- virtual int num_samples () const

  get the current number of samples
- virtual void sampling_reset (int min_samples, bool all_data_flag, bool stats_flag)

  reset sampling iterator to use at least min_samples

- virtual void sampling_reference (int samples_ref)

  set reference number of samples, which is a lower bound during reset
- virtual unsigned short sampling_scheme () const

  return sampling name
- virtual bool compact_mode () const

  returns Analyzer::compactMode
- void init_communicators (ParLevLIter pl_iter)

  initialize the communicators associated with this Iterator instance
- void set_communicators (ParLevLIter pl_iter)

  set the communicators associated with this Iterator instance
- void free_communicators (ParLevLIter pl_iter)

  free the communicators associated with this Iterator instance
- void parallel_configuration_iterator (ParConfigLIter pc_iter)

  set methodPCIter
- ParConfigLIter parallel_configuration_iterator () const

  return methodPCIter
- void run (ParLevLIter pl_iter)

  invoke set_communicators(pl_iter) prior to run()
- void run ()

  orchestrate initialize/pre/core/post/finalize phases
- void assign_rep (Iterator *iterator_rep, bool ref_count_incr=true)

  replaces existing letter with a new one
- void iterated_model (const Model &model)

  set the iteratedModel (iterators and meta-iterators using a single model instance)
- Model & iterated_model ()

  return the iteratedModel (iterators & meta-iterators using a single model instance)
- ProblemDescDB & problem_description_db () const

  return the problem description database (probDescDB)
- ParallelLibrary & parallel_library () const

  return the parallel library (parallelLib)
- void method_name (unsigned short m_name)

  set the method name to an enumeration value
- unsigned short method_name () const

  return the method name via its native enumeration value
- void method_string (const String &m_str)

  set the method name by string
- String method_string () const

  return the method name by string
- String method_enum_to_string (unsigned short method_name) const

  convert a method name enumeration value to a string
- unsigned short method_string_to_enum (const String &method_name) const

  convert a method name string to an enumeration value
- String submethod_enum_to_string (unsigned short submethod_name) const

convert a method name enumeration value to a string

- const String & method_id () const

  return the method identifier (methodId)

- int maximum_evaluation_concurrency () const

  return the maximum evaluation concurrency supported by the iterator

- void maximum_evaluation_concurrency (int max_conc)

  set the maximum evaluation concurrency supported by the iterator

- void convergence_tolerance (Real conv_tol)

  set the method convergence tolerance (convergenceTol)

- Real convergence_tolerance () const

  return the method convergence tolerance (convergenceTol)

- void output_level (short out_lev)

  set the method output level (outputLevel)

- short output_level () const

  return the method output level (outputLevel)

- void summary_output (bool summary_output_flag)

  Set summary output control; true enables evaluation/results summary.

- size_t num_final_solutions () const

  return the number of solutions to retain in best variables/response arrays

- void num_final_solutions (size_t num_final)

  set the number of solutions to retain in best variables/response arrays

- void active_set (const ActiveSet &set)

  set the default active set vector (for use with iterators that employ evaluate_parameter_sets())

- const ActiveSet & active_set () const

  return the default active set vector (used by iterators that employ evaluate_parameter_sets())

- void sub_iterator_flag (bool si_flag)

  set subIteratorFlag (and update summaryOutputFlag if needed)

- void active_variable_mappings (const SizetArray &c_index1, const SizetArray &di_index1, const SizetArray &ds_index1, const SizetArray &dr_index1, const ShortArray &c_target2, const ShortArray &di_target2, const ShortArray &ds_target2, const ShortArray &dr_target2)

  set primaryA{CV,DIV,DRV}MapIndices, secondaryA{CV,DIV,DRV}MapTargets

- bool is_null () const

  function to check iteratorRep (does this envelope contain a letter?)

- Iterator * iterator_rep () const

  returns iteratorRep for access to derived class member functions that are not mapped to the top Iterator level

- virtual void eval_tag_prefix (const String &eval_id_str)

  set the hierarchical eval ID tag prefix

Protected Member Functions

- Iterator (BaseConstructor, ProblemDescDB &problem_db)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
- Iterator (NoDBBaseConstructor, unsigned short method_name, Model &model)

  alternate constructor for base iterator classes constructed on the fly
- Iterator (NoDBBaseConstructor, unsigned short method_name)

  alternate constructor for base iterator classes constructed on the fly
- virtual void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- virtual void update_from_model (const Model &model)

  set inherited data attributes based on extractions from incoming model
- virtual const VariablesArray & initial_points () const

  gets the multiple initial points for this iterator. This will only be meaningful after a call to initial_points mutator.
- StrStrSizet run_identifier () const

  get the unique run identifier based on method name, id, and number of executions

Protected Attributes

- ProblemDescDB & probDescDB

  class member reference to the problem description database
- ParallelLibrary & parallelLib

  class member reference to the parallel library
- ParConfigLIter methodPCIter

  the active ParallelConfiguration used by this Iterator instance
- Model iteratedModel

  the model to be iterated (for iterators and meta-iterators employing a single model instance)
- unsigned short methodName

  name of the iterator (the user's method spec)
- Real convergenceTol

  iteration convergence tolerance
- int maxIterations

  maximum number of iterations for the iterator
- int maxFunctionEvals

  maximum number of fn evaluations for the iterator
- int maxEvalConcurrency

  maximum number of concurrent model evaluations
- ActiveSet activeSet

  the response data requirements on each function evaluation
- size_t numFinalSolutions

  number of solutions to retain in best variables/response arrays
- VariablesArray bestVariablesArray

  collection of N best solution variables found during the study; always in context of Model originally passed to the Iterator (any in-flight Recasts must be undone)

- ResponseArray bestResponseArray

  collection of N best solution responses found during the study; always in context of Model originally passed to the Iterator (any in-flight Recasts must be undone)

- bool subIteratorFlag

  flag indicating if this Iterator is a sub-iterator (NestedModel::subIterator or DataFitSurrModel::daceIterator)

- SizetArray primaryACVarMapIndices

  "primary" all continuous variable mapping indices flowed down from higher level iteration

- SizetArray primaryADIVarMapIndices

  "primary" all discrete int variable mapping indices flowed down from higher level iteration

- SizetArray primaryADSVarMapIndices

  "primary" all discrete string variable mapping indices flowed down from higher level iteration

- SizetArray primaryADRVarMapIndices

  "primary" all discrete real variable mapping indices flowed down from higher level iteration

- ShortArray secondaryACVarMapTargets

  "secondary" all continuous variable mapping targets flowed down from higher level iteration

- ShortArray secondaryADIVarMapTargets

  "secondary" all discrete int variable mapping targets flowed down from higher level iteration

- ShortArray secondaryADSVarMapTargets

  "secondary" all discrete string variable mapping targets flowed down from higher level iteration

- ShortArray secondaryADRVarMapTargets

  "secondary" all discrete real variable mapping targets flowed down from higher level iteration

- short outputLevel

  output verbosity level: {SILENT,QUIET,NORMAL,VERBOSE,DEBUG}_OUTPUT

- bool summaryOutputFlag

  flag for summary output (evaluation stats, final results); default true, but false for on-the-fly (helper) iterators and sub-iterator use cases

- ResultsManager & resultsDB

  reference to the global iterator results database

- ResultsNames resultsNames

  valid names for iterator results

Private Member Functions

- Iterator * get_iterator (ProblemDescDB &problem_db)

  Used by the envelope to instantiate the correct letter class.

- Iterator * get_iterator (ProblemDescDB &problem_db, Model &model)

  Used by the envelope to instantiate the correct letter class.

- Iterator * get_iterator (const String &method_string, Model &model)

  Used by the envelope to instantiate the correct letter class.

Private Attributes

- String methodId

  method identifier string from the input file
- size_t execNum

  an execution number for this instance of the class, unique across all instances of same methodName/methodId
- std::map< size_t, ParConfigLIter > methodPCIterMap

  track the available configurations that have been created (init_communicators) and are available for activation at run time (set_communicators)
- Iterator * iteratorRep

  pointer to the letter (initialized only for the envelope)
- int referenceCount

  number of objects sharing iteratorRep

13.53.1 Detailed Description

Base class for the iterator class hierarchy.

The Iterator class is the base class for one of the primary class hierarchies in DAKOTA. The iterator hierarchy contains all of the iterative algorithms which use repeated execution of simulations as function evaluations. For memory efficiency and enhanced polymorphism, the iterator hierarchy employs the "letter/envelope idiom" (see Coplien "Advanced C++", p. 133), for which the base class (Iterator) serves as the envelope and one of the derived classes (selected in Iterator::get_iterator()) serves as the letter.

13.53.2 Constructor & Destructor Documentation

Iterator ( )

default constructor

The default constructor is used in Vector<Iterator> instantiations and for initialization of Iterator objects contained in meta-Iterators and Model recursions. iteratorRep is NULL in this case, making it necessary to check for NULL pointers in the copy constructor, assignment operator, and destructor.

Referenced by SurrBasedGlobalMinimizer::SurrBasedGlobalMinimizer(), and SurrBasedLocalMinimizer::-SurrBasedLocalMinimizer().

Iterator (ProblemDescDB & *problem_db*)

standard envelope constructor, which constructs its own model(s)

Envelope constructor only needs to extract enough data to properly execute get_iterator(), since letter holds the actual base class data. This version is used for top-level ProblemDescDB-driven construction of all Iterators and MetaIterators, which construct their own Model instances.

References Dakota::abort_handler(), Iterator::get_iterator(), and Iterator::iteratorRep.

Iterator (ProblemDescDB & *problem_db*, Model & *model*)

alternate envelope constructor which uses the ProblemDescDB but accepts a model from a higher level (meta-iterator) context, instead of constructing its own

Envelope constructor only needs to extract enough data to properly execute get_iterator(), since letter holds the actual base class data. This version is used for ProblemDescDB-driven construction of Iterators that are passed a Model from a higher-level context (e.g., a MetaIterator instantiates its sub-iterator(s) by name instead of pointer and passes in its iteratedModel, since these sub-iterators lack their own model pointers).

References Dakota::abort_handler(), Iterator::get_iterator(), and Iterator::iteratorRep.

Iterator (const String & *method_string*, Model & *model*)

alternate envelope constructor for instantiations by name without the ProblemDescDB

Used in sub-iterator instantiations within iterator constructors. Envelope constructor only needs to extract enough data to properly execute get_iterator(), since letter holds the actual base class data. This version is used for lightweight constructions without the ProblemDescDB.

References Dakota::abort_handler(), Iterator::get_iterator(), and Iterator::iteratorRep.

Iterator (const Iterator & *iterator*)

copy constructor

Copy constructor manages sharing of iteratorRep and incrementing of referenceCount.

References Iterator::iteratorRep, and Iterator::referenceCount.

~Iterator ( ) [virtual]

destructor

Destructor decrements referenceCount and only deletes iteratorRep when referenceCount reaches zero.

References Iterator::iteratorRep, and Iterator::referenceCount.

Iterator (BaseConstructor, ProblemDescDB & *problem_db*) [protected]

constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor builds the base class data for all inherited iterators, including meta-iterators. get_iterator() instantiates a derived class and the derived class selects this base class constructor in its initialization list (to avoid the recursion of the base class constructor calling get_iterator() again). Since the letter IS the representation, its representation pointer is set to NULL (an uninitialized pointer causes problems in ~Iterator).

References Iterator::method_enum_to_string(), Iterator::methodName, and Iterator::outputLevel.

Iterator (NoDBBaseConstructor, unsigned short *method_name*, Model & *model*) [protected]

alternate constructor for base iterator classes constructed on the fly

This alternate constructor builds base class data for inherited iterators. It is used for on-the-fly instantiations for which DB queries cannot be used, and is not used for construction of meta-iterators. It has no incoming model, so only sets up a minimal set of defaults. However, its use is preferable to the default constructor, which should remain as minimal as possible.

Iterator (NoDBBaseConstructor, unsigned short *method_name*) [protected]

alternate constructor for base iterator classes constructed on the fly

This alternate constructor builds base class data for inherited iterators. It is used for on-the-fly instantiations for which DB queries cannot be used, and is not used for construction of meta-iterators. It has no incoming model, so only sets up a minimal set of defaults. However, its use is preferable to the default constructor, which should remain as minimal as possible.

13.53.3 Member Function Documentation

Iterator operator= (const Iterator & *iterator*)

assignment operator

Assignment operator decrements referenceCount for old iteratorRep, assigns new iteratorRep, and increments referenceCount for new iteratorRep.

References Iterator::iteratorRep, and Iterator::referenceCount.

void initialize_run ( ) [virtual]

utility function to perform common operations prior to pre_run(); typically memory initialization; setting of instance pointers

Perform initialization phases of run sequence, like allocating memory and setting instance pointers. Commonly used in sub-iterator executions. This is a virtual function; when re-implementing, a derived class must call its nearest parent's initialize_run(), typically *before* performing its own implementation steps.

Reimplemented in SNLLOptimizer, NonD, NLPQLPOptimizer, Analyzer, SNLLLeastSq, Minimizer, CONMINOptimizer, DOTOptimizer, Optimizer, and LeastSq.

References Iterator::initialize_run(), and Iterator::iteratorRep.

Referenced by Iterator::initialize_run(), Iterator::run(), and SeqHybridMetaIterator::run_sequential_adaptive().

void pre_run ( ) [virtual]

pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

pre-run phase, which a derived iterator may optionally reimplement; when not present, pre-run is likely integrated into the derived run function. This is a virtual function; when re-implementing, a derived class must call its nearest parent's pre_run(), if implemented, typically *before* performing its own implementation steps.

Reimplemented in NonDLHSSampling, ConcurrentMetaIterator, DDACEDesignCompExp, ParamStudy, FSUDesignCompExp, and PSUADEDesignCompExp.

References Iterator::iteratorRep, and Iterator::pre_run().

Referenced by ParamStudy::pre_run(), Iterator::pre_run(), and Iterator::run().

void core_run ( ) [virtual]

core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post

Virtual run function for the iterator class hierarchy. All derived classes need to redefine it.

Reimplemented in NonD, SeqHybridMetaIterator, ConcurrentMetaIterator, Optimizer, EmbedHybridMetaIterator, CollabHybridMetaIterator, LeastSq, PStudyDACE, SurrBasedMinimizer, and Verification.

References Dakota::abort_handler(), Iterator::core_run(), and Iterator::iteratorRep.

Referenced by Iterator::core_run(), and Iterator::run().

void post_run (std::ostream & *s*) [virtual]

post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/Responses and perform final analysis phase in a standalone way

Post-run phase, which a derived iterator may optionally reimplement; when not present, post-run is likely integrated into run. This is a virtual function; when re-implementing, a derived class must call its nearest parent's post_run(), typically *after* performing its own implementation steps.

Reimplemented in SNLLOptimizer, COLINOptimizer, SNLLLeastSq, Analyzer, Minimizer, NonDLHSSampling, Optimizer, DDACEDesignCompExp, ParamStudy, FSUDesignCompExp, MetaIterator, LeastSq, and PSUADE-DesignCompExp.

References Iterator::iteratorRep, and Iterator::post_run().

Referenced by Iterator::post_run(), and Iterator::run().

void finalize_run ( ) [virtual]

utility function to perform common operations following post_run(); deallocation and resetting of instance pointers

Optional: perform finalization phases of run sequence, like deallocating memory and resetting instance pointers. Commonly used in sub-iterator executions. This is a virtual function; when re-implementing, a derived class must call its nearest parent's finalize_run(), typically *after* performing its own implementation steps.

Reimplemented in SNLLOptimizer, NonD, SNLLLeastSq, Minimizer, Optimizer, and LeastSq.

References Iterator::finalize_run(), and Iterator::iteratorRep.

Referenced by Minimizer::finalize_run(), NonD::finalize_run(), Iterator::finalize_run(), Iterator::run(), and SeqHybridMetaIterator::run_sequential_adaptive().

void initialize_graphics (int *iterator_server_id* = *1*) [virtual]

initialize the 2D graphics window and the tabular graphics data

This is a convenience function for encapsulating graphics initialization operations. It is overridden by derived classes that specialize the graphics display.

Reimplemented in SurrBasedMinimizer, and NonDReliability.

References Model::auto_graphics(), Graphics::create_plots_2d(), OutputManager::create_tabular_datastream(), Model::current_response(), Model::current_variables(), OutputManager::graph2DFlag, OutputManager::graphics(), Iterator::initialize_graphics(), Iterator::iteratedModel, Iterator::iteratorRep, ParallelLibrary::output_manager(), Iterator::parallelLib, and OutputManager::tabularDataFlag.

Referenced by CollabHybridMetaIterator::core_run(), EmbedHybridMetaIterator::core_run(), ConcurrentMetaIterator::core_run(), Environment::execute(), Iterator::initialize_graphics(), SeqHybridMetaIterator::run_sequential(), and SeqHybridMetaIterator::run_sequential_adaptive().

void print_results (std::ostream & *s*) [virtual]

print the final iterator results

This virtual function provides additional iterator-specific final results outputs beyond the function evaluation summary printed in finalize_run().

Reimplemented in Analyzer, NonDLHSSampling, NonDLocalReliability, NonDPOFDarts, SeqHybridMetaIterator, NonDAdaptiveSampling, ConcurrentMetaIterator, NonDGPImpSampling, Optimizer, NonDAdaptImpSampling, NonDGlobalReliability, NonDIncremLHSSampling, NonDInterval, NonDExpansion, LeastSq, PStudyDACE, SurrBasedMinimizer, RichExtrapVerification, and Verification.

References Iterator::iteratorRep, and Iterator::print_results().

Referenced by MetaIterator::post_run(), Minimizer::post_run(), Iterator::print_results(), and EfficientSubspaceMethod::reduced_space_uq().

unsigned short uses_method ( ) const [virtual]

return name of any enabling iterator used by this iterator

This is used to avoid clashes in state between non-object-oriented (i.e., F77, C) iterator executions, when such iterators could potentially be executing simulataneously (e.g., nested execution). It is not an issue (and a used method is not reported) in cases where a helper execution is completed before a lower level one could be initiated;

an example of this is DIRECT for maximization of expected improvement: the EIF maximization is completed before a new point evaluation (which could include nested iteration) is performed.

Reimplemented in NonDLocalReliability, and NonDLocalInterval.

References Iterator::iteratorRep, and Iterator::uses_method().

Referenced by DOTOptimizer::initialize(), CONMINOptimizer::initialize(), NCSUOptimizer::initialize(), NLPQLPOptimizer::initialize(), NonDLocalInterval::NonDLocalInterval(), NonDLocalReliability::NonDLocalReliability(), SOLBase::SOLBase(), and Iterator::uses_method().

void run ( )

orchestrate initialize/pre/core/post/finalize phases

Iterator supports a construct/initialize-run/pre-run/core-run/post-run/ finalize-run/destruct progression. This member (non-virtual) function sequences these run phases.

References ParallelLibrary::command_line_post_run(), ParallelLibrary::command_line_pre_run(), ParallelLibrary::command_line_run(), Iterator::core_run(), Iterator::execNum, Iterator::finalize_run(), ResultsID::increment_id(), Iterator::initialize_run(), ResultsID::instance(), Iterator::iteratorRep, Iterator::method_enum_to_string(), Iterator::method_id(), Iterator::method_string(), Iterator::methodName, Iterator::outputLevel, Iterator::parallelLib, Iterator::post_input(), Iterator::post_run(), Iterator::pre_output(), Iterator::pre_run(), Iterator::run(), and Iterator::summaryOutputFlag.

Referenced by Iterator::run().

void assign_rep (Iterator * *iterator_rep*, bool *ref_count_incr* = `true`)

replaces existing letter with a new one

Similar to the assignment operator, the assign_rep() function decrements referenceCount for the old iteratorRep and assigns the new iteratorRep. It is different in that it is used for publishing derived class letters to existing envelopes, as opposed to sharing representations among multiple envelopes (in particular, assign_rep is passed a letter object and operator= is passed an envelope object). Letter assignment supports two models as governed by ref_count_incr:

- ref_count_incr = true (default): the incoming letter belongs to another envelope. In this case, increment the reference count in the normal manner so that deallocation of the letter is handled properly.
- ref_count_incr = false: the incoming letter is instantiated on the fly and has no envelope. This case is modeled after get_iterator(): a letter is dynamically allocated using new and passed into assign_rep, the letter's reference count is not incremented, and the letter is not remotely deleted (its memory management is passed over to the envelope).

References Dakota::abort_handler(), Iterator::iterator_rep(), Iterator::iteratorRep, and Iterator::referenceCount.

Referenced by NonDExpansion::construct_cubature(), NonDExpansion::construct_expansion_sampler(), NonDAdaptiveSampling::construct_fsu_sampler(), NonD::construct_lhs(), NonDExpansion::construct_quadrature(), NonDExpansion::construct_sparse_grid(), EffGlobalMinimizer::EffGlobalMinimizer(), EfficientSubspaceMethod::init_fullspace_sampler(), NonDLocalInterval::method_recourse(), NonDLocalReliability::method_recourse(), NonDAdaptiveSampling::NonDAdaptiveSampling(), NonDBayesCalibration::NonDBayesCalibration(), NonDGlobalInterval::NonDGlobalInterval(), NonDGlobalReliability::NonDGlobalReliability(), NonDGPImpSampling::NonDGPImpSampling(), NonDGPMSABayesCalibration::NonDGPMSABayesCalibration(), NonDLHSInterval::NonDLHSInterval(), NonDLocalInterval::NonDLocalInterval(), NonDLocalReliability::NonDLocalReliability(), GaussProcApproximation::optimize_theta_global(), GaussProcApproximation::optimize_theta_multipoint(), and SurrBasedLocalMinimizer::relax_constraints().

void eval_tag_prefix (const String & *eval_id_str*) [virtual]

set the hierarchical eval ID tag prefix

This prepend may need to become a virtual function if the tagging should propagate to other subModels or helper Iterators an Iterator may contain.

References Model::eval_tag_prefix(), Iterator::eval_tag_prefix(), Iterator::iteratedModel, and Iterator::iteratorRep.

Referenced by DataFitSurrModel::build_global(), NestedModel::derived_compute_response(), Iterator::eval_tag_prefix(), and NestedModel::initialize_iterator().

Iterator * get_iterator (ProblemDescDB & *problem_db*) [private]

Used by the envelope to instantiate the correct letter class.

Used only by the envelope constructor to initialize iteratorRep to the appropriate derived type, as given by the DB's method_name. Supports all iterators and meta-iterators. These instantiations will NOT recurse on the Iterator(problem_db) constructor due to the use of BaseConstructor.

References ProblemDescDB::get_model(), ProblemDescDB::get_ushort(), Iterator::method_enum_to_string(), Iterator::method_name(), and Dakota::SUBMETHOD_COLLABORATIVE.

Referenced by Iterator::Iterator().

Iterator * get_iterator (ProblemDescDB & *problem_db,* Model & *model*) [private]

Used by the envelope to instantiate the correct letter class.

Used only by the envelope constructor to initialize iteratorRep to the appropriate derived type. Alternate construction of meta-iterators is supported to enable use of meta-iterators as components. These instantiations will NOT recurse on the Iterator(problem_db, model) constructor due to the use of BaseConstructor.

References ProblemDescDB::get_ushort(), Iterator::method_enum_to_string(), Iterator::method_name(), Iterator::probDescDB, and Dakota::SUBMETHOD_COLLABORATIVE.

Iterator * get_iterator (const String & *method_string,* Model & *model*) [private]

Used by the envelope to instantiate the correct letter class.

Used only by the envelope constructor to initialize iteratorRep to the appropriate derived type, as given by the passed method_string. Lightweight instantiations by name are supported by a subset of Iterators (primarily Minimizers).

References Dakota::strbegins(), and Dakota::strends().

13.53.4 Member Data Documentation

ProblemDescDB& probDescDB [protected]

class member reference to the problem description database

Iterator and Model cannot use a shallow copy of ProblemDescDB due to circular destruction dependency (reference counts can't get to 0), since ProblemDescDB contains {iterator,model}List.

Referenced by MetaIterator::allocate_by_name(), MetaIterator::allocate_by_pointer(), Analyzer::Analyzer(), MetaIterator::check_model(), COLINOptimizer::COLINOptimizer(), CollabHybridMetaIterator::CollabHybridMetaIterator(), NonDExpansion::construct_expansion_sampler(), Minimizer::data_transform_model(), SurrBasedMinimizer::derived_init_communicators(), CollabHybridMetaIterator::derived_init_communicators(), EmbedHybridMetaIterator::derived_init_communicators(), ConcurrentMetaIterator::derived_init_communicators(), SeqHybridMetaIterator::derived_init_communicators(), EffGlobalMinimizer::EffGlobalMinimizer(), MetaIterator::estimate_by_name(), MetaIterator::estimate_by_pointer(), FSUDesignCompExp::FSUDesignCompExp(), Iterator::get_iterator(),

ConcurrentMetaIterator::initialize_model(), Minimizer::initialize_scaling(), JEGAOptimizer::JEGAOptimizer(), MetaIterator::new_model(), NLSSOLLeastSq::NLSSOLLeastSq(), NomadOptimizer::NomadOptimizer(), NonD::NonD(), NonDAdaptiveSampling::NonDAdaptiveSampling(), NonDBayesCalibration::NonDBayesCalibration(), NonDCalibration::NonDCalibration(), NonDGlobalInterval::NonDGlobalInterval(), NonDGlobalReliability::NonDGlobalReliability(), NonDGPImpSampling::NonDGPImpSampling(), NonDGPMSABayesCalibration::NonDGPMSABayesCalibration(), NonDInterval::NonDInterval(), NonDLocalInterval::NonDLocalInterval(), NonDLocalReliability::NonDLocalReliability(), NonDPolynomialChaos::NonDPolynomialChaos(), NonDQuadrature::NonDQuadrature(), NonDSparseGrid::NonDSparseGrid(), NonDStochCollocation::NonDStochCollocation(), NPSOLOptimizer::NPSOLOptimizer(), OptDartsOptimizer::OptDartsOptimizer(), Optimizer::Optimizer(), ParamStudy::ParamStudy(), NonlinearCGOptimizer::parse_options(), NonDAdaptiveSampling::parse_options(), Iterator::problem_description_db(), SeqHybridMetaIterator::SeqHybridMetaIterator(), APPSOptimizer::set_apps_parameters(), COLINOptimizer::set_solver_parameters(), SNLLLeastSq::SNLLLeastSq(), SNLLOptimizer::SNLLOptimizer(), COLINOptimizer::solver_setup(), SurrBasedGlobalMinimizer::SurrBasedGlobalMinimizer(), SurrBasedLocalMinimizer::SurrBasedLocalMinimizer(), and SurrBasedMinimizer::SurrBasedMinimizer().

int maxEvalConcurrency `[protected]`

maximum number of concurrent model evaluations

This is important for parallel configuration init/set/free and may be set within empty envelope instances. Therefore, it cannot be pushed down into Analyzer/Minimizer derived classes.

Referenced by DDACEDesignCompExp::DDACEDesignCompExp(), EfficientSubspaceMethod::derived_free_communicators(), SurrBasedMinimizer::derived_free_communicators(), NonDGlobalReliability::derived_free_communicators(), NonDLocalInterval::derived_free_communicators(), NonDBayesCalibration::derived_free_communicators(), NonDExpansion::derived_free_communicators(), NonDGlobalInterval::derived_free_communicators(), NonDPolynomialChaos::derived_free_communicators(), NonDAdaptiveSampling::derived_free_communicators(), NonDGPImpSampling::derived_free_communicators(), NonDLocalReliability::derived_free_communicators(), Iterator::derived_free_communicators(), EfficientSubspaceMethod::derived_init_communicators(), SurrBasedMinimizer::derived_init_communicators(), NonDGlobalReliability::derived_init_communicators(), NonDLocalInterval::derived_init_communicators(), NonDBayesCalibration::derived_init_communicators(), NonDExpansion::derived_init_communicators(), NonDGlobalInterval::derived_init_communicators(), NonDPolynomialChaos::derived_init_communicators(), NonDAdaptiveSampling::derived_init_communicators(), NonDGPImpSampling::derived_init_communicators(), NonDLocalReliability::derived_init_communicators(), Iterator::derived_init_communicators(), EfficientSubspaceMethod::derived_set_communicators(), SurrBasedMinimizer::derived_set_communicators(), NonDExpansion::derived_set_communicators(), NonDBayesCalibration::derived_set_communicators(), NonDPolynomialChaos::derived_set_communicators(), NonDLocalReliability::derived_set_communicators(), Iterator::derived_set_communicators(), NonD::derived_set_communicators(), EffGlobalMinimizer::EffGlobalMinimizer(), EfficientSubspaceMethod::EfficientSubspaceMethod(), FSUDesignCompExp::FSUDesignCompExp(), NonDCubature::initialize_grid(), NonDQuadrature::initialize_grid(), NonDSparseGrid::initialize_grid(), NonDExpansion::initialize_u_space_model(), JEGAOptimizer::JEGAOptimizer(), Iterator::maximum_evaluation_concurrency(), NonDAdaptImpSampling::NonDAdaptImpSampling(), NonDBayesCalibration::NonDBayesCalibration(), NonDCubature::NonDCubature(), NonDGlobalInterval::NonDGlobalInterval(), NonDGlobalReliability::NonDGlobalReliability(), NonDLHSInterval::NonDLHSInterval(), NonDPolynomialChaos::NonDPolynomialChaos(), NonDQuadrature::NonDQuadrature(), NonDSampling::NonDSampling(), NonDSparseGrid::NonDSparseGrid(), Analyzer::num_samples(), ParamStudy::ParamStudy(), PSUADEDesignCompExp::PSUADEDesignCompExp(), RichExtrapVerification::RichExtrapVerification(), APPSOptimizer::set_apps_parameters(), COLINOptimizer::set_solver_parameters(), SNLLOptimizer::SNLLOptimizer(), and Iterator::update_from_model().

The documentation for this class was generated from the following files:

- DakotaIterator.hpp
- DakotaIterator.cpp

13.54 IteratorScheduler Class Reference

Environment corresponding to execution as a stand-alone application.

Public Member Functions

- IteratorScheduler (ParallelLibrary &parallel_lib, bool peer_assign_jobs, int num_servers=0, int procs_per_iterator=0, short scheduling=DEFAULT_SCHEDULING)

  constructor
- ~IteratorScheduler ()

  destructor
- void init_iterator_parallelism (int max_iterator_concurrency, int min_procs_per_iterator=1, int max_procs_per_iterator=0, short default_config=PUSH_DOWN)

  convenience function for initializing iterator communicators, setting parallel configuration attributes, and managing outputs and restart.
- int init_evaluation_concurrency (ProblemDescDB &problem_db, Iterator &the_iterator, Model &the_model)

  convenience function for performing sufficient initialization to define the maximum evaluation concurrency
- int init_evaluation_concurrency (const String &method_string, Iterator &the_iterator, Model &the_model)

  convenience function for performing sufficient initialization to define the maximum evaluation concurrency
- void init_iterator (ProblemDescDB &problem_db, Iterator &the_iterator, Model &the_model)

  invokes static version of this function with appropriate parallelism level
- void init_iterator (const String &method_string, Iterator &the_iterator, Model &the_model)

  invokes static version of this function with appropriate parallelism level
- void set_iterator (Iterator &the_iterator)

  invokes static version of this function with appropriate parallelism level
- void run_iterator (Iterator &the_iterator)

  invokes static version of this function with appropriate parallelism level
- void free_iterator (Iterator &the_iterator)

  invokes static version of this function with appropriate parallelism level
- void free_iterator_parallelism ()

  convenience function for deallocating the concurrent iterator parallelism level
- template<typename MetaType >
  void schedule_iterators (MetaType &meta_object, Iterator &sub_iterator)

  short convenience function for distributing control among master_dynamic_schedule_iterators(), serve_iterators(), and peer_static_schedule_iterators()
- template<typename MetaType >
  void master_dynamic_schedule_iterators (MetaType &meta_object)

  executed by the scheduler master to manage a dynamic schedule of iterator jobs among slave iterator servers
- void stop_iterator_servers ()

  executed by the scheduler master to terminate slave iterator servers
- template<typename MetaType >
  void serve_iterators (MetaType &meta_object, Iterator &sub_iterator)

  executed on the slave iterator servers to perform iterator jobs assigned by the scheduler master
- template<typename MetaType >
  void peer_static_schedule_iterators (MetaType &meta_object, Iterator &sub_iterator)

executed on iterator peers to manage a static schedule of iterator jobs

- void update (ParConfigLIter pc_iter)

  update schedPCIter
- void update (size_t index)

  update miPLIndex as well as associated settings for concurrent iterator scheduling from the corresponding ParallelLevel
- void update (ParConfigLIter pc_iter, size_t index)

  invoke update(ParConfigLIter) and update(size_t) in sequence
- void iterator_message_lengths (int params_msg_len, int results_msg_len)

  update paramsMsgLen and resultsMsgLen
- bool lead_rank () const

  determines if current processor is rank 0 of the parent comm

Static Public Member Functions

- static void init_iterator (ProblemDescDB &problem_db, Iterator &the_iterator, ParLevLIter pl_iter)

  convenience function for allocation of an iterator and (parallel) initialization of its comms
- static void init_iterator (ProblemDescDB &problem_db, Iterator &the_iterator, Model &the_model, ParLevLIter pl_iter)

  convenience function for allocation of an iterator and (parallel) initialization of its comms
- static void init_iterator (const String &method_string, Iterator &the_iterator, Model &the_model, ParLevLIter pl_iter)

  convenience function for lightweight allocation of an iterator and (parallel) initialization of its comms
- static void set_iterator (Iterator &the_iterator, ParLevLIter pl_iter)

  convenience function for setting comms prior to running an iterator
- static void run_iterator (Iterator &the_iterator, ParLevLIter pl_iter)

  Convenience function for invoking an iterator and managing parallelism. This version omits communicator repartitioning. Function must be public due to use by MINLPNode.
- static void free_iterator (Iterator &the_iterator, ParLevLIter pl_iter)

  convenience function for deallocating comms after running an iterator

Public Attributes

- ParallelLibrary & parallelLib

  reference to the ParallelLibrary instance
- int numIteratorJobs

  number of iterator executions to schedule
- int numIteratorServers

  number of concurrent iterator partitions
- int procsPerIterator

  partition size request
- int iteratorCommRank

  processor rank in iteratorComm
- int iteratorCommSize

  number of processors in iteratorComm

- int iteratorServerId

  identifier for an iterator server
- bool messagePass

  flag for message passing among iterator servers
- short iteratorScheduling

  {DEFAULT,MASTER,PEER}_SCHEDULING
- bool peerAssignJobs

  `flag indicating need for peer 1 to assign jobs`
  < to peers 2-n
- ParConfigLIter schedPCIter

  iterator for active parallel configuration
- size_t miPLIndex

  `index of active parallel level (corresponding`
  < to ParallelConfiguration::miPLIters) to use < for parallelLib send/recv

Private Attributes

- int paramsMsgLen

  length of MPI buffer for parameter input instance(s)
- int resultsMsgLen

  length of MPI buffer for results output instance(s)

13.54.1 Detailed Description

Environment corresponding to execution as a stand-alone application.

This environment corresponds to a stand-alone executable program, e.g., main.cpp. It sets up the ParallelLibrary, and ProblemDescDB objects based on access to command line arguments.

13.54.2 Constructor & Destructor Documentation

IteratorScheduler (ParallelLibrary & *parallel_lib*, bool *peer_assign_jobs*, int *num_servers* = *0*, int *procs_per_iterator* = *0*, short *scheduling* = *DEFAULT_SCHEDULING*)

constructor

Current constructor parameters are the input specification components, which are requests subject to override by ParallelLibrary::init_iterator_communicators().

13.54.3 Member Function Documentation

void init_iterator (ProblemDescDB & *problem_db*, Iterator & *the_iterator*, ParLevLIter *pl_iter*) [static]

convenience function for allocation of an iterator and (parallel) initialization of its comms

This is a convenience function for encapsulating the allocation of communicators prior to running an iterator.

References ProblemDescDB::get_iterator(), ProblemDescDB::get_model(), ProblemDescDB::get_ushort(), Model::init_comms_bcast_flag(), Iterator::init_communicators(), Iterator::is_null(), Model::is_null(), Iterator::iterated_model(), Iterator::maximum_evaluation_concurrency(), Iterator::method_name(), Model::serve_init(), and Model::stop_init().

Referenced by MetaIterator::allocate_by_name(), MetaIterator::allocate_by_pointer(), Environment::construct(), ConcurrentMetaIterator::derived_init_communicators(), NestedModel::derived_init_communicators(), and IteratorScheduler::init_iterator().

void init_iterator (ProblemDescDB & *problem_db,* Iterator & *the_iterator,* Model & *the_model,* ParLevLIter *pl_iter*) `[static]`

convenience function for allocation of an iterator and (parallel) initialization of its comms

This is a convenience function for encapsulating the allocation of communicators prior to running an iterator.

References ProblemDescDB::get_iterator(), ProblemDescDB::get_ushort(), Model::init_comms_bcast_flag(), Iterator::init_communicators(), Iterator::is_null(), Iterator::iterated_model(), Iterator::maximum_evaluation_concurrency(), Iterator::method_name(), Model::serve_init(), and Model::stop_init().

void init_iterator (const String & *method_string,* Iterator & *the_iterator,* Model & *the_model,* ParLevLIter *pl_iter*) `[static]`

convenience function for lightweight allocation of an iterator and (parallel) initialization of its comms

This is a convenience function for encapsulating the allocation of communicators prior to running an iterator.

References Model::init_comms_bcast_flag(), Iterator::init_communicators(), Iterator::is_null(), Iterator::iterated_model(), Iterator::maximum_evaluation_concurrency(), Iterator::method_string(), Model::serve_init(), and Model::stop_init().

void set_iterator (Iterator & *the_iterator,* ParLevLIter *pl_iter*) `[static]`

convenience function for setting comms prior to running an iterator

This is a convenience function for encapsulating the deallocation of communicators after running an iterator.

References Iterator::derived_set_communicators(), and Iterator::set_communicators().

Referenced by CollabHybridMetaIterator::derived_set_communicators(), EmbedHybridMetaIterator::derived_set_communicators(), ConcurrentMetaIterator::derived_set_communicators(), SeqHybridMetaIterator::derived_set_communicators(), NestedModel::derived_set_communicators(), and IteratorScheduler::set_iterator().

void run_iterator (Iterator & *the_iterator,* ParLevLIter *pl_iter*) `[static]`

Convenience function for invoking an iterator and managing parallelism. This version omits communicator repartitioning. Function must be public due to use by MINLPNode.

This is a convenience function for encapsulating the parallel features (run/serve) of running an iterator. This function omits allocation/deallocation of communicators to provide greater efficiency in approaches that involve multiple iterator executions but only require communicator allocation/deallocation to be performed once.

References Iterator::iterated_model(), Iterator::maximum_evaluation_concurrency(), Iterator::method_name(), Iterator::run(), Model::serve_run(), and Model::stop_servers().

Referenced by NestedModel::derived_compute_response(), Environment::execute(), IteratorScheduler::peer_static_schedule_iterators(), IteratorScheduler::run_iterator(), and IteratorScheduler::serve_iterators().

void free_iterator (Iterator & *the_iterator,* ParLevLIter *pl_iter*) `[static]`

convenience function for deallocating comms after running an iterator

This is a convenience function for encapsulating the deallocation of communicators after running an iterator.

References Iterator::derived_free_communicators(), Iterator::free_communicators(), and Iterator::method_name().

Referenced by CollabHybridMetaIterator::derived_free_communicators(), EmbedHybridMetaIterator::derived_free_communicators(), ConcurrentMetaIterator::derived_free_communicators(), SeqHybridMetaIterator::derived_free_communicators(), NestedModel::derived_free_communicators(), Environment::destruct(), and IteratorScheduler::free_iterator().

void init_iterator_parallelism (int *max_iterator_concurrency,* int *min_procs_per_iterator = 1,* int *max_procs_per_iterator = 0,* short *default_config = PUSH_DOWN*)

convenience function for initializing iterator communicators, setting parallel configuration attributes, and managing outputs and restart.

Called from derived class constructors once maxIteratorConcurrency is defined but prior to instantiating Iterators and Models.

References ParallelLibrary::init_iterator_communicators(), IteratorScheduler::iteratorScheduling, IteratorScheduler::numIteratorServers, ParallelLibrary::parallel_configuration_iterator(), IteratorScheduler::parallelLib, IteratorScheduler::procsPerIterator, ParallelLibrary::push_output_tag(), IteratorScheduler::update(), and ParallelLibrary::world_size().

Referenced by CollabHybridMetaIterator::derived_init_communicators(), EmbedHybridMetaIterator::derived_init_communicators(), ConcurrentMetaIterator::derived_init_communicators(), SeqHybridMetaIterator::derived_init_communicators(), and NestedModel::derived_init_communicators().

int init_evaluation_concurrency (ProblemDescDB & *problem_db,* Iterator & *the_iterator,* Model & *the_model*)

convenience function for performing sufficient initialization to define the maximum evaluation concurrency

This is a convenience function for computing the maximum evaluation concurrency prior to concurrent iterator partitioning.

References ParallelLibrary::bcast(), ProblemDescDB::get_iterator(), Iterator::is_null(), Iterator::maximum_evaluation_concurrency(), IteratorScheduler::parallelLib, IteratorScheduler::schedPCIter, ParallelLevel::server_communicator_rank(), and ParallelLevel::server_communicator_size().

Referenced by ConcurrentMetaIterator::derived_init_communicators(), NestedModel::derived_init_communicators(), MetaIterator::estimate_by_name(), and MetaIterator::estimate_by_pointer().

int init_evaluation_concurrency (const String & *method_string,* Iterator & *the_iterator,* Model & *the_model*)

convenience function for performing sufficient initialization to define the maximum evaluation concurrency

This is a convenience function for computing the maximum evaluation concurrency prior to concurrent iterator partitioning.

References ParallelLibrary::bcast(), Iterator::is_null(), Iterator::maximum_evaluation_concurrency(), IteratorScheduler::parallelLib, IteratorScheduler::schedPCIter, ParallelLevel::server_communicator_rank(), and ParallelLevel::server_communicator_size().

void schedule_iterators (MetaType & *meta_object,* Iterator & *sub_iterator*)

short convenience function for distributing control among master_dynamic_schedule_iterators(), serve_iterators(), and peer_static_schedule_iterators()

This implementation supports the scheduling of multiple jobs using a single iterator/model pair. Additional future (overloaded) implementations could involve independent iterator instances.

References IteratorScheduler::iteratorScheduling, IteratorScheduler::iteratorServerId, IteratorScheduler::lead_rank(), IteratorScheduler::master_dynamic_schedule_iterators(), IteratorScheduler::numIteratorServers, ParallelLibrary::parallel_configuration_iterator(), IteratorScheduler::parallelLib, IteratorScheduler::peer_static_schedule_iterators(), IteratorScheduler::serve_iterators(), and IteratorScheduler::stop_iterator_servers().

Referenced by CollabHybridMetaIterator::core_run(), EmbedHybridMetaIterator::core_run(), ConcurrentMetaIterator::core_run(), NestedModel::derived_synchronize(), SeqHybridMetaIterator::run_sequential(), and NestedModel::serve_run().

void master_dynamic_schedule_iterators (MetaType & *meta_object*)

executed by the scheduler master to manage a dynamic schedule of iterator jobs among slave iterator servers

This function is adapted from ApplicationInterface::master_dynamic_schedule_evaluations().

References ParallelLibrary::free(), ParallelLibrary::irecv_mi(), ParallelLibrary::isend_mi(), IteratorScheduler::miPLIndex, IteratorScheduler::numIteratorJobs, IteratorScheduler::numIteratorServers, IteratorScheduler::parallelLib, MPIPackBuffer::reset(), MPIUnpackBuffer::resize(), IteratorScheduler::resultsMsgLen, ParallelLibrary::waitall(), and ParallelLibrary::waitsome().

Referenced by IteratorScheduler::schedule_iterators().

void serve_iterators (MetaType & *meta_object,* Iterator & *sub_iterator*)

executed on the slave iterator servers to perform iterator jobs assigned by the scheduler master

This function is similar in structure to ApplicationInterface::serve_evaluations_synch().

References ParallelLibrary::bcast_i(), IteratorScheduler::iteratorCommRank, IteratorScheduler::iteratorCommSize, IteratorScheduler::miPLIndex, ParallelLibrary::parallel_time(), IteratorScheduler::parallelLib, IteratorScheduler::paramsMsgLen, ParallelLibrary::recv_mi(), IteratorScheduler::resultsMsgLen, IteratorScheduler::run_iterator(), and ParallelLibrary::send_mi().

Referenced by IteratorScheduler::schedule_iterators().

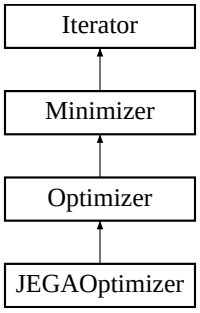
The documentation for this class was generated from the following files:

- IteratorScheduler.hpp
- IteratorScheduler.cpp

13.55 JEGAOptimizer Class Reference

A version of Dakota::Optimizer for instantiation of John Eddy's Genetic Algorithms (JEGA).

Inheritance diagram for JEGAOptimizer:



Classes

- class Driver

  A subclass of the JEGA front end driver that exposes the individual protected methods to execute the algorithm.
- class Evaluator

An evaluator specialization that knows how to interact with Dakota.

- class EvaluatorCreator

  A specialization of the JEGA::FrontEnd::EvaluatorCreator that creates a new instance of a Evaluator.

Public Member Functions

- virtual void find_optimum ()

  Performs the iterations to determine the optimal set of solutions.
- virtual bool accepts_multiple_points () const

  Overridden to return true since JEGA algorithms can accept multiple initial points.
- virtual bool returns_multiple_points () const

  Overridden to return true since JEGA algorithms can return multiple final points.
- virtual void initial_points (const VariablesArray &pts)

  Overridden to assign the _initPts member variable to the passed in collection of Dakota::Variables.
- virtual const VariablesArray & initial_points () const

  Overridden to return the collection of initial points for the JEGA algorithm created and run by this JEGAOptimizer.
- JEGAOptimizer (ProblemDescDB &problem_db, Model &model)

  Constructs a JEGAOptimizer class object.
- ~JEGAOptimizer ()

  Destructs a JEGAOptimizer.

Protected Member Functions

- void LoadDakotaResponses (const JEGA::Utilities::Design &from, Variables &vars, Response &resp) const

  Loads the JEGA-style Design class into equivolent Dakota-style Variables and Response objects.
- void ReCreateTheParameterDatabase ()

  Destroys the current parameter database and creates a new empty one.
- void LoadTheParameterDatabase ()

  Reads information out of the known Dakota::ProblemDescDB and puts it into the current parameter database.
- void LoadAlgorithmConfig (JEGA::FrontEnd::AlgorithmConfig &aConfig)

  Completely initializes the supplied algorithm configuration.
- void LoadProblemConfig (JEGA::FrontEnd::ProblemConfig &pConfig)

  Completely initializes the supplied problem configuration.
- void LoadTheDesignVariables (JEGA::FrontEnd::ProblemConfig &pConfig)

  Adds DesignVariableInfo objects into the problem configuration object.
- void LoadTheObjectiveFunctions (JEGA::FrontEnd::ProblemConfig &pConfig)

  Adds ObjectiveFunctionInfo objects into the problem configuration object.
- void LoadTheConstraints (JEGA::FrontEnd::ProblemConfig &pConfig)

  Adds ConstraintInfo objects into the problem configuration object.
- void GetBestSolutions (const JEGA::Utilities::DesignOFSortSet &from, std::multimap< RealRealPair, JEGA::Utilities::Design * > &designSortMap)

  Returns up to _numBest designs sorted by DAKOTA's fitness (L2 constraint violation, then utopia or objective), taking into account the algorithm type. The front of the returned map can be viewed as a single "best".
- void GetBestMOSolutions (const JEGA::Utilities::DesignOFSortSet &from, std::multimap< RealRealPair, JEGA::Utilities::Design * > &designSortMap)

Retreive the best Designs from a set of solutions assuming that they are generated by a multi objective algorithm.

- void GetBestSOSolutions (const JEGA::Utilities::DesignOFSortSet &from, std::multimap< RealRealPair, JEGA::Utilities::Design ∗ > &designSortMap)

  Retreive the best Designs from a set of solutions assuming that they are generated by a single objective algorithm.

- JEGA::DoubleMatrix ToDoubleMatrix (const VariablesArray &variables) const

  Converts the items in a VariablesArray into a DoubleMatrix whereby the items in the matrix are the design variables.

Private Attributes

- EvaluatorCreator ∗ _theEvalCreator

  A pointer to an EvaluatorCreator used to create the evaluator used by JEGA in Dakota (a JEGAEvaluator).

- JEGA::Utilities::ParameterDatabase ∗ _theParamDB

  A pointer to the ParameterDatabase from which all parameters are retrieved by the created algorithms.

- VariablesArray _initPts

  An array of initial points to use as an initial population.

Additional Inherited Members

13.55.1 Detailed Description

A version of Dakota::Optimizer for instantiation of John Eddy's Genetic Algorithms (JEGA).

This class encapsulates the necessary functionality for creating and properly initializing the JEGA algorithms (MOGA and SOGA).

13.55.2 Constructor & Destructor Documentation

JEGAOptimizer (ProblemDescDB & *problem_db*, Model & *model*)

Constructs a JEGAOptimizer class object.

This method does some of the initialization work for the algorithm. In particular, it initialized the JEGA core.

Parameters

problem_db	The Dakota::ProblemDescDB with information on how the algorithm controls should be set.
model	The Dakota::Model that will be used by this optimizer for problem information, etc.

References JEGAOptimizer::_theEvalCreator, ProblemDescDB::get_int(), ProblemDescDB::get_short(), Iterator::iteratedModel, JEGAOptimizer::LoadTheParameterDatabase(), Iterator::maxEvalConcurrency, Iterator::methodName, Iterator::numFinalSolutions, and Iterator::probDescDB.

13.55.3 Member Function Documentation

void LoadDakotaResponses (const JEGA::Utilities::Design & *from*, Dakota::Variables & *vars*, Dakota::Response & *resp*) const `[protected]`

Loads the JEGA-style Design class into equivolent Dakota-style Variables and Response objects.

This version is meant for the case where a Variables and a Response object exist and just need to be loaded.

Parameters

from	The JEGA Design class object from which to extract the variable and response information for Dakota.
vars	The Dakota::Variables object into which to load the design variable values of *from*.
resp	The Dakota::Response object into which to load the objective function and constraint values of *from*.

References Variables::continuous_variables(), Variables::discrete_int_variables(), Variables::discrete_real_variables(), and Response::function_values().

void LoadTheParameterDatabase ( ) [protected]

Reads information out of the known Dakota::ProblemDescDB and puts it into the current parameter database.

This should be called from the JEGAOptimizer constructor since it is the only time when the problem description database is certain to be configured to supply data for this optimizer.

Referenced by JEGAOptimizer::JEGAOptimizer().

void LoadAlgorithmConfig (JEGA::FrontEnd::AlgorithmConfig & *aConfig*) [protected]

Completely initializes the supplied algorithm configuration.

This loads the supplied configuration object with appropriate data retrieved from the parameter database.

Parameters

aConfig	The algorithm configuration object to load.

void LoadProblemConfig (JEGA::FrontEnd::ProblemConfig & *pConfig*) [protected]

Completely initializes the supplied problem configuration.

This loads the fresh configuration object using the LoadTheDesignVariables, LoadTheObjectiveFunctions, and LoadTheConstraints methods.

Parameters

pConfig	The problem configuration object to load.

void LoadTheDesignVariables (JEGA::FrontEnd::ProblemConfig & *pConfig*) [protected]

Adds DesignVariableInfo objects into the problem configuration object.

This retrieves design variable information from the ParameterDatabase and creates DesignVariableInfo's from it.

Parameters

pConfig	The problem configuration object to load.

References Model::continuous_lower_bounds().

void LoadTheObjectiveFunctions (JEGA::FrontEnd::ProblemConfig & *pConfig*) [protected]

Adds ObjectiveFunctionInfo objects into the problem configuration object.

This retrieves objective function information from the ParameterDatabase and creates ObjectiveFunctionInfo's from it.

Parameters

pConfig	The problem configuration object to load.

void LoadTheConstraints (JEGA::FrontEnd::ProblemConfig & *pConfig*) [protected]

Adds ConstraintInfo objects into the problem configuration object.

This retrieves constraint function information from the ParameterDatabase and creates ConstraintInfo's from it.

Parameters

pConfig	The problem configuration object to load.

References Dakota::asstring(), Dakota::copy_row_vector(), and Model::nonlinear_ineq_constraint_lower_bounds().

void GetBestSolutions (const JEGA::Utilities::DesignOFSortSet & *from*, std::multimap< RealRealPair, JEGA::Utilities::Design * > & *designSortMap*) [protected]

Returns up to _numBest designs sorted by DAKOTA's fitness (L2 constraint violation, then utopia or objective), taking into account the algorithm type. The front of the returned map can be viewed as a single "best".

Parameters

from	The full set of designs returned by the solver.
designSortMap	Map of best solutions with key pair<constraintViolation, fitness>

eventually this functionality must be moved into a separate post-processing application for MO datasets.

void GetBestMOSolutions (const JEGA::Utilities::DesignOFSortSet & *from*, std::multimap< RealRealPair, JEGA::Utilities::Design * > & *designSortMap*) [protected]

Retreive the best Designs from a set of solutions assuming that they are generated by a multi objective algorithm.

eventually this functionality must be moved into a separate post-processing application for MO datasets.

void GetBestSOSolutions (const JEGA::Utilities::DesignOFSortSet & *from*, std::multimap< RealRealPair, JEGA::Utilities::Design * > & *designSortMap*) [protected]

Retreive the best Designs from a set of solutions assuming that they are generated by a single objective algorithm.

eventually this functionality must be moved into a separate post-processing application for MO datasets.

JEGA::DoubleMatrix ToDoubleMatrix (const VariablesArray & *variables*) const [protected]

Converts the items in a VariablesArray into a DoubleMatrix whereby the items in the matrix are the design variables.

The matrix will not contain responses but when being used by Dakota, this doesn't matter. JEGA will attempt to re-evaluate these points but Dakota will recognize that they do not require re-evaluation and thus it will be a cheap operation.

Parameters

variables	The array of DakotaVariables objects to use as the contents of the returned matrix.

Returns

The matrix created using the supplied VariablesArray.

void find_optimum ( ) [virtual]

Performs the iterations to determine the optimal set of solutions.

Override of pure virtual method in Optimizer base class.

The extraction of parameter values actually occurs in this method when the JEGA::FrontEnd::Driver::ExecuteAlgorithm is called. Also the loading of the problem and algorithm configurations occurs in this method. That way, if it is called more than once and the algorithm or problem has changed, it will be accounted for.

Implements Optimizer.

References JEGAOptimizer::Driver::DestroyAlgorithm(), JEGAOptimizer::Driver::ExtractAllData(), and JEGAOptimizer::Driver::PerformIterations().

bool accepts_multiple_points ( ) const [virtual]

Overridden to return true since JEGA algorithms can accept multiple initial points.

Returns

true, always.

Reimplemented from Iterator.

bool returns_multiple_points ( ) const [virtual]

Overridden to return true since JEGA algorithms can return multiple final points.

Returns

true, always.

Reimplemented from Iterator.

void initial_points (const VariablesArray & *pts*) [virtual]

Overridden to assign the _initPts member variable to the passed in collection of Dakota::Variables.

Parameters

pts	The array of initial points for the JEGA algorithm created and run by this JEGAOptimizer.

Reimplemented from Iterator.

const VariablesArray & initial_points ( ) const [virtual]

Overridden to return the collection of initial points for the JEGA algorithm created and run by this JEGAOptimizer.

Returns

The collection of initial points for the JEGA algorithm created and run by this JEGAOptimizer.

Reimplemented from Iterator.

13.55.4 Member Data Documentation

VariablesArray _initPts `[private]`

An array of initial points to use as an initial population.

This member is here to help support the use of JEGA algorithms in Dakota strategies. If this array is populated, then whatever initializer is specified will be ignored and the DoubleMatrix initializer will be used instead on a matrix created from the data in this array.

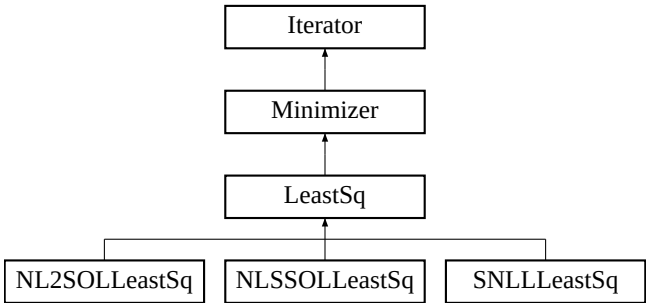
The documentation for this class was generated from the following files:

- JEGAOptimizer.hpp
- JEGAOptimizer.cpp

13.56 LeastSq Class Reference

Base class for the nonlinear least squares branch of the iterator hierarchy.

Inheritance diagram for LeastSq:



Protected Member Functions

- LeastSq ()

  default constructor
- LeastSq (ProblemDescDB &problem_db, Model &model)

  standard constructor
- LeastSq (unsigned short method_name, Model &model)

  alternate "on the fly" constructor
- ~LeastSq ()

  destructor
- void initialize_run ()
- void core_run ()

  core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post
- void post_run (std::ostream &s)
- void finalize_run ()

  utility function to perform common operations following post_run(); deallocation and resetting of instance pointers
- void print_results (std::ostream &s)
- virtual void minimize_residuals ()=0

Used within the least squares branch for minimizing the sum of squares residuals. Redefines the run virtual function for the least squares branch.

- void get_confidence_intervals ()

  Calculate confidence intervals on estimated parameters.

Protected Attributes

- size_t numLeastSqTerms

  number of least squares terms

- LeastSq * prevLSqInstance

  pointer containing previous value of leastSqInstance

- bool weightFlag

  flag indicating whether weighted least squares is active

- RealVector confBoundsLower

  lower bounds for confidence intervals on calibration parameters

- RealVector confBoundsUpper

  upper bounds for confidence intervals on calibration parameters

Static Protected Attributes

- static LeastSq * leastSqInstance

  pointer to LeastSq instance used in static member functions

Private Member Functions

- void weight_model ()

  Wrap iteratedModel in a RecastModel that weights the residuals.

Static Private Member Functions

- static void primary_resp_weighter (const Variables &unweighted_vars, const Variables &weighted_vars, const Response &unweighted_response, Response &weighted_response)

  Recast callback function to weight least squares residuals, gradients, and Hessians.

Additional Inherited Members

13.56.1 Detailed Description

Base class for the nonlinear least squares branch of the iterator hierarchy.

The LeastSq class provides common data and functionality for least squares solvers (including NL2OL, NLSSOLLeastSq, and SNLLLeastSq.

13.56.2 Constructor & Destructor Documentation

LeastSq (ProblemDescDB & *problem_db,* Model & *model*) [protected]

standard constructor

This constructor extracts the inherited data for the least squares branch and performs sanity checking on gradient and constraint settings.

References Dakota::abort_handler(), Iterator::bestVariablesArray, Variables::copy(), Model::current_variables(), Minimizer::data_transform_model(), Iterator::iteratedModel, Iterator::methodName, Minimizer::minimizerRecasts, Minimizer::numIterPrimaryFns, LeastSq::numLeastSqTerms, Minimizer::numRowsExpData, Minimizer::numUserPrimaryFns, Minimizer::obsDataFlag, Minimizer::optimizationFlag, Minimizer::scale_model(), Minimizer::scaleFlag, LeastSq::weight_model(), and LeastSq::weightFlag.

13.56.3 Member Function Documentation

void initialize_run ( ) [protected], [virtual]

This function should be invoked (or reimplemented) by any derived implementations of initialize_run() (which would otherwise hide it).

Reimplemented from Iterator.

Reimplemented in SNLLLeastSq.

References Minimizer::initialize_run(), Iterator::iteratedModel, LeastSq::leastSqInstance, Minimizer::obsDataFlag, LeastSq::prevLSqInstance, Minimizer::scaleFlag, and Model::update_from_subordinate_model().

Referenced by SNLLLeastSq::initialize_run().

void core_run ( ) [inline], [protected], [virtual]

core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post

Virtual run function for the iterator class hierarchy. All derived classes need to redefine it.

Reimplemented from Iterator.

References LeastSq::minimize_residuals().

void post_run (std::ostream & *s*) [protected], [virtual]

Implements portions of post_run specific to LeastSq for scaling back to native variables and functions. This function should be invoked (or reimplemented) by any derived implementations of post_run() (which would otherwise hide it).

Reimplemented from Iterator.

Reimplemented in SNLLLeastSq.

References Dakota::abort_handler(), Response::active_set_request_vector(), Iterator::bestResponseArray, Iterator::bestVariablesArray, Variables::continuous_variables(), Response::copy(), Minimizer::cvScaleMultipliers, Minimizer::cvScaleOffsets, Minimizer::cvScaleTypes, Response::function_value(), Response::function_values(), Iterator::iteratedModel, Minimizer::modify_s2n(), Minimizer::need_resp_trans_byvars(), LeastSq::numLeastSqTerms, Minimizer::numNonlinearConstraints, Minimizer::post_run(), Model::primary_response_fn_weights(), Minimizer::primaryRespScaleFlag, Minimizer::response_modify_s2n(), Minimizer::secondaryRespScaleFlag, Model::subordinate_model(), Response::update_partial(), Minimizer::varsScaleFlag, and LeastSq::weightFlag.

void finalize_run ( ) [inline], [protected], [virtual]

utility function to perform common operations following post_run(); deallocation and resetting of instance pointers

Optional: perform finalization phases of run sequence, like deallocating memory and resetting instance pointers. Commonly used in sub-iterator executions. This is a virtual function; when re-implementing, a derived class must call its nearest parent's finalize_run(), typically *after* performing its own implementation steps.

Reimplemented from Iterator.

Reimplemented in SNLLLeastSq.

References Minimizer::finalize_run(), LeastSq::leastSqInstance, and LeastSq::prevLSqInstance.

Referenced by SNLLLeastSq::finalize_run().

void print_results (std::ostream & *s*) [protected], [virtual]

Redefines default iterator results printing to include nonlinear least squares results (residual terms and constraints).

Reimplemented from Iterator.

References Iterator::activeSet, Minimizer::archive_allocate_best(), Minimizer::archive_best(), Iterator::bestResponseArray, Iterator::bestVariablesArray, LeastSq::confBoundsLower, LeastSq::confBoundsUpper, Model::continuous_variable_labels(), Dakota::data_pairs, Model::interface_id(), Iterator::iteratedModel, Dakota::lookup_by_val(), Minimizer::numContinuousVars, Minimizer::numFunctions, LeastSq::numLeastSqTerms, Model::primary_response_fn_weights(), ActiveSet::request_values(), Model::subordinate_model(), and Dakota::write_precision.

void get_confidence_intervals ( ) [protected]

Calculate confidence intervals on estimated parameters.

Calculate individual confidence intervals for each parameter. These bounds are based on a linear approximation of the nonlinear model.

References Iterator::activeSet, Iterator::bestResponseArray, Iterator::bestVariablesArray, Model::compute_response(), LeastSq::confBoundsLower, LeastSq::confBoundsUpper, Model::continuous_variables(), Model::current_response(), Response::function_gradients(), Iterator::iteratedModel, Minimizer::numContinuousVars, LeastSq::numLeastSqTerms, ActiveSet::request_values(), Minimizer::scaleFlag, and Minimizer::vendorNumericalGradFlag.

Referenced by NL2SOLLeastSq::minimize_residuals(), NLSSOLLeastSq::minimize_residuals(), and SNLLLeastSq::post_run().

void weight_model ( ) [private]

Wrap iteratedModel in a RecastModel that weights the residuals.

Setup Recast for weighting model the weighting transformation doesn't resize, so use numUserPrimaryFns. No vars, active set or secondary mapping. All indices are one-to-one mapped (no change in counts).

References Model::assign_rep(), Iterator::iteratedModel, Minimizer::numContinuousVars, LeastSq::numLeastSqTerms, Minimizer::numNonlinearConstraints, Minimizer::numNonlinearIneqConstraints, Iterator::outputLevel, LeastSq::primary_resp_weighter(), Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), Minimizer::secondary_resp_copier(), and Model::subordinate_model().

Referenced by LeastSq::LeastSq().

void primary_resp_weighter (const Variables & *unweighted_vars,* const Variables & *weighted_vars,* const Response & *unweighted_response,* Response & *weighted_response*) [static], [private]

Recast callback function to weight least squares residuals, gradients, and Hessians.

Apply weights to least squares residuals

References Dakota::_NPOS, Response::active_set_derivative_vector(), Response::active_set_request_vector(), Variables::acv(), Variables::all_continuous_variable_ids(), Variables::continuous_variable_ids(), Variables::cv(), Dakota::find_index(), Response::function_gradients(), Response::function_gradients_view(), Response::function_hessian(),

Response::function_hessian_view(), Response::function_values(), Response::function_values_view(), Variables::icv(), Variables::inactive_continuous_variable_ids(), Iterator::iteratedModel, LeastSq::leastSqInstance, LeastSq::numLeastSqTerms, Iterator::outputLevel, Model::primary_response_fn_weights(), and Model::subordinate_model().

Referenced by LeastSq::weight_model().

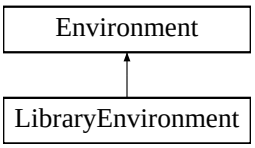
The documentation for this class was generated from the following files:

- DakotaLeastSq.hpp
- DakotaLeastSq.cpp

13.57 LibraryEnvironment Class Reference

Environment corresponding to execution as an embedded library.

Inheritance diagram for LibraryEnvironment:



Public Member Functions

- LibraryEnvironment ()

  default constructor
- LibraryEnvironment (ProgramOptions prog_opts, bool check_bcast_construct=true, DbCallbackFunctionPtr callback=NULL, void *callback_data=NULL)

  Primary constructor: program options typically specifies an input file or input string. Optionally specify a callback function to be invoked after parsing. Set check_bcast_construct if performing late updates and later calling done_modifying_db().
- LibraryEnvironment (MPI_Comm dakota_mpi_comm, ProgramOptions prog_opts=ProgramOptions(), bool check_bcast_construct=true, DbCallbackFunctionPtr callback=NULL, void *callback_data=NULL)

  Alternate constructor accepting communicator, same options as primary.
- ~LibraryEnvironment ()

  destructor
- void insert_nodes (Dakota::DataMethod &dme, Dakota::DataModel &dmo, Dakota::DataVariables &dv, Dakota::DataInterface &di, Dakota::DataResponses &dr)

  Insert DB nodes for a {Method,Model,Variables,Interface,Responses} set.
- void done_modifying_db ()

  Check database contents, broadcast, and construct iterators.
- bool plugin_interface (const String &model_type, const String &interf_type, const String &an_driver, Interface *plugin_iface)

  Plug-in the passed interface into any interface matching the specified (possibly empty) model, interface, and driver strings; returns true if a plugin was performed.
- InterfaceList filtered_interface_list (const String &interf_type, const String &an_driver)

  filter the available Interface instances based on matching interface type and analysis drivers (empty String matches any)

- ModelList filtered_model_list (const String &model_type, const String &interf_type, const String &an_driver)

  filter the available Model instances based on matching model type, interface type, and analysis drivers (empty String matches any)

Additional Inherited Members

13.57.1 Detailed Description

Environment corresponding to execution as an embedded library.

This environment corresponds to use of Dakota as a library within another application, e.g., within library_mode.cpp. It sets up the ParallelLibrary and ProblemDescDB objects without access to command line arguments.

13.57.2 Constructor & Destructor Documentation

LibraryEnvironment (ProgramOptions *prog_opts*, bool *check_bcast_construct* = `true`, DbCallbackFunctionPtr *callback* = `NULL`, void * *callback_data* = `NULL`)

Primary constructor: program options typically specifies an input file or input string. Optionally specify a callback function to be invoked after parsing. Set check_bcast_construct if performing late updates and later calling done_modifying_db().

Construct library environment, optionally performing check/bcast of database and iterator construction

References Environment::construct(), OutputManager::output_startup_message(), Environment::outputManager, and Environment::parse().

LibraryEnvironment (MPI_Comm *dakota_mpi_comm*, ProgramOptions *prog_opts* = ProgramOptions `()`, bool *check_bcast_construct* = `true`, DbCallbackFunctionPtr *callback* = `NULL`, void * *callback_data* = `NULL`)

Alternate constructor accepting communicator, same options as primary.

Construct library environment on passed MPI Comm, optionally performing check/bcast of database and iterator construction. MPI Comm is first argument so client doesn't have to pass all args

References Environment::construct(), OutputManager::output_startup_message(), Environment::outputManager, and Environment::parse().

13.57.3 Member Function Documentation

InterfaceList filtered_interface_list (const String & *interf_type*, const String & *an_driver*)

filter the available Interface instances based on matching interface type and analysis drivers (empty String matches any)

This convenience function helps clients locate and plugin to the right Interface instance for simple cases. Pass an empty string to match any instead of a specific instance

References Interface::analysis_drivers(), Dakota::contains(), Interface::interface_type(), ProblemDescDB::model_list(), and Environment::probDescDB.

ModelList filtered_model_list (const String & *model_type*, const String & *interf_type*, const String & *an_driver*)

filter the available Model instances based on matching model type, interface type, and analysis drivers (empty String matches any)

This convenience function helps clients locate and plugin to the right Interface instance for cases where the parallel configuration is needed in constructing a parallel plugin. Pass an empty string to match any instead of a specific instance

References Interface::analysis_drivers(), Dakota::contains(), Interface::interface_type(), ProblemDescDB::model_list(), and Environment::probDescDB.

Referenced by parallel_interface_plugin(), LibraryEnvironment::plugin_interface(), run_dakota(), and run_dakota_mixed().

The documentation for this class was generated from the following files:

- LibraryEnvironment.hpp
- LibraryEnvironment.cpp

13.58 LightWtBaseConstructor Struct Reference

Dummy struct for overloading constructors used in on-the-fly Model instantiations.

Public Member Functions

- LightWtBaseConstructor (int=0)

  C++ structs can have constructors.

13.58.1 Detailed Description

Dummy struct for overloading constructors used in on-the-fly Model instantiations.

LightWtBaseConstructor is used to overload the constructor used for on-the-fly Model instantiations. Putting this struct here avoids circular dependencies.

The documentation for this struct was generated from the following file:

- dakota_global_defs.hpp

13.59 MatchesWC Struct Reference

Predicate that returns true when the passed path matches the wild_card with which it was configured. Currently supports $*$ and ?.

Public Member Functions

- MatchesWC (const bfs::path &wild_card)

  ctor that builds and stores the regular expression
- bool operator() (const bfs::path &dir_entry)

  return true is dir_entry matches wildCardRegEx

Public Attributes

- boost::basic_regex
  $<$ bfs::path::value_type $>$ wildCardRegEx

  archived RegEx; wchar-based on Windows

13.59.1 Detailed Description

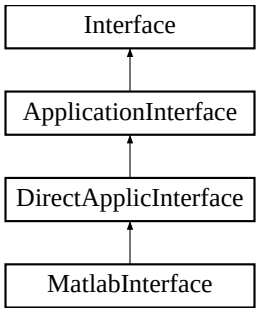
Predicate that returns true when the passed path matches the wild_card with which it was configured. Currently supports ∗ and ?.

The documentation for this struct was generated from the following file:

- WorkdirHelper.hpp

13.60 MatlabInterface Class Reference

Inheritance diagram for MatlabInterface:



Public Member Functions

- MatlabInterface (const ProblemDescDB &problem_db)

  Constructor: start Matlab engine.
- ∼MatlabInterface ()

  Destructor: close Matlab engine.

Protected Member Functions

- virtual int derived_map_ac (const String &ac_name)

  execute an analysis code portion of a direct evaluation invocation
- int matlab_engine_run (const Dakota::String &ac_name)

  Helper function supporting derived_map_ac. Sends data to Matlab, executes analysis, collects return data.
- int matlab_field_prep (mxArray ∗dakota_matlab, const char ∗field_name)

  check that the dakota_matlab strucutre has the specified field_name and add if necessary; free structure memory in preparation for new alloc

Protected Attributes

- engine ∗ matlabEngine

  pointer to the MATLAB engine used for direct evaluations

13.60.1 Detailed Description

Specialization of DirectApplicInterface to link to Matlab analysis drivers. Includes convenience functions to map data to/from Matlab

13.60.2 Member Function Documentation

int derived_map_ac (const String & *ac_name*) [protected], [virtual]

execute an analysis code portion of a direct evaluation invocation

Matlab specialization of dervied analysis components.

Reimplemented from DirectApplicInterface.

References ApplicationInterface::analysisServerId, and MatlabInterface::matlab_engine_run().

int matlab_engine_run (const Dakota::String & *ac_name*) [protected]

Helper function supporting derived_map_ac. Sends data to Matlab, executes analysis, collects return data.

Direct interface to Matlab through Mathworks external API. m-file executed is specified through analysis_drivers, extra strings through analysis_components. (Original BMA 11/28/2005)

Special thanks to Lee Peterson for substantial enhancements 12/15/2007: Added output buffer for the MATLAB command response and error messages Made the Dakota variable persistent in the MATLAB engine workspace Added robustness to the user deleting required Dakota fields

References Dakota::abort_handler(), DirectApplicInterface::analysisComponents, DirectApplicInterface::analysisDriverIndex, Interface::currEvalId, DirectApplicInterface::directFnASV, DirectApplicInterface::directFnDVV, Dakota::FIELD_NAMES, DirectApplicInterface::fnGrads, DirectApplicInterface::fnHessians, Interface::fnLabels, DirectApplicInterface::fnVals, DirectApplicInterface::gradFlag, DirectApplicInterface::hessFlag, MatlabInterface::matlab_field_prep(), MatlabInterface::matlabEngine, DirectApplicInterface::numACV, DirectApplicInterface::numADIV, DirectApplicInterface::numADRV, Dakota::NUMBER_OF_FIELDS, DirectApplicInterface::numDerivVars, DirectApplicInterface::numFns, DirectApplicInterface::numVars, Interface::outputLevel, DirectApplicInterface::xC, DirectApplicInterface::xCLabels, DirectApplicInterface::xDI, DirectApplicInterface::xDILabels, DirectApplicInterface::xDR, and DirectApplicInterface::xDRLabels.

Referenced by MatlabInterface::derived_map_ac().

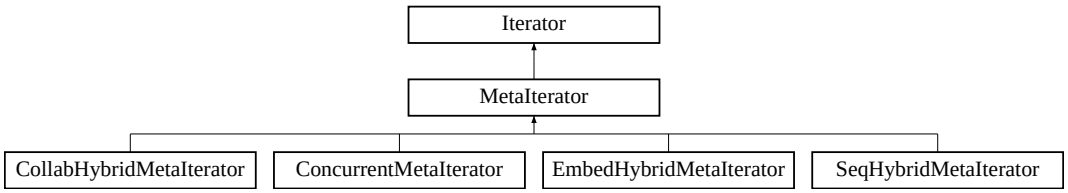
The documentation for this class was generated from the following files:

- MatlabInterface.hpp
- MatlabInterface.cpp

13.61 MetaIterator Class Reference

Base class for meta-iterators.

Inheritance diagram for MetaIterator:



Protected Member Functions

- MetaIterator (ProblemDescDB &problem_db)

  standard constructor
- MetaIterator (ProblemDescDB &problem_db, Model &model)

alternate constructor

- ~MetaIterator ()

  destructor

- void post_run (std::ostream &s)

  post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way

- bool new_model (const String &method_ptr, const String &model_ptr)

  identify presence of a new model specification identified by pointer, necessitating a new instantiation

- void check_model (const String &method_ptr, const String &model_ptr)

  check that any model identified by pointer has the same id as the passed iteratedModel

- void allocate_by_pointer (const String &method_ptr, Iterator &the_iterator, Model &the_model)

  initialize the_iterator and the_model based on method_ptr

- void allocate_by_name (const String &method_string, const String &model_ptr, Iterator &the_iterator, Model &the_model)

  initialize the_iterator based on method_string

- std::pair< int, int > estimate_by_pointer (const String &method_ptr, Iterator &the_iterator, Model &the_-model)

  estimate minimum and maximum processors per iterator needed for init_iterator_parallelism(); instantiates the_-iterator and the_model as needed, but on minimal processor ranks (is later augmented by allocate_by_pointer())

- std::pair< int, int > estimate_by_name (const String &method_string, const String &model_ptr, Iterator &the_iterator, Model &the_model)

  estimate minimum and maximum processors per iterator needed for init_iterator_parallelism(); instantiates the_-iterator and the_model as needed, but on minimal processor ranks (is later augmented by allocate_by_name())

Protected Attributes

- IteratorScheduler iterSched

  scheduler for concurrent execution of Iterators

- int maxIteratorConcurrency

  maximum number of concurrent sub-iterator executions

Additional Inherited Members

13.61.1 Detailed Description

Base class for meta-iterators.

This base class shares code for concurrent and hybrid meta-iterators, where the former supports multi-start and Pareto set iteration and the latter supports sequential, embedded, and collaborative hybrids.

13.61.2 Member Function Documentation

void post_run (std::ostream & *s*) `[protected], [virtual]`

post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way

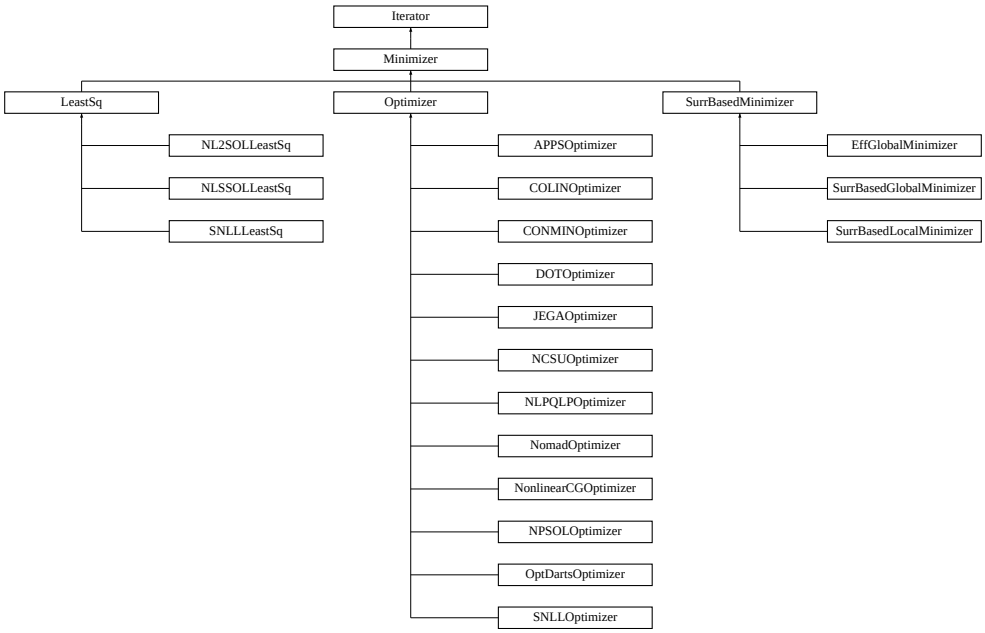
Post-run phase, which a derived iterator may optionally reimplement; when not present, post-run is likely integrated into run. This is a virtual function; when re-implementing, a derived class must call its nearest parent's post_run(), typically *after* performing its own implementation steps.

Reimplemented from Iterator.
References MetaIterator::iterSched, IteratorScheduler::lead_rank(), and Iterator::print_results().
The documentation for this class was generated from the following files:

- MetaIterator.hpp
- MetaIterator.cpp

13.62 Minimizer Class Reference

Base class for the optimizer and least squares branches of the iterator hierarchy.
Inheritance diagram for Minimizer:



Public Member Functions

- void constraint_tolerance (Real constr_tol)

  set the method constraint tolerance (constraintTol)
- Real constraint_tolerance () const

  return the method constraint tolerance (constraintTol)

Protected Member Functions

- Minimizer ()

  default constructor
- Minimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor
- Minimizer (unsigned short method_name, Model &model)

alternate constructor for "on the fly" instantiations

- Minimizer (unsigned short method_name, size_t num_lin_ineq, size_t num_lin_eq, size_t num_nln_ineq, size_t num_nln_eq)

  alternate constructor for "on the fly" instantiations

- ~Minimizer ()

  destructor

- void update_from_model (const Model &model)

  set inherited data attributes based on extractions from incoming model

- void initialize_run ()

  utility function to perform common operations prior to pre_run(); typically memory initialization; setting of instance pointers

- void post_run (std::ostream &s)

  post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way

- void finalize_run ()

  utility function to perform common operations following post_run(); deallocation and resetting of instance pointers

- const Model & algorithm_space_model () const
- bool data_transform_model (bool weight_flag=false)

  Wrap iteratedModel in a RecastModel that subtracts provided observed data from the primary response functions (variables and secondary responses are unchanged)

- void scale_model ()

  Wrap iteratedModel in a RecastModel that performs variable and/or response scaling.

- bool need_resp_trans_byvars (const ShortArray &asv, int start_index, int num_resp)

  determine if response transformation is needed due to variable transformations

- RealVector modify_s2n (const RealVector &scaled_vars, const IntArray &scale_types, const RealVector &multipliers, const RealVector &offsets) const

  general RealVector mapping from scaled to native variables (and values)

- void response_modify_s2n (const Variables &native_vars, const Response &scaled_response, Response &native_response, int start_offset, int num_responses) const

  map responses from scaled to native space

- Real objective (const RealVector &fn_vals, const BoolDeque &max_sense, const RealVector &primary_wts) const

  compute a composite objective value from one or more primary functions

- Real objective (const RealVector &fn_vals, size_t num_fns, const BoolDeque &max_sense, const RealVector &primary_wts) const

  compute a composite objective with specified number of source primary functions, instead of userPrimaryFns

- void objective_gradient (const RealVector &fn_vals, const RealMatrix &fn_grads, const BoolDeque &max_sense, const RealVector &primary_wts, RealVector &obj_grad) const

  compute the gradient of the composite objective function

- void objective_gradient (const RealVector &fn_vals, size_t num_fns, const RealMatrix &fn_grads, const BoolDeque &max_sense, const RealVector &primary_wts, RealVector &obj_grad) const

  compute the gradient of the composite objective function

- void objective_hessian (const RealVector &fn_vals, const RealMatrix &fn_grads, const RealSymMatrixArray &fn_hessians, const BoolDeque &max_sense, const RealVector &primary_wts, RealSymMatrix &obj_hess) const

compute the Hessian of the composite objective function

- void objective_hessian (const RealVector &fn_vals, size_t num_fns, const RealMatrix &fn_grads, const RealSymMatrixArray &fn_hessians, const BoolDeque &max_sense, const RealVector &primary_wts, RealSymMatrix &obj_hess) const

  compute the Hessian of the composite objective function
- void archive_allocate_best (size_t num_points)

  allocate results arrays and labels for multipoint storage
- void archive_best (size_t index, const Variables &best_vars, const Response &best_resp)

  archive the best point into the results array
- void resize_best_vars_array (size_t newsize)

  Safely resize the best variables array to newsize taking into account the envelope-letter design pattern and any recasting.
- void resize_best_resp_array (size_t newsize)

  Safely resize the best response array to newsize taking into account the envelope-letter design pattern and any recasting.

Static Protected Member Functions

- static void gnewton_set_recast (const Variables &recast_vars, const ActiveSet &recast_set, ActiveSet &sub_model_set)

  conversion of request vector values for the Gauss-Newton Hessian approximation
- static void replicate_set_recast (const Variables &recast_vars, const ActiveSet &recast_set, ActiveSet &sub_model_set)

  conversion of request vector values for Least Squares
- static void secondary_resp_copier (const Variables &input_vars, const Variables &output_vars, const Response &input_response, Response &output_response)

  copy the partial response for secondary functions when needed (data and reduction transforms)

Protected Attributes

- size_t numFunctions

  number of response functions
- size_t numContinuousVars

  number of active continuous vars
- size_t numDiscreteIntVars

  number of active discrete integer vars
- size_t numDiscreteStringVars

  number of active discrete string vars
- size_t numDiscreteRealVars

  number of active discrete real vars
- Real constraintTol

  optimizer/least squares constraint tolerance
- Real bigRealBoundSize

  cutoff value for inequality constraint and continuous variable bounds
- int bigIntBoundSize

cutoff value for discrete variable bounds

- size_t numNonlinearIneqConstraints

  number of nonlinear inequality constraints
- size_t numNonlinearEqConstraints

  number of nonlinear equality constraints
- size_t numLinearIneqConstraints

  number of linear inequality constraints
- size_t numLinearEqConstraints

  number of linear equality constraints
- size_t numNonlinearConstraints

  total number of nonlinear constraints
- size_t numLinearConstraints

  total number of linear constraints
- size_t numConstraints

  total number of linear and nonlinear constraints
- unsigned short minimizerRecasts

  number of RecastModels locally (in Minimizer or derived classes) layered over the initially passed in Model
- bool optimizationFlag

  flag for use where optimization and NLS must be distinguished
- size_t numUserPrimaryFns

  number of objective functions or least squares terms in the user's model always initialize at Minimizer, even if overridden later
- size_t numIterPrimaryFns

  number of objective functions or least squares terms in iterator's view always initialize at Minimizer, even if overridden later
- bool boundConstraintFlag

  convenience flag for denoting the presence of user-specified bound constraints. Used for method selection and error checking.
- bool speculativeFlag

  flag for speculative gradient evaluations
- String obsDataFilename

  filename from which to read observed data
- bool obsDataFlag

  flag indicating whether user-supplied data is active
- ExperimentData expData

  Container for experimental data to which to calibrate model using least squares or other formulations which minimize SSE.
- size_t numExperiments

  number of experiments
- size_t numRowsExpData

  number of total rows of data since we are allowing varying numbers of experiments and replicates per experiment
- bool scaleFlag

  flag for overall scaling status
- bool varsScaleFlag

flag for variables scaling

- bool primaryRespScaleFlag

  flag for primary response scaling

- bool secondaryRespScaleFlag

  flag for secondary response scaling

- IntArray cvScaleTypes

  scale flags for continuous vars.

- RealVector cvScaleMultipliers

  scales for continuous variables

- RealVector cvScaleOffsets

  offsets for continuous variables

- IntArray responseScaleTypes

  scale flags for all responses

- RealVector responseScaleMultipliers

  scales for all responses

- RealVector responseScaleOffsets

  `offsets for all responses (zero` < *for functions, not for nonlin con)*

- IntArray linearIneqScaleTypes

  scale flags for linear ineq

- RealVector linearIneqScaleMultipliers

  scales for linear ineq constrs.

- RealVector linearIneqScaleOffsets

  offsets for linear ineq constrs.

- IntArray linearEqScaleTypes

  scale flags for linear eq.

- RealVector linearEqScaleMultipliers

  scales for linear constraints

- RealVector linearEqScaleOffsets

  offsets for linear constraints

- Minimizer ∗ prevMinInstance

  pointer containing previous value of minimizerInstance

- bool vendorNumericalGradFlag

  convenience flag for gradient_type == numerical && method_source == vendor

Static Protected Attributes

- static Minimizer ∗ minimizerInstance

  pointer to Minimizer used in static member functions

Private Member Functions

- bool data_difference_core (const Response &raw_response, Response &residual_response)

  Core of data difference, which doesn't perform any output.
- void initialize_scaling ()

  initialize scaling types, multipliers, and offsets; perform error checking
- void compute_scaling (int object_type, int auto_type, int num_vars, RealVector &lbs, RealVector &ubs, RealVector &targets, const StringArray &scale_strings, const RealVector &scales, IntArray &scale_types, RealVector &scale_mults, RealVector &scale_offsets)

  general helper function for initializing scaling types and factors on a vector of variables, functions, constraints, etc.
- bool compute_scale_factor (const Real lower_bound, const Real upper_bound, Real *multiplier, Real *offset)

  automatically compute a single scaling factor – bounds case
- bool compute_scale_factor (const Real target, Real *multiplier)

  automatically compute a single scaling factor – target case
- void response_scaler_core (const Variables &native_vars, const Variables &scaled_vars, const Response &native_response, Response &iterator_response, size_t start_offset, size_t num_responses)

  Core of response scaling, which doesn't perform any output.
- RealVector modify_n2s (const RealVector &native_vars, const IntArray &scale_types, const RealVector &multipliers, const RealVector &offsets) const

  general RealVector mapping from native to scaled variables vectors:
- void response_modify_n2s (const Variables &scaled_vars, const Response &native_response, Response &scaled_response, int start_offset, int num_responses) const

  map reponses from native to scaled variable space
- RealMatrix lin_coeffs_modify_n2s (const RealMatrix &native_coeffs, const RealVector &cv_multipliers, const RealVector &lin_multipliers) const

  general linear coefficients mapping from native to scaled space
- void print_scaling (const String &info, const IntArray &scale_types, const RealVector &scale_mults, const RealVector &scale_offsets, const StringArray &labels)

  print scaling information for a particular response type in tabular form

Static Private Member Functions

- static void primary_resp_differencer (const Variables &raw_vars, const Variables &residual_vars, const Response &raw_response, Response &residual_response)

  Recast callback function to difference residuals with observed data.
- static void variables_scaler (const Variables &scaled_vars, Variables &native_vars)

  RecastModel callback for variables scaling: transform variables from scaled to native (user) space.
- static void primary_resp_scaler (const Variables &native_vars, const Variables &scaled_vars, const Response &native_response, Response &iterator_response)

  RecastModel callback for primary response scaling: transform responses (grads, Hessians) from native (user) to scaled space.
- static void secondary_resp_scaler (const Variables &native_vars, const Variables &scaled_vars, const Response &native_response, Response &scaled_response)

  RecastModel callback for secondary response scaling: transform constraints (grads, Hessians) from native (user) to scaled space.

Friends

- class SOLBase

  the SOLBase class is not derived the iterator hierarchy but still needs access to iterator hierarchy data (to avoid attribute replication)

- class SNLLBase

  the SNLLBase class is not derived the iterator hierarchy but still needs access to iterator hierarchy data (to avoid attribute replication)

13.62.1 Detailed Description

Base class for the optimizer and least squares branches of the iterator hierarchy.

The Minimizer class provides common data and functionality for Optimizer and LeastSq.

13.62.2 Constructor & Destructor Documentation

Minimizer (ProblemDescDB & *problem_db*, Model & *model*) `[protected]`

standard constructor

This constructor extracts inherited data for the optimizer and least squares branches and performs sanity checking on constraint settings.

References Iterator::iteratedModel, Iterator::maxIterations, Iterator::methodName, Iterator::numFinalSolutions, and Minimizer::update_from_model().

13.62.3 Member Function Documentation

void initialize_run ( ) `[protected], [virtual]`

utility function to perform common operations prior to pre_run(); typically memory initialization; setting of instance pointers

Perform initialization phases of run sequence, like allocating memory and setting instance pointers. Commonly used in sub-iterator executions. This is a virtual function; when re-implementing, a derived class must call its nearest parent's initialize_run(), typically *before* performing its own implementation steps.

Reimplemented from Iterator.

Reimplemented in SNLLOptimizer, NLPQLPOptimizer, SNLLLeastSq, DOTOptimizer, and Optimizer.

References Model::all_continuous_variables(), Model::all_discrete_int_variables(), Model::all_discrete_real_variables(), Iterator::bestVariablesArray, Model::is_null(), Iterator::iteratedModel, Minimizer::minimizerInstance, Minimizer::minimizerRecasts, Minimizer::prevMinInstance, Model::set_evaluation_reference(), Iterator::subIteratorFlag, Model::subordinate_model(), and Iterator::summaryOutputFlag.

Referenced by LeastSq::initialize_run(), and Optimizer::initialize_run().

void post_run (std::ostream & *s*) `[protected], [virtual]`

post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/Responses and perform final analysis phase in a standalone way

Post-run phase, which a derived iterator may optionally reimplement; when not present, post-run is likely integrated into run. This is a virtual function; when re-implementing, a derived class must call its nearest parent's post_run(), typically *after* performing its own implementation steps.

Reimplemented from Iterator.

Reimplemented in SNLLOptimizer, SNLLLeastSq, and Optimizer.

References Model::is_null(), Iterator::iteratedModel, Model::print_evaluation_summary(), Iterator::print_results(), Iterator::resultsDB, Iterator::summaryOutputFlag, and ResultsManager::write_databases().

Referenced by LeastSq::post_run(), Optimizer::post_run(), and SNLLLeastSq::post_run().

void finalize_run ( ) [inline], [protected], [virtual]

utility function to perform common operations following post_run(); deallocation and resetting of instance pointers

Optional: perform finalization phases of run sequence, like deallocating memory and resetting instance pointers. Commonly used in sub-iterator executions. This is a virtual function; when re-implementing, a derived class must call its nearest parent's finalize_run(), typically *after* performing its own implementation steps.

Reimplemented from Iterator.

Reimplemented in SNLLOptimizer, SNLLLeastSq, and Optimizer.

References Iterator::finalize_run(), Minimizer::minimizerInstance, and Minimizer::prevMinInstance.

Referenced by LeastSq::finalize_run(), and Optimizer::finalize_run().

const Model & algorithm_space_model ( ) const [inline], [protected], [virtual]

default definition that gets redefined in selected derived Minimizers

Reimplemented from Iterator.

Reimplemented in EffGlobalMinimizer.

References Iterator::iteratedModel.

bool data_transform_model (bool *weight_flag* = `false`) [protected]

Wrap iteratedModel in a RecastModel that subtracts provided observed data from the primary response functions (variables and secondary responses are unchanged)

Reads observation data to compute least squares residuals. Does not change size of responses, and is the first wrapper, therefore sizes are based on iteratedModel. This will set weights to sigma[i]^-2 if appropriate. weight_flag is true is there already exist user-specified weights in the calling context.

References Dakota::abort_handler(), Iterator::activeSet, Model::assign_rep(), Minimizer::expData, ProblemDescDB::get_bool(), ProblemDescDB::get_sizet(), Iterator::iteratedModel, ExperimentData::load_scalar(), Minimizer::numContinuousVars, Minimizer::numExperiments, Minimizer::numFunctions, Minimizer::numNonlinearConstraints, Minimizer::numNonlinearIneqConstraints, Minimizer::numRowsExpData, Minimizer::numUserPrimaryFns, Minimizer::obsDataFilename, Iterator::outputLevel, Minimizer::primary_resp_differencer(), Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), Iterator::probDescDB, Minimizer::replicate_set_recast(), ActiveSet::request_vector(), ExperimentData::scalar_data(), ExperimentData::scalar_sigma(), Minimizer::secondary_resp_copier(), and Model::subordinate_model().

Referenced by LeastSq::LeastSq(), and Optimizer::Optimizer().

void scale_model ( ) [protected]

Wrap iteratedModel in a RecastModel that performs variable and/or response scaling.

Wrap the iteratedModel in a scaling transformation, such that iteratedModel now contains a scaling recast model. Potentially affects variables, primary, and secondary responses

References Model::assign_rep(), Minimizer::cvScaleTypes, RecastModel::initialize(), Minimizer::initialize_scaling(), Iterator::iteratedModel, Model::model_rep(), Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, Minimizer::numNonlinearIneqConstraints, Minimizer::numUserPrimaryFns, Iterator::outputLevel, Minimizer::primary_resp_scaler(), Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), Minimizer::primaryRespScaleFlag, Minimizer::responseScaleTypes, Minimizer::secondary_resp_scaler(), Minimizer::secondaryRespScaleFlag, Model::subordinate_model(), Minimizer::variables_scaler(), and Minimizer::varsScaleFlag.

Referenced by LeastSq::LeastSq(), and Optimizer::Optimizer().

void gnewton_set_recast (const Variables & *recast_vars,* const ActiveSet & *recast_set,* ActiveSet & *sub_model_set*) [static], [protected]

conversion of request vector values for the Gauss-Newton Hessian approximation

For Gauss-Newton Hessian requests, activate the 2 bit and mask the 4 bit.

References ActiveSet::request_value(), and ActiveSet::request_vector().

Referenced by Optimizer::reduce_model(), and SurrBasedLocalMinimizer::SurrBasedLocalMinimizer().

void secondary_resp_copier (const Variables & *input_vars,* const Variables & *output_vars,* const Response & *input_response,* Response & *output_response*) [static], [protected]

copy the partial response for secondary functions when needed (data and reduction transforms)

Constraint function map from user/native space to iterator/scaled/combined space using a RecastModel.

References Minimizer::minimizerInstance, Minimizer::numIterPrimaryFns, Minimizer::numNonlinearConstraints, Minimizer::numUserPrimaryFns, and Response::update_partial().

Referenced by Minimizer::data_transform_model(), Optimizer::reduce_model(), and LeastSq::weight_model().

bool need_resp_trans_byvars (const ShortArray & *asv,* int *start_index,* int *num_resp*) [protected]

determine if response transformation is needed due to variable transformations

Determine if variable transformations present and derivatives requested, which implies a response transformation is necessay

References Minimizer::varsScaleFlag.

Referenced by LeastSq::post_run(), Optimizer::post_run(), SNLLLeastSq::post_run(), and Minimizer::response_scaler_core().

RealVector modify_s2n (const RealVector & *scaled_vars,* const IntArray & *scale_types,* const RealVector & *multipliers,* const RealVector & *offsets*) const [protected]

general RealVector mapping from scaled to native variables (and values)

general RealVector mapping from scaled to native variables and/or vals; loosely, in greatest generality: scaled_var = (LOG_BASE^scaled_var) ∗ multiplier + offset

Referenced by LeastSq::post_run(), Optimizer::post_run(), SNLLLeastSq::post_run(), and Minimizer::variables_scaler().

void response_modify_s2n (const Variables & *native_vars,* const Response & *scaled_response,* Response & *native_response,* int *start_offset,* int *num_responses*) const [protected]

map responses from scaled to native space

Unscaling response mapping: modifies response from scaled (iterator) to native (user) space. Maps num_responses starting at response_offset

References Response::active_set(), Variables::acv(), Variables::all_continuous_variable_ids(), Variables::all_continuous_variables(), Variables::continuous_variable_ids(), Variables::continuous_variables(), Dakota::copy_data(), Variables::cv(), Minimizer::cvScaleMultipliers, Minimizer::cvScaleOffsets, Minimizer::cvScaleTypes, ActiveSet::derivative_vector(), Dakota::find_index(), Response::function_gradient_view(), Response::function_gradients(), Response::function_hessian_view(), Response::function_hessians(), Response::function_labels(), Response::function_value(), Response::function_values(), Variables::icv(), Variables::inactive_continuous_variable_ids(), Variables::inactive_continuous_variables(), Minimizer::numUserPrimaryFns, Iterator::outputLevel, ActiveSet::request_vector(), Minimizer::responseScaleMultipliers, Minimizer::responseScaleOffsets, Minimizer::responseScaleTypes, and Dakota::write_precision.

Referenced by LeastSq::post_run(), and Optimizer::post_run().

Real objective (const RealVector & *fn_vals,* const BoolDeque & *max_sense,* const RealVector & *primary_wts*) const [protected]

compute a composite objective value from one or more primary functions

The composite objective computation sums up the contributions from one of more primary functions using the primary response fn weights.

References Minimizer::numUserPrimaryFns.

Referenced by SurrBasedLocalMinimizer::approx_subprob_objective_eval(), SurrBasedMinimizer::augmented_lagrangian_merit(), EffGlobalMinimizer::expected_improvement(), SurrBasedMinimizer::lagrangian_merit(), Optimizer::objective_reduction(), SurrBasedMinimizer::penalty_merit(), COLINOptimizer::post_run(), SurrBasedMinimizer::update_filter(), and SurrBasedLocalMinimizer::update_penalty().

Real objective (const RealVector & *fn_vals,* size_t *num_fns,* const BoolDeque & *max_sense,* const RealVector & *primary_wts*) const [protected]

compute a composite objective with specified number of source primary functions, instead of userPrimaryFns

This "composite" objective is a more general case of the previous objective(), but doesn't presume a reduction map from user to iterated space. Used to apply weights and sense in COLIN results sorting. Leaving as a duplicate implementation pending resolution of COLIN lookups.

References Minimizer::optimizationFlag.

void objective_gradient (const RealVector & *fn_vals,* size_t *num_fns,* const RealMatrix & *fn_grads,* const BoolDeque & *max_sense,* const RealVector & *primary_wts,* RealVector & *obj_grad*) const [protected]

compute the gradient of the composite objective function

The composite objective gradient computation combines the contributions from one of more primary function gradients, including the effect of any primary function weights. In the case of a linear mapping (MOO), only the primary function gradients are required, but in the case of a nonlinear mapping (NLS), primary function values are also needed. Within RecastModel::set_mapping(), the active set requests are automatically augmented to make values available when needed, based on nonlinearRespMapping settings.

References Minimizer::numContinuousVars, and Minimizer::optimizationFlag.

void objective_hessian (const RealVector & *fn_vals,* size_t *num_fns,* const RealMatrix & *fn_grads,* const RealSymMatrixArray & *fn_hessians,* const BoolDeque & *max_sense,* const RealVector & *primary_wts,* RealSymMatrix & *obj_hess*) const [protected]

compute the Hessian of the composite objective function

The composite objective Hessian computation combines the contributions from one of more primary function Hessians, including the effect of any primary function weights. In the case of a linear mapping (MOO), only the primary function Hessians are required, but in the case of a nonlinear mapping (NLS), primary function values and gradients are also needed in general (gradients only in the case of a Gauss-Newton approximation). Within the default RecastModel::set_mapping(), the active set requests are automatically augmented to make values and gradients available when needed, based on nonlinearRespMapping settings.

References Dakota::abort_handler(), Minimizer::numContinuousVars, and Minimizer::optimizationFlag.

void resize_best_vars_array (size_t *newsize*) [protected]

Safely resize the best variables array to newsize taking into account the envelope-letter design pattern and any recasting.

Uses data from the innermost model, should any Minimizer recasts be active. Called by multipoint return solvers. Do not directly call resize on the bestVariablesArray object unless you intend to share the internal content (letter) with other objects after assignment.

References Iterator::bestVariablesArray, Variables::copy(), Model::current_variables(), Iterator::iteratedModel, Minimizer::minimizerRecasts, and Model::subordinate_model().

Referenced by COLINOptimizer::post_run().

void resize_best_resp_array (size_t *newsize*) [protected]

Safely resize the best response array to newsize taking into account the envelope-letter design pattern and any recasting.

Uses data from the innermost model, should any Minimizer recasts be active. Called by multipoint return solvers. Do not directly call resize on the bestResponseArray object unless you intend to share the internal content (letter) with other objects after assignment.

References Iterator::bestResponseArray, Response::copy(), Model::current_response(), Iterator::iteratedModel, Minimizer::minimizerRecasts, and Model::subordinate_model().

Referenced by COLINOptimizer::post_run().

void primary_resp_differencer (const Variables & *raw_vars,* const Variables & *residual_vars,* const Response & *raw_response,* Response & *residual_response*) [static], [private]

Recast callback function to difference residuals with observed data.

Difference the primary responses with observed data

References Minimizer::data_difference_core(), Response::function_labels(), Response::function_values(), Minimizer::minimizerInstance, and Iterator::outputLevel.

Referenced by Minimizer::data_transform_model().

void initialize_scaling ( ) [private]

initialize scaling types, multipliers, and offsets; perform error checking

Initialize scaling types, multipliers, and offsets. Update the iteratedModel appropriately

References Dakota::abort_handler(), Minimizer::compute_scaling(), Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variable_labels(), Model::continuous_variables(), Dakota::copy_data(), Minimizer::cvScaleMultipliers, Minimizer::cvScaleOffsets, Minimizer::cvScaleTypes, ProblemDescDB::get_rv(), ProblemDescDB::get_sa(), Iterator::iteratedModel, Minimizer::lin_coeffs_modify_n2s(), Model::linear_eq_constraint_coeffs(), Model::linear_eq_constraint_targets(), Model::linear_ineq_constraint_coeffs(), Model::linear_ineq_constraint_lower_bounds(), Model::linear_ineq_constraint_upper_bounds(), Minimizer::linearEqScaleMultipliers, Minimizer::linearEqScaleOffsets, Minimizer::linearEqScaleTypes, Minimizer::linearIneqScaleMultipliers, Minimizer::linearIneqScaleOffsets, Minimizer::linearIneqScaleTypes, Model::model_rep(), Minimizer::modify_n2s(), Model::nonlinear_eq_constraint_targets(), Model::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Minimizer::numContinuousVars, Minimizer::numFunctions, Minimizer::numLinearEqConstraints, Minimizer::numLinearIneqConstraints, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, Minimizer::numUserPrimaryFns, Iterator::outputLevel, Minimizer::primaryRespScaleFlag, Minimizer::print_scaling(), Iterator::probDescDB, Model::response_labels(), Minimizer::responseScaleMultipliers, Minimizer::responseScaleOffsets, Minimizer::responseScaleTypes, Minimizer::secondaryRespScaleFlag, RecastModel::submodel_supports_derivative_estimation(), Model::subordinate_model(), Model::supports_derivative_estimation(), and Minimizer::varsScaleFlag.

Referenced by Minimizer::scale_model().

void variables_scaler (const Variables & *scaled_vars,* Variables & *native_vars*) `[static], [private]`

RecastModel callback for variables scaling: transform variables from scaled to native (user) space.

Variables map from iterator/scaled space to user/native space using a RecastModel.

References Variables::continuous_variable_labels(), Variables::continuous_variables(), Minimizer::cvScaleMultipliers, Minimizer::cvScaleOffsets, Minimizer::cvScaleTypes, Minimizer::minimizerInstance, Minimizer::modify_s2n(), and Iterator::outputLevel.

Referenced by Minimizer::scale_model().

void secondary_resp_scaler (const Variables & *native_vars,* const Variables & *scaled_vars,* const Response & *native_response,* Response & *iterator_response*) `[static], [private]`

RecastModel callback for secondary response scaling: transform constraints (grads, Hessians) from native (user) to scaled space.

Constraint function map from user/native space to iterator/scaled/combined space using a RecastModel.

References Minimizer::minimizerInstance, Minimizer::numNonlinearConstraints, Minimizer::numUserPrimaryFns, and Minimizer::response_scaler_core().

Referenced by Minimizer::scale_model().

RealVector modify_n2s (const RealVector & *native_vars,* const IntArray & *scale_types,* const RealVector & *multipliers,* const RealVector & *offsets*) const `[private]`

general RealVector mapping from native to scaled variables vectors:

general RealVector mapping from native to scaled variables; loosely, in greatest generality: scaled_var = log((native_var - offset) / multiplier)

Referenced by Minimizer::initialize_scaling().

void response_modify_n2s (const Variables & *native_vars,* const Response & *native_response,* Response & *recast_response,* int *start_offset,* int *num_responses*) const `[private]`

map reponses from native to scaled variable space

Scaling response mapping: modifies response from a model (user/native) for use in iterators (scaled). Maps num_responses starting at response_offset

References Response::active_set(), Variables::acv(), Variables::all_continuous_variable_ids(), Variables::all_continuous_variables(), Variables::continuous_variable_ids(), Variables::continuous_variables(), Dakota::copy_data(), Variables::cv(), Minimizer::cvScaleMultipliers, Minimizer::cvScaleOffsets, Minimizer::cvScaleTypes, ActiveSet::derivative_vector(), Dakota::find_index(), Response::function_gradient_view(), Response::function_gradients(), Response::function_hessian_view(), Response::function_hessians(), Response::function_labels(), Response::function_value(), Response::function_values(), Variables::icv(), Variables::inactive_continuous_variable_ids(), Variables::inactive_continuous_variables(), Minimizer::numUserPrimaryFns, Iterator::outputLevel, ActiveSet::request_vector(), Minimizer::responseScaleMultipliers, Minimizer::responseScaleOffsets, Minimizer::responseScaleTypes, and Dakota::write_precision.

Referenced by Minimizer::response_scaler_core().

RealMatrix lin_coeffs_modify_n2s (const RealMatrix & *src_coeffs,* const RealVector & *cv_multipliers,* const RealVector & *lin_multipliers*) const `[private]`

general linear coefficients mapping from native to scaled space

compute scaled linear constraint matrix given design variable multipliers and linear scaling multipliers. Only scales components corresponding to continuous variables so for src_coeffs of size MxN, lin_multipliers.size() <= M, cv_multipliers.size() <= N

Referenced by Minimizer::initialize_scaling().

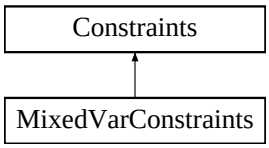
The documentation for this class was generated from the following files:

- DakotaMinimizer.hpp
- DakotaMinimizer.cpp

13.63 MixedVarConstraints Class Reference

Derived class within the Constraints hierarchy which separates continuous and discrete variables (no domain type array merging).

Inheritance diagram for MixedVarConstraints:



Public Member Functions

- MixedVarConstraints (const SharedVariablesData &svd)

  lightweight constructor
- MixedVarConstraints (const ProblemDescDB &problem_db, const SharedVariablesData &svd)

  standard constructor
- ~MixedVarConstraints ()

  destructor
- void write (std::ostream &s) const

  write a variable constraints object to an std::ostream
- void read (std::istream &s)

  read a variable constraints object from an std::istream

Additional Inherited Members

13.63.1 Detailed Description

Derived class within the Constraints hierarchy which separates continuous and discrete variables (no domain type array merging).

Derived variable constraints classes take different views of the design, uncertain, and state variable types and the continuous and discrete domain types. The MixedVarConstraints derived class separates the continuous and discrete domain types (see Variables::get_variables(problem_db) for variables type selection; variables type is passed to the Constraints constructor in Model).

13.63.2 Constructor & Destructor Documentation

MixedVarConstraints (const ProblemDescDB & *problem_db*, const SharedVariablesData & *svd*)

standard constructor

In this class, mixed continuous/discrete variables are used. Most iterators/strategies use this approach, which is the default in Constraints::get_constraints().

References Constraints::allContinuousLowerBnds, Constraints::allContinuousUpperBnds, Constraints::allDiscreteIntLowerBnds, Constraints::allDiscreteIntUpperBnds, Constraints::allDiscreteRealLowerBnds, Constraints::allDiscreteRealUpperBnds, Dakota::copy_data_partial(), ProblemDescDB::get_iv(), ProblemDescDB::get_rv(), Constraints::sharedVarsData, and SharedVariablesData::view().

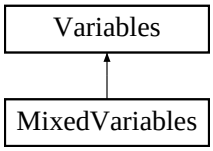
The documentation for this class was generated from the following files:

- MixedVarConstraints.hpp
- MixedVarConstraints.cpp

13.64 MixedVariables Class Reference

Derived class within the Variables hierarchy which separates continuous and discrete variables (no domain type array merging).

Inheritance diagram for MixedVariables:



Public Member Functions

- MixedVariables (const ProblemDescDB &problem_db, const std::pair< short, short > &view)

  standard constructor
- MixedVariables (const SharedVariablesData &svd)

  lightweight constructor
- ~MixedVariables ()

  destructor

Protected Member Functions

- void read (std::istream &s)

  read a variables object from an std::istream
- void write (std::ostream &s) const

  write a variables object to an std::ostream
- void write_aprepro (std::ostream &s) const

  write a variables object to an std::ostream in aprepro format
- void read_tabular (std::istream &s, bool active_only=false)
- void write_tabular (std::ostream &s, bool active_only=false) const

  write a variables object in tabular format to an std::ostream

Additional Inherited Members

13.64.1 Detailed Description

Derived class within the Variables hierarchy which separates continuous and discrete variables (no domain type array merging).

Derived variables classes take different views of the design, uncertain, and state variable types and the continuous and discrete domain types. The MixedVariables derived class separates the continuous and discrete domain types (see Variables::get_variables(problem_db)).

13.64.2 Constructor & Destructor Documentation

MixedVariables (const ProblemDescDB & *problem_db*, const std::pair< short, short > & *view*)

standard constructor

In this class, the distinct approach is used (design, uncertain, and state variable types and continuous and discrete domain types are distinct). Most iterators/strategies use this approach.

References Variables::allContinuousVars, Variables::allDiscreteIntVars, Variables::allDiscreteRealVars, Variables::allDiscreteStringVars, Dakota::copy_data_partial(), ProblemDescDB::get_iv(), ProblemDescDB::get_rv(), ProblemDescDB::get_sa(), Variables::sharedVarsData, and SharedVariablesData::view().

13.64.3 Member Function Documentation

void read_tabular (std::istream & *s*, bool *active_only* = `false`) `[protected], [virtual]`

Tabular reader that reads data in order design, aleatory, epistemic, state according to counts in vc_totals (extract in order: cdv/ddiv/ddrv, cauv/dauiv/daurv, ceuv/deuiv/deurv, csv/dsiv/dsrv, which might reflect active or all depending on context. Assumes container sized, since might be a view into a larger array.

Reimplemented from Variables.

References SharedVariablesData::active_components_totals(), Variables::allContinuousVars, Variables::allDiscreteIntVars, Variables::allDiscreteRealVars, Variables::allDiscreteStringVars, SharedVariablesData::components_totals(), and Variables::sharedVarsData.

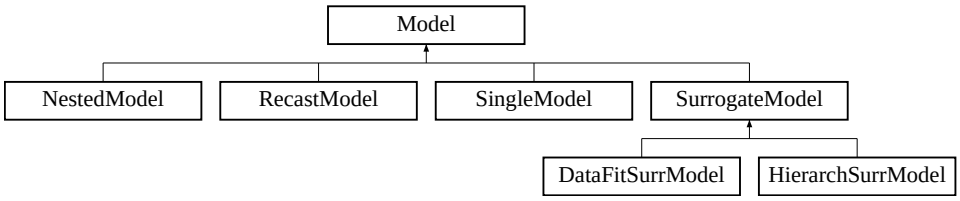
The documentation for this class was generated from the following files:

- MixedVariables.hpp
- MixedVariables.cpp

13.65 Model Class Reference

Base class for the model class hierarchy.

Inheritance diagram for Model:



Public Member Functions

- Model ()

  default constructor
- Model (ProblemDescDB &problem_db)

  standard constructor for envelope
- Model (const Model &model)

  copy constructor
- virtual ~Model ()

  destructor
- Model operator= (const Model &model)

  assignment operator
- virtual Iterator & subordinate_iterator ()

  return the sub-iterator in nested and surrogate models
- virtual Model & subordinate_model ()

  return a single sub-model defined from subModel in nested and recast models and truth_model() in surrogate models; used for a directed dive through model recursions that may bypass some components.
- virtual Model & surrogate_model ()

  return the approximation sub-model in surrogate models
- virtual Model & truth_model ()

  return the truth sub-model in surrogate models
- virtual void derived_subordinate_models (ModelList &ml, bool recurse_flag)

  portion of subordinate_models() specific to derived model classes
- virtual void update_from_subordinate_model (bool recurse_flag=true)

  propagate vars/labels/bounds/targets from the bottom up
- virtual Interface & derived_interface ()

  return the interface employed by the derived model class, if present: SingleModel::userDefinedInterface, DataFitSurrModel::approxInterface, or NestedModel::optionalInterface
- virtual void primary_response_fn_weights (const RealVector &wts, bool recurse_flag=true)

  set the relative weightings for multiple objective functions or least squares terms
- virtual void surrogate_function_indices (const IntSet &surr_fn_indices)

  set the (currently active) surrogate function index set
- virtual void build_approximation ()

  build a new SurrogateModel approximation
- virtual bool build_approximation (const Variables &vars, const IntResponsePair &response_pr)

  build a new SurrogateModel approximation using/enforcing response at vars
- virtual void update_approximation (bool rebuild_flag)

  replace the approximation data within an existing surrogate based on data updates propagated elsewhere
- virtual void update_approximation (const Variables &vars, const IntResponsePair &response_pr, bool rebuild_flag)

  replace the anchor point data within an existing surrogate
- virtual void update_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map, bool rebuild_flag)

  replace the data points within an existing surrogate

- virtual void append_approximation (bool rebuild_flag)

  append to the existing approximation data within a surrogate based on data updates propagated elsewhere
- virtual void append_approximation (const Variables &vars, const IntResponsePair &response_pr, bool rebuild_flag)

  append a single point to an existing surrogate's data
- virtual void append_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map, bool rebuild_flag)

  append multiple points to an existing surrogate's data
- virtual void pop_approximation (bool save_surr_data, bool rebuild_flag=false)

  remove the previous data set addition to a surrogate (e.g., due to a previous append_approximation() call); flag manages storing of surrogate data for use in a subsequent restore_approximation()
- virtual void restore_approximation ()

  restore a previous approximation data state within a surrogate
- virtual bool restore_available ()

  query for whether a trial increment is restorable within a surrogate
- virtual void finalize_approximation ()

  finalize an approximation by applying all previous trial increments
- virtual void store_approximation ()

  move the current approximation into storage for later combination
- virtual void combine_approximation (short corr_type)

  combine the current approximation with one previously stored
- virtual bool force_rebuild ()

  determine whether a surrogate model rebuild should be forced based on changes in the inactive data
- virtual SharedApproxData & shared_approximation ()

  retrieve the shared approximation data within the ApproximationInterface of a DataFitSurrModel
- virtual std::vector < Approximation > & approximations ()

  retrieve the set of Approximations within the ApproximationInterface of a DataFitSurrModel
- virtual const Pecos::SurrogateData & approximation_data (size_t index)

  retrieve the approximation data from a particular Approximation instance within the ApproximationInterface of a DataFitSurrModel
- virtual const RealVectorArray & approximation_coefficients (bool normalized=false)

  retrieve the approximation coefficients from each Approximation within a DataFitSurrModel
- virtual void approximation_coefficients (const RealVectorArray &approx_coeffs, bool normalized=false)

  set the approximation coefficients for each Approximation within a DataFitSurrModel
- virtual const RealVector & approximation_variances (const Variables &vars)

  retrieve the approximation variances from each Approximation within a DataFitSurrModel
- virtual void surrogate_response_mode (short mode)

  set response computation mode used in SurrogateModels for forming currentResponse
- virtual short surrogate_response_mode () const

  return response computation mode used in SurrogateModels for forming currentResponse
- virtual DiscrepancyCorrection & discrepancy_correction ()

  return the DiscrepancyCorrection object used by SurrogateModels

- virtual void component_parallel_mode (short mode)

  update component parallel mode for supporting parallelism in a model's interface component, sub-model component, or neither component [componentParallelMode = 0 (none), 1 (INTERFACE/APPROX_INTERFACE/OPTIONAL_INTERFACE/LF_MODEL/SURROGATE_MODEL), or 2 (SUB_MODEL/ACTUAL_MODEL/HF_MODEL/TRUTH_MODEL)].
- virtual size_t mi_parallel_level_index () const

  return the index for the metaiterator-iterator parallelism level within ParallelConfiguration::miPLIters that is active for use in a particular Model at runtime
- virtual short local_eval_synchronization ()

  return derived model synchronization setting
- virtual int local_eval_concurrency ()

  return derived model asynchronous evaluation concurrency
- virtual void serve_run (ParLevLIter pl_iter, int max_eval_concurrency)

  Service job requests received from the master. Completes when a termination message is received from stop_servers().
- virtual void stop_servers ()

  Executed by the master to terminate all server operations for a particular model when iteration on the model is complete.
- virtual bool derived_master_overload () const

  Return a flag indicating the combination of multiprocessor evaluations and a dedicated master iterator scheduling. Used in synchronous compute_response functions to prevent the error of trying to run a multiprocessor job on the master.
- virtual void inactive_view (short view, bool recurse_flag=true)

  update the Model's inactive view based on higher level (nested) context
- virtual const String & interface_id () const

  return the interface identifier
- virtual int evaluation_id () const

  Return the value of the evaluation id counter for the Model.
- virtual bool evaluation_cache () const

  Indicates the usage of an evaluation cache by the Model.
- virtual void set_evaluation_reference ()

  Set the reference points for the evaluation counters within the Model.
- virtual void fine_grained_evaluation_counters ()

  Request fine-grained evaluation reporting within the Model.
- virtual void print_evaluation_summary (std::ostream &s, bool minimal_header=false, bool relative_count=true) const

  Print an evaluation summary for the Model.
- virtual void eval_tag_prefix (const String &eval_id_str)

  set the hierarchical eval ID tag prefix
- ModelList & subordinate_models (bool recurse_flag=true)

  return the sub-models in nested and surrogate models
- void compute_response ()

  Compute the Response at currentVariables (default ActiveSet).
- void compute_response (const ActiveSet &set)

  Compute the Response at currentVariables (specified ActiveSet).

- void asynch_compute_response ()

  Spawn an asynchronous job (or jobs) that computes the value of the Response at currentVariables (default ActiveSet).
- void asynch_compute_response (const ActiveSet &set)

  Spawn an asynchronous job (or jobs) that computes the value of the Response at currentVariables (specified ActiveSet).
- const IntResponseMap & synchronize ()

  Execute a blocking scheduling algorithm to collect the complete set of results from a group of asynchronous evaluations.
- const IntResponseMap & synchronize_nowait ()

  Execute a nonblocking scheduling algorithm to collect all available results from a group of asynchronous evaluations.
- void init_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  allocate communicator partitions for a model and store configuration in modelPCIterMap
- void init_serial ()

  for cases where init_communicators() will not be called, modify some default settings to behave properly in serial.
- void set_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set active parallel configuration for the model (set modelPCIter from modelPCIterMap)
- void free_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  deallocate communicator partitions for a model
- MPI_Comm analysis_comm () const

  retrieve the MPI communicator on which this model is configured to conduct function evaluation analyses (provided for library clients)
- void stop_init (ParLevLIter pl_iter)

  called from IteratorScheduler::init_iterator() for iteratorComm rank 0 to terminate serve_init() on other iteratorComm processors
- int serve_init (ParLevLIter pl_iter)

  called from IteratorScheduler::init_iterator() for iteratorComm rank != 0 to balance init_communicators() calls on iteratorComm rank 0
- void estimate_message_lengths ()

  estimate messageLengths for a model
- void assign_rep (Model *model_rep, bool ref_count_incr=true)

  replaces existing letter with a new one
- size_t tv () const

  returns total number of vars
- size_t cv () const

  returns number of active continuous variables
- size_t div () const

  returns number of active discrete integer vars
- size_t dsv () const

  returns number of active discrete string vars
- size_t drv () const

  returns number of active discrete real vars
- size_t icv () const

returns number of inactive continuous variables

- size_t idiv () const

  returns number of inactive discrete integer vars
- size_t idsv () const

  returns number of inactive discrete string vars
- size_t idrv () const

  returns number of inactive discrete real vars
- size_t acv () const

  returns total number of continuous variables
- size_t adiv () const

  returns total number of discrete integer vars
- size_t adsv () const

  returns total number of discrete string vars
- size_t adrv () const

  returns total number of discrete real vars
- void active_variables (const Variables &vars)

  set the active variables in currentVariables
- const RealVector & continuous_variables () const

  return the active continuous variables from currentVariables
- void continuous_variables (const RealVector &c_vars)

  set the active continuous variables in currentVariables
- void continuous_variable (Real c_var, size_t i)

  set an active continuous variable in currentVariables
- const IntVector & discrete_int_variables () const

  return the active discrete integer variables from currentVariables
- void discrete_int_variables (const IntVector &d_vars)

  set the active discrete integer variables in currentVariables
- void discrete_int_variable (int d_var, size_t i)

  set an active discrete integer variable in currentVariables
- StringMultiArrayConstView discrete_string_variables () const

  return the active discrete string variables from currentVariables
- void discrete_string_variables (StringMultiArrayConstView d_vars)

  set the active discrete string variables in currentVariables
- void discrete_string_variable (const String &d_var, size_t i)

  set an active discrete string variable in currentVariables
- const RealVector & discrete_real_variables () const

  return the active discrete real variables from currentVariables
- void discrete_real_variables (const RealVector &d_vars)

  set the active discrete real variables in currentVariables
- void discrete_real_variable (Real d_var, size_t i)

  set an active discrete real variable in currentVariables
- UShortMultiArrayConstView continuous_variable_types () const

  return the active continuous variable types from currentVariables

- void continuous_variable_types (UShortMultiArrayConstView cv_types)

  set the active continuous variable types in currentVariables
- void continuous_variable_type (unsigned short cv_type, size_t i)

  set an active continuous variable type in currentVariables
- UShortMultiArrayConstView discrete_int_variable_types () const

  return the active discrete variable types from currentVariables
- void discrete_int_variable_types (UShortMultiArrayConstView div_types)

  set the active discrete variable types in currentVariables
- void discrete_int_variable_type (unsigned short div_type, size_t i)

  set an active discrete variable type in currentVariables
- UShortMultiArrayConstView discrete_string_variable_types () const

  return the active discrete variable types from currentVariables
- void discrete_string_variable_types (UShortMultiArrayConstView div_types)

  set the active discrete variable types in currentVariables
- void discrete_string_variable_type (unsigned short div_type, size_t i)

  set an active discrete variable type in currentVariables
- UShortMultiArrayConstView discrete_real_variable_types () const

  return the active discrete variable types from currentVariables
- void discrete_real_variable_types (UShortMultiArrayConstView drv_types)

  set the active discrete variable types in currentVariables
- void discrete_real_variable_type (unsigned short drv_type, size_t i)

  set an active discrete variable type in currentVariables
- SizetMultiArrayConstView continuous_variable_ids () const

  return the active continuous variable identifiers from currentVariables
- void continuous_variable_ids (SizetMultiArrayConstView cv_ids)

  set the active continuous variable identifiers in currentVariables
- void continuous_variable_id (size_t cv_id, size_t i)

  set an active continuous variable identifier in currentVariables
- const RealVector & inactive_continuous_variables () const

  return the inactive continuous variables in currentVariables
- void inactive_continuous_variables (const RealVector &i_c_vars)

  set the inactive continuous variables in currentVariables
- const IntVector & inactive_discrete_int_variables () const

  return the inactive discrete variables in currentVariables
- void inactive_discrete_int_variables (const IntVector &i_d_vars)

  set the inactive discrete variables in currentVariables
- StringMultiArrayConstView inactive_discrete_string_variables () const

  return the inactive discrete variables in currentVariables
- void inactive_discrete_string_variables (StringMultiArrayConstView i_d_vars)

  set the inactive discrete variables in currentVariables
- const RealVector & inactive_discrete_real_variables () const

  return the inactive discrete variables in currentVariables
- void inactive_discrete_real_variables (const RealVector &i_d_vars)

set the inactive discrete variables in currentVariables

- UShortMultiArrayConstView inactive_continuous_variable_types () const

  return the inactive continuous variable types from currentVariables
- SizetMultiArrayConstView inactive_continuous_variable_ids () const

  return the inactive continuous variable identifiers from currentVariables
- const RealVector & all_continuous_variables () const

  return all continuous variables in currentVariables
- void all_continuous_variables (const RealVector &a_c_vars)

  set all continuous variables in currentVariables
- void all_continuous_variable (Real a_c_var, size_t i)

  set a variable within the all continuous variables in currentVariables
- const IntVector & all_discrete_int_variables () const

  return all discrete variables in currentVariables
- void all_discrete_int_variables (const IntVector &a_d_vars)

  set all discrete variables in currentVariables
- void all_discrete_int_variable (int a_d_var, size_t i)

  set a variable within the all discrete variables in currentVariables
- StringMultiArrayConstView all_discrete_string_variables () const

  return all discrete variables in currentVariables
- void all_discrete_string_variables (StringMultiArrayConstView a_d_vars)

  set all discrete variables in currentVariables
- void all_discrete_string_variable (const String &a_d_var, size_t i)

  set a variable within the all discrete variables in currentVariables
- const RealVector & all_discrete_real_variables () const

  return all discrete variables in currentVariables
- void all_discrete_real_variables (const RealVector &a_d_vars)

  set all discrete variables in currentVariables
- void all_discrete_real_variable (Real a_d_var, size_t i)

  set a variable within the all discrete variables in currentVariables
- UShortMultiArrayConstView all_continuous_variable_types () const

  return all continuous variable types from currentVariables
- UShortMultiArrayConstView all_discrete_int_variable_types () const

  return all discrete variable types from currentVariables
- UShortMultiArrayConstView all_discrete_string_variable_types () const

  return all discrete variable types from currentVariables
- UShortMultiArrayConstView all_discrete_real_variable_types () const

  return all discrete variable types from currentVariables
- SizetMultiArrayConstView all_continuous_variable_ids () const

  return all continuous variable identifiers from currentVariables
- const IntSetArray & discrete_design_set_int_values () const

  return the sets of values available for each of the discrete design set integer variables
- void discrete_design_set_int_values (const IntSetArray &isa)

  define the sets of values available for each of the discrete design set integer variables

- const StringSetArray & discrete_design_set_string_values () const

  return the sets of values available for each of the discrete design set string variables
- void discrete_design_set_string_values (const StringSetArray &ssa)

  define the sets of values available for each of the discrete design set string variables
- const RealSetArray & discrete_design_set_real_values () const

  return the sets of values available for each of the discrete design set real variables
- void discrete_design_set_real_values (const RealSetArray &rsa)

  define the sets of values available for each of the discrete design set real variables
- const IntSetArray & discrete_state_set_int_values () const

  return the sets of values available for each of the discrete state set integer variables
- void discrete_state_set_int_values (const IntSetArray &isa)

  define the sets of values available for each of the discrete state set integer variables
- const StringSetArray & discrete_state_set_string_values () const

  return the sets of values available for each of the discrete state set string variables
- void discrete_state_set_string_values (const StringSetArray &ssa)

  define the sets of values available for each of the discrete state set string variables
- const RealSetArray & discrete_state_set_real_values () const

  return the sets of values available for each of the discrete state set real variables
- void discrete_state_set_real_values (const RealSetArray &rsa)

  define the sets of values available for each of the discrete state set real variables
- const BitArray & discrete_int_sets ()

  define and return discreteIntSets using active view from currentVariables
- const BitArray & discrete_int_sets (short active_view)

  define and return discreteIntSets using passed active view
- const IntSetArray & discrete_set_int_values ()

  return the sets of values available for each of the active discrete set integer variables (aggregated in activeDiscSetIntValues)
- const IntSetArray & discrete_set_int_values (short active_view)

  return the sets of values available for each of the active discrete set integer variables (aggregated in activeDiscSetIntValues)
- const StringSetArray & discrete_set_string_values ()

  return the sets of values available for each of the active discrete set string variables (aggregated in activeDiscSetStringValues)
- const StringSetArray & discrete_set_string_values (short active_view)

  return the sets of values available for each of the active discrete set string variables (aggregated in activeDiscSetStringValues)
- const RealSetArray & discrete_set_real_values ()

  return the sets of values available for each of the active discrete set real variables (aggregated in activeDiscSetRealValues)
- const RealSetArray & discrete_set_real_values (short active_view)

  return the sets of values available for each of the active discrete set real variables (aggregated in activeDiscSetRealValues)
- Pecos::AleatoryDistParams & aleatory_distribution_parameters ()

  return aleatDistParams

- const Pecos::AleatoryDistParams & aleatory_distribution_parameters () const

  return aleatDistParams
- void aleatory_distribution_parameters (const Pecos::AleatoryDistParams &adp)

  set aleatDistParams
- Pecos::EpistemicDistParams & epistemic_distribution_parameters ()

  return epistDistParams
- const Pecos::EpistemicDistParams & epistemic_distribution_parameters () const

  return epistDistParams
- void epistemic_distribution_parameters (const Pecos::EpistemicDistParams &edp)

  set epistDistParams
- Real continuous_probability_density () const

  compute a multivariate probability density from the marginals for the active aleatory random variables (must be uncorrelated)
- std::pair< Real, Real > continuous_distribution_bounds (size_t cv_index) const

  return a set of distribution bounds for a particular index within the active aleatory random variables
- StringMultiArrayConstView continuous_variable_labels () const

  return the active continuous variable labels from currentVariables
- void continuous_variable_labels (StringMultiArrayConstView c_v_labels)

  set the active continuous variable labels in currentVariables
- StringMultiArrayConstView discrete_int_variable_labels () const

  return the active discrete variable labels from currentVariables
- void discrete_int_variable_labels (StringMultiArrayConstView d_v_labels)

  set the active discrete variable labels in currentVariables
- StringMultiArrayConstView discrete_string_variable_labels () const

  return the active discrete variable labels from currentVariables
- void discrete_string_variable_labels (StringMultiArrayConstView d_v_labels)

  set the active discrete variable labels in currentVariables
- StringMultiArrayConstView discrete_real_variable_labels () const

  return the active discrete variable labels from currentVariables
- void discrete_real_variable_labels (StringMultiArrayConstView d_v_labels)

  set the active discrete variable labels in currentVariables
- StringMultiArrayConstView inactive_continuous_variable_labels () const

  return the inactive continuous variable labels in currentVariables
- void inactive_continuous_variable_labels (StringMultiArrayConstView i_c_v_labels)

  set the inactive continuous variable labels in currentVariables
- StringMultiArrayConstView inactive_discrete_int_variable_labels () const

  return the inactive discrete variable labels in currentVariables
- void inactive_discrete_int_variable_labels (StringMultiArrayConstView i_d_v_labels)

  set the inactive discrete variable labels in currentVariables
- StringMultiArrayConstView inactive_discrete_string_variable_labels () const

  return the inactive discrete variable labels in currentVariables
- void inactive_discrete_string_variable_labels (StringMultiArrayConstView i_d_v_labels)

  set the inactive discrete variable labels in currentVariables

- StringMultiArrayConstView inactive_discrete_real_variable_labels () const

  return the inactive discrete variable labels in currentVariables
- void inactive_discrete_real_variable_labels (StringMultiArrayConstView i_d_v_labels)

  set the inactive discrete variable labels in currentVariables
- StringMultiArrayConstView all_continuous_variable_labels () const

  return all continuous variable labels in currentVariables
- void all_continuous_variable_labels (StringMultiArrayConstView a_c_v_labels)

  set all continuous variable labels in currentVariables
- void all_continuous_variable_label (const String &a_c_v_label, size_t i)

  set a label within the all continuous labels in currentVariables
- StringMultiArrayConstView all_discrete_int_variable_labels () const

  return all discrete variable labels in currentVariables
- void all_discrete_int_variable_labels (StringMultiArrayConstView a_d_v_labels)

  set all discrete variable labels in currentVariables
- void all_discrete_int_variable_label (const String &a_d_v_label, size_t i)

  set a label within the all discrete labels in currentVariables
- StringMultiArrayConstView all_discrete_string_variable_labels () const

  return all discrete variable labels in currentVariables
- void all_discrete_string_variable_labels (StringMultiArrayConstView a_d_v_labels)

  set all discrete variable labels in currentVariables
- void all_discrete_string_variable_label (const String &a_d_v_label, size_t i)

  set a label within the all discrete labels in currentVariables
- StringMultiArrayConstView all_discrete_real_variable_labels () const

  return all discrete variable labels in currentVariables
- void all_discrete_real_variable_labels (StringMultiArrayConstView a_d_v_labels)

  set all discrete variable labels in currentVariables
- void all_discrete_real_variable_label (const String &a_d_v_label, size_t i)

  set a label within the all discrete labels in currentVariables
- const StringArray & response_labels () const

  return the response labels from currentResponse
- void response_labels (const StringArray &resp_labels)

  set the response labels in currentResponse
- const RealVector & continuous_lower_bounds () const

  return the active continuous lower bounds from userDefinedConstraints
- void continuous_lower_bounds (const RealVector &c_l_bnds)

  set the active continuous lower bounds in userDefinedConstraints
- void continuous_lower_bound (Real c_l_bnd, size_t i)

  set the i-th active continuous lower bound in userDefinedConstraints
- const RealVector & continuous_upper_bounds () const

  return the active continuous upper bounds from userDefinedConstraints
- void continuous_upper_bounds (const RealVector &c_u_bnds)

  set the active continuous upper bounds in userDefinedConstraints
- void continuous_upper_bound (Real c_u_bnd, size_t i)

set the i-th active continuous upper bound from userDefinedConstraints

- const IntVector & discrete_int_lower_bounds () const

  return the active discrete int lower bounds from userDefinedConstraints
- void discrete_int_lower_bounds (const IntVector &d_l_bnds)

  set the active discrete int lower bounds in userDefinedConstraints
- void discrete_int_lower_bound (int d_l_bnd, size_t i)

  set the i-th active discrete int lower bound in userDefinedConstraints
- const IntVector & discrete_int_upper_bounds () const

  return the active discrete int upper bounds from userDefinedConstraints
- void discrete_int_upper_bounds (const IntVector &d_u_bnds)

  set the active discrete int upper bounds in userDefinedConstraints
- void discrete_int_upper_bound (int d_u_bnd, size_t i)

  set the i-th active discrete int upper bound in userDefinedConstraints
- const RealVector & discrete_real_lower_bounds () const

  return the active discrete real lower bounds from userDefinedConstraints
- void discrete_real_lower_bounds (const RealVector &d_l_bnds)

  set the active discrete real lower bounds in userDefinedConstraints
- void discrete_real_lower_bound (Real d_l_bnd, size_t i)

  set the i-th active discrete real lower bound in userDefinedConstraints
- const RealVector & discrete_real_upper_bounds () const

  return the active discrete real upper bounds from userDefinedConstraints
- void discrete_real_upper_bounds (const RealVector &d_u_bnds)

  set the active discrete real upper bounds in userDefinedConstraints
- void discrete_real_upper_bound (Real d_u_bnd, size_t i)

  set the i-th active discrete real upper bound in userDefinedConstraints
- const RealVector & inactive_continuous_lower_bounds () const

  return the inactive continuous lower bounds in userDefinedConstraints
- void inactive_continuous_lower_bounds (const RealVector &i_c_l_bnds)

  set the inactive continuous lower bounds in userDefinedConstraints
- const RealVector & inactive_continuous_upper_bounds () const

  return the inactive continuous upper bounds in userDefinedConstraints
- void inactive_continuous_upper_bounds (const RealVector &i_c_u_bnds)

  set the inactive continuous upper bounds in userDefinedConstraints
- const IntVector & inactive_discrete_int_lower_bounds () const

  return the inactive discrete lower bounds in userDefinedConstraints
- void inactive_discrete_int_lower_bounds (const IntVector &i_d_l_bnds)

  set the inactive discrete lower bounds in userDefinedConstraints
- const IntVector & inactive_discrete_int_upper_bounds () const

  return the inactive discrete upper bounds in userDefinedConstraints
- void inactive_discrete_int_upper_bounds (const IntVector &i_d_u_bnds)

  set the inactive discrete upper bounds in userDefinedConstraints
- const RealVector & inactive_discrete_real_lower_bounds () const

  return the inactive discrete lower bounds in userDefinedConstraints

- void inactive_discrete_real_lower_bounds (const RealVector &i_d_l_bnds)

  set the inactive discrete lower bounds in userDefinedConstraints
- const RealVector & inactive_discrete_real_upper_bounds () const

  return the inactive discrete upper bounds in userDefinedConstraints
- void inactive_discrete_real_upper_bounds (const RealVector &i_d_u_bnds)

  set the inactive discrete upper bounds in userDefinedConstraints
- const RealVector & all_continuous_lower_bounds () const

  return all continuous lower bounds in userDefinedConstraints
- void all_continuous_lower_bounds (const RealVector &a_c_l_bnds)

  set all continuous lower bounds in userDefinedConstraints
- void all_continuous_lower_bound (Real a_c_l_bnd, size_t i)

  set a lower bound within continuous lower bounds in userDefinedConstraints
- const RealVector & all_continuous_upper_bounds () const

  return all continuous upper bounds in userDefinedConstraints
- void all_continuous_upper_bounds (const RealVector &a_c_u_bnds)

  set all continuous upper bounds in userDefinedConstraints
- void all_continuous_upper_bound (Real a_c_u_bnd, size_t i)

  set an upper bound within all continuous upper bounds in userDefinedConstraints
- const IntVector & all_discrete_int_lower_bounds () const

  return all discrete lower bounds in userDefinedConstraints
- void all_discrete_int_lower_bounds (const IntVector &a_d_l_bnds)

  set all discrete lower bounds in userDefinedConstraints
- void all_discrete_int_lower_bound (int a_d_l_bnd, size_t i)

  set a lower bound within all discrete lower bounds in userDefinedConstraints
- const IntVector & all_discrete_int_upper_bounds () const

  return all discrete upper bounds in userDefinedConstraints
- void all_discrete_int_upper_bounds (const IntVector &a_d_u_bnds)

  set all discrete upper bounds in userDefinedConstraints
- void all_discrete_int_upper_bound (int a_d_u_bnd, size_t i)

  set an upper bound within all discrete upper bounds in userDefinedConstraints
- const RealVector & all_discrete_real_lower_bounds () const

  return all discrete lower bounds in userDefinedConstraints
- void all_discrete_real_lower_bounds (const RealVector &a_d_l_bnds)

  set all discrete lower bounds in userDefinedConstraints
- void all_discrete_real_lower_bound (Real a_d_l_bnd, size_t i)

  set a lower bound within all discrete lower bounds in userDefinedConstraints
- const RealVector & all_discrete_real_upper_bounds () const

  return all discrete upper bounds in userDefinedConstraints
- void all_discrete_real_upper_bounds (const RealVector &a_d_u_bnds)

  set all discrete upper bounds in userDefinedConstraints
- void all_discrete_real_upper_bound (Real a_d_u_bnd, size_t i)

  set an upper bound within all discrete upper bounds in userDefinedConstraints
- size_t num_linear_ineq_constraints () const

return the number of linear inequality constraints

- size_t num_linear_eq_constraints () const

  return the number of linear equality constraints
- const RealMatrix & linear_ineq_constraint_coeffs () const

  return the linear inequality constraint coefficients
- void linear_ineq_constraint_coeffs (const RealMatrix &lin_ineq_coeffs)

  set the linear inequality constraint coefficients
- const RealVector & linear_ineq_constraint_lower_bounds () const

  return the linear inequality constraint lower bounds
- void linear_ineq_constraint_lower_bounds (const RealVector &lin_ineq_l_bnds)

  set the linear inequality constraint lower bounds
- const RealVector & linear_ineq_constraint_upper_bounds () const

  return the linear inequality constraint upper bounds
- void linear_ineq_constraint_upper_bounds (const RealVector &lin_ineq_u_bnds)

  set the linear inequality constraint upper bounds
- const RealMatrix & linear_eq_constraint_coeffs () const

  return the linear equality constraint coefficients
- void linear_eq_constraint_coeffs (const RealMatrix &lin_eq_coeffs)

  set the linear equality constraint coefficients
- const RealVector & linear_eq_constraint_targets () const

  return the linear equality constraint targets
- void linear_eq_constraint_targets (const RealVector &lin_eq_targets)

  set the linear equality constraint targets
- size_t num_nonlinear_ineq_constraints () const

  return the number of nonlinear inequality constraints
- size_t num_nonlinear_eq_constraints () const

  return the number of nonlinear equality constraints
- const RealVector & nonlinear_ineq_constraint_lower_bounds () const

  return the nonlinear inequality constraint lower bounds
- void nonlinear_ineq_constraint_lower_bounds (const RealVector &nln_ineq_l_bnds)

  set the nonlinear inequality constraint lower bounds
- const RealVector & nonlinear_ineq_constraint_upper_bounds () const

  return the nonlinear inequality constraint upper bounds
- void nonlinear_ineq_constraint_upper_bounds (const RealVector &nln_ineq_u_bnds)

  set the nonlinear inequality constraint upper bounds
- const RealVector & nonlinear_eq_constraint_targets () const

  return the nonlinear equality constraint targets
- void nonlinear_eq_constraint_targets (const RealVector &nln_eq_targets)

  set the nonlinear equality constraint targets
- const Variables & current_variables () const

  return the current variables (currentVariables) as const reference (preferred)
- Variables & current_variables ()

  return the current variables (currentVariables) in mutable form (special cases)

- const Constraints & user_defined_constraints () const

  return the user-defined constraints (userDefinedConstraints)
- const Response & current_response () const

  return the current response (currentResponse)
- ProblemDescDB & problem_description_db () const

  return the problem description database (probDescDB)
- ParallelLibrary & parallel_library () const

  return the parallel library (parallelLib)
- const String & model_type () const

  return the model type (modelType)
- const String & surrogate_type () const

  return the surrogate type (surrogateType)
- const String & model_id () const

  return the model identifier (modelId)
- size_t num_functions () const

  return number of functions in currentResponse
- const String & gradient_type () const

  return the gradient evaluation type (gradientType)
- const String & method_source () const

  return the numerical gradient evaluation method source (methodSource)
- const String & interval_type () const

  return the numerical gradient evaluation interval type (intervalType)
- bool ignore_bounds () const

  option for ignoring bounds when numerically estimating derivatives
- bool central_hess () const

  option for using old 2nd-order scheme when computing finite-diff Hessian
- const RealVector & fd_gradient_step_size () const

  return the finite difference gradient step size (fdGradStepSize)
- const String & fd_gradient_step_type () const

  return the finite difference gradient step type (fdGradStepType)
- const IntSet & gradient_id_analytic () const

  return the mixed gradient analytic IDs (gradIdAnalytic)
- const IntSet & gradient_id_numerical () const

  return the mixed gradient numerical IDs (gradIdNumerical)
- const String & hessian_type () const

  return the Hessian evaluation type (hessianType)
- const String & quasi_hessian_type () const

  return the Hessian evaluation type (quasiHessType)
- const RealVector & fd_hessian_by_grad_step_size () const

  return gradient-based finite difference Hessian step size (fdHessByGradStepSize)
- const RealVector & fd_hessian_by_fn_step_size () const

  return function-based finite difference Hessian step size (fdHessByFnStepSize)
- const String & fd_hessian_step_type () const

return the finite difference Hessian step type (fdHessStepType)

- const IntSet & hessian_id_analytic () const

  return the mixed Hessian analytic IDs (hessIdAnalytic)
- const IntSet & hessian_id_numerical () const

  return the mixed Hessian analytic IDs (hessIdNumerical)
- const IntSet & hessian_id_quasi () const

  return the mixed Hessian analytic IDs (hessIdQuasi)
- void primary_response_fn_sense (const BoolDeque &sense)

  set the optimization sense for multiple objective functions
- const BoolDeque & primary_response_fn_sense () const

  get the optimization sense for multiple objective functions
- const RealVector & primary_response_fn_weights () const

  get the relative weightings for multiple objective functions or least squares terms
- bool derivative_estimation ()

  indicates potential usage of estimate_derivatives() based on gradientType/hessianType
- void supports_derivative_estimation (bool sed_flag)

  set whether this model should perform or pass on derivative estimation
- void init_comms_bcast_flag (bool icb_flag)

  set initCommsBcastFlag
- int evaluation_capacity () const

  return the evaluation capacity for use in iterator logic
- int derivative_concurrency () const

  return the gradient concurrency for use in parallel configuration logic
- bool asynch_flag () const

  return the asynchronous evaluation flag (asynchEvalFlag)
- void asynch_flag (const bool flag)

  set the asynchronous evaluation flag (asynchEvalFlag)
- short output_level () const

  return the outputLevel
- void output_level (const short level)

  set the outputLevel
- const IntArray & message_lengths () const

  return the array of MPI packed message buffer lengths (messageLengths)
- void parallel_configuration_iterator (ParConfigLIter pc_iter)

  set modelPCIter
- ParConfigLIter parallel_configuration_iterator () const

  return modelPCIter
- void auto_graphics (const bool flag)

  set modelAutoGraphicsFlag to activate posting of graphics data within compute_response/synchronize functions (automatic graphics posting in the model as opposed to graphics posting at the strategy level).
- bool is_null () const

  function to check modelRep (does this envelope contain a letter)
- Model * model_rep () const

returns modelRep for access to derived class member functions that are not mapped to the top Model level

- Real initialize_h (Real x_j, Real lb_j, Real ub_j, Real step_size, String step_type)

  function to determine initial finite difference h (before step length adjustment) based on type of step desired

- Real FDstep1 (Real x0_j, Real lb_j, Real ub_j, Real h_mag)

  function returning finite-difference step size (affected by bounds)

- Real FDstep2 (Real x0_j, Real lb_j, Real ub_j, Real h)

  function returning second central-difference step size (affected by bounds)

Public Attributes

- bool shortStep

  flags finite-difference step size adjusted by bounds

Protected Member Functions

- Model (BaseConstructor, ProblemDescDB &problem_db)

  constructor initializing the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

- Model (LightWtBaseConstructor, ProblemDescDB &problem_db, ParallelLibrary &parallel_lib, const SharedVariablesData &svd, const ActiveSet &set, short output_level)

  constructor initializing base class for derived model class instances constructed on the fly

- Model (LightWtBaseConstructor, ProblemDescDB &problem_db, ParallelLibrary &parallel_lib)

  constructor initializing base class for recast model instances

- virtual void derived_compute_response (const ActiveSet &set)

  portion of compute_response() specific to derived model classes

- virtual void derived_asynch_compute_response (const ActiveSet &set)

  portion of asynch_compute_response() specific to derived model classes

- virtual const IntResponseMap & derived_synchronize ()

  portion of synchronize() specific to derived model classes

- virtual const IntResponseMap & derived_synchronize_nowait ()

  portion of synchronize_nowait() specific to derived model classes

- virtual void derived_init_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  portion of init_communicators() specific to derived model classes

- virtual void derived_init_serial ()

  portion of init_serial() specific to derived model classes

- virtual void derived_set_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  portion of set_communicators() specific to derived model classes

- virtual void derived_free_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  portion of free_communicators() specific to derived model classes

- void set_ie_asynchronous_mode (int max_eval_concurrency)

  default logic for defining asynchEvalFlag and evaluationCapacity based on ie_pl settings

Protected Attributes

- Variables currentVariables

  the set of current variables used by the model for performing function evaluations
- size_t numDerivVars

  the number of active continuous variables used in computing most response derivatives (i.e., in places such as quasi-Hessians and response corrections where only the active continuous variables are supported)
- Response currentResponse

  the set of current responses that holds the results of model function evaluations
- size_t numFns

  the number of functions in currentResponse
- Constraints userDefinedConstraints

  Explicit constraints on variables are maintained in the Constraints class hierarchy. Currently, this includes linear constraints and bounds, but could be extended in the future to include other explicit constraints which (1) have their form specified by the user, and (2) are not catalogued in Response since their form and coefficients are published to an iterator at startup.
- String modelId

  model identifier string from the input file
- String modelType

  type of model: single, nested, or surrogate
- String surrogateType

  type of surrogate model: local_*, multipoint_*, global_*, or hierarchical
- String gradientType

  type of gradient data: analytic, numerical, mixed, or none
- String methodSource

  source of numerical gradient routine: dakota or vendor
- String intervalType

  type of numerical gradient interval: central or forward
- String hessianType

  type of Hessian data: analytic, numerical, quasi, mixed, or none
- RealVector fdGradStepSize

  relative finite difference step size for numerical gradients
- String fdGradStepType

  type of finite difference step to use for numerical gradient: relative - step length is relative to x absolute - step length is what is specified bounds - step length is relative to range of x
- RealVector fdHessByGradStepSize

  relative finite difference step size for numerical Hessians estimated using first-order differences of gradients
- RealVector fdHessByFnStepSize

  relative finite difference step size for numerical Hessians estimated using second-order differences of function values
- String fdHessStepType

  type of finite difference step to use for numerical Hessian: relative - step length is relative to x absolute - step length is what is specified bounds - step length is relative to range of x
- bool ignoreBounds

  option to ignore bounds when computing finite diffs
- bool centralHess

option to use old 2nd-order finite diffs for Hessians

- bool supportsEstimDerivs

  whether model should perform or forward derivative estimation

- String quasiHessType

  quasi-Hessian type: bfgs, damped_bfgs, sr1

- IntSet gradIdAnalytic

  analytic id's for mixed gradients

- IntSet gradIdNumerical

  numerical id's for mixed gradients

- IntSet hessIdAnalytic

  analytic id's for mixed Hessians

- IntSet hessIdNumerical

  numerical id's for mixed Hessians

- IntSet hessIdQuasi

  quasi id's for mixed Hessians

- IntArray messageLengths

  length of packed MPI buffers containing vars, vars/set, response, and PRPair

- ProblemDescDB & probDescDB

  class member reference to the problem description database

- ParallelLibrary & parallelLib

  class member reference to the parallel library

- ParConfigLIter modelPCIter

  the ParallelConfiguration node used by this Model instance

- short componentParallelMode

  the component parallelism mode: 0 (none), 1 (INTERFACE/LF_MODEL), or 2 (SUB_MODEL/HF_MODEL/TRUTH_MODEL)

- bool asynchEvalFlag

  flags asynch evaluations (local or distributed)

- int evaluationCapacity

  capacity for concurrent evaluations supported by the Model

- short outputLevel

  output verbosity level: {SILENT,QUIET,NORMAL,VERBOSE,DEBUG}_OUTPUT

- ModelList modelList

  used to collect sub-models for subordinate_models()

- IntSetArray discreteDesignSetIntValues

  array of IntSet's, each containing the set of allowable integer values corresponding to a discrete design integer set variable

- StringSetArray discreteDesignSetStringValues

  array of IntSet's, each containing the set of allowable integer values corresponding to a discrete design string set variable

- RealSetArray discreteDesignSetRealValues

  array of RealSet's, each containing the set of allowable real values corresponding to a discrete design real set variable

- IntSetArray discreteStateSetIntValues

array of IntSet's, each containing the set of allowable integer values corresponding to a discrete state integer set variable

- StringSetArray discreteStateSetStringValues

  array of IntSet's, each containing the set of allowable integer values corresponding to a discrete state string set variable

- RealSetArray discreteStateSetRealValues

  array of RealSet's, each containing the set of allowable real values corresponding to a discrete state real set variable

- Pecos::AleatoryDistParams aleatDistParams

  container for aleatory random variable distribution parameters

- Pecos::EpistemicDistParams epistDistParams

  container for epistemic random variable distribution parameters

- BoolDeque primaryRespFnSense

  array of flags (one per primary function) for switching the sense to maximize the primary function (default is minimize)

- RealVector primaryRespFnWts

  primary response function weightings (either weights for multiobjective optimization or weighted least squares)

- bool hierarchicalTagging

  whether to perform hierarchical evalID tagging of params/results

Private Member Functions

- Model * get_model (ProblemDescDB &problem_db)

  Used by the envelope to instantiate the correct letter class.

- int estimate_derivatives (const ShortArray &map_asv, const ShortArray &fd_grad_asv, const ShortArray &fd_hess_asv, const ShortArray &quasi_hess_asv, const ActiveSet &original_set, const bool asynch_flag)

  evaluate numerical gradients using finite differences. This routine is selected with "method_source dakota" (the default method_source) in the numerical gradient specification.

- void synchronize_derivatives (const Variables &vars, const IntResponseMap &fd_responses, Response &new_response, const ShortArray &fd_grad_asv, const ShortArray &fd_hess_asv, const ShortArray &quasi_hess_asv, const ActiveSet &original_set)

  combine results from an array of finite difference response objects (fd_grad_responses) into a single response (new_response)

- void update_response (const Variables &vars, Response &new_response, const ShortArray &fd_grad_asv, const ShortArray &fd_hess_asv, const ShortArray &quasi_hess_asv, const ActiveSet &original_set, Response &initial_map_response, const RealMatrix &new_fn_grads, const RealSymMatrixArray &new_fn_hessians)

  overlay results to update a response object

- void update_quasi_hessians (const Variables &vars, Response &new_response, const ActiveSet &original_set)

  perform quasi-Newton Hessian updates

- Real finite_difference_lower_bound (UShortMultiArrayConstView cv_types, const RealVector &global_c_l_bnds, size_t cv_index)

  return the lower bound for a finite difference offset, drawn from global or distribution bounds

- Real finite_difference_upper_bound (UShortMultiArrayConstView cv_types, const RealVector &global_c_u_bnds, size_t cv_index)

return the upper bound for a finite difference offset, drawn from global or distribution bounds

- bool manage_asv (const ShortArray &asv_in, ShortArray &map_asv_out, ShortArray &fd_grad_asv_out, ShortArray &fd_hess_asv_out, ShortArray &quasi_hess_asv_out)

  Coordinates usage of estimate_derivatives() calls based on asv_in.

Private Attributes

- int modelEvalCntr

  evaluation counter for top-level compute_response() and asynch_compute_response() calls. Differs from lower level counters in case of numerical derivative estimation (several lower level evaluations are assimilated into a single higher level evaluation)

- bool estDerivsFlag

  flags presence of estimated derivatives within a set of calls to asynch_compute_response()

- std::map< SizetIntPair, ParConfigLIter > modelPCIterMap

  map<> used for tracking modelPCIter instances using depth of parallelism level and max evaluation concurrency as the lookup keys

- bool initCommsBcastFlag

  flag for determining need to bcast the max concurrency from init_communicators(); set from IteratorScheduler::init_iterator()

- bool modelAutoGraphicsFlag

  flag for posting of graphics data within compute_response (automatic graphics posting in the model as opposed to graphics posting at the strategy level)

- VariablesList varsList

  history of vars populated in asynch_compute_response() and used in synchronize().

- std::list< ShortArray > asvList

  if estimate_derivatives() is used, transfers ASVs from asynch_compute_response() to synchronize()

- std::list< ActiveSet > setList

  if estimate_derivatives() is used, transfers ActiveSets from asynch_compute_response() to synchronize()

- BoolList initialMapList

  transfers initial_map flag values from estimate_derivatives() to synchronize_derivatives()

- BoolList dbCaptureList

  transfers db_capture flag values from estimate_derivatives() to synchronize_derivatives()

- ResponseList dbResponseList

  transfers database captures from estimate_derivatives() to synchronize_derivatives()

- RealList deltaList

  transfers deltas from estimate_derivatives() to synchronize_derivatives()

- IntIntMap numFDEvalsMap

  tracks the number of evaluations used within estimate_derivatives(). Used in synchronize() as a key for combining finite difference responses into numerical gradients.

- IntIntMap rawEvalIdMap

  maps from the raw evaluation ids returned by derived_synchronize() and derived_synchronize_nowait() to the corresponding modelEvalCntr id. Used for rekeying responseMap.

- RealVectorArray xPrev

  previous parameter vectors used in computing s for quasi-Newton updates

- RealMatrix fnGradsPrev

  previous gradient vectors used in computing y for quasi-Newton updates

- RealSymMatrixArray quasiHessians

  quasi-Newton Hessian approximations

- SizetArray numQuasiUpdates

  number of quasi-Newton Hessian updates applied

- IntResponseMap responseMap

  used to return a map of responses for asynchronous evaluations in final concatenated form. The similar map in Interface contains raw responses.

- IntResponseMap graphicsRespMap

  used to cache the data returned from derived_synchronize_nowait() prior to sequential input into the graphics

- IntSetArray activeDiscSetIntValues

  aggregation of the admissible value sets for all active discrete set integer variables

- StringSetArray activeDiscSetStringValues

  aggregation of the admissible value sets for all active discrete set string variables

- RealSetArray activeDiscSetRealValues

  aggregation of the admissible value sets for all active discrete set real variables

- BitArray discreteIntSets

  key for identifying discrete integer set variables within the active discrete integer variables

- Model * modelRep

  pointer to the letter (initialized only for the envelope)

- int referenceCount

  number of objects sharing modelRep

Friends

- bool operator== (const Model &m1, const Model &m2)

  equality operator (detect same letter instance)

- bool operator!= (const Model &m1, const Model &m2)

  inequality operator (detect different letter instances)

13.65.1 Detailed Description

Base class for the model class hierarchy.

The Model class is the base class for one of the primary class hierarchies in DAKOTA. The model hierarchy contains a set of variables, an interface, and a set of responses, and an iterator operates on the model to map the variables into responses using the interface. For memory efficiency and enhanced polymorphism, the model hierarchy employs the "letter/envelope idiom" (see Coplien "Advanced C++", p. 133), for which the base class (Model) serves as the envelope and one of the derived classes (selected in Model::get_model()) serves as the letter.

13.65.2 Constructor & Destructor Documentation

Model ( )

default constructor

The default constructor is used in vector<Model> instantiations and for initialization of Model objects contained in Iterator and derived Strategy classes. modelRep is NULL in this case (a populated problem_db is needed to build a meaningful Model object). This makes it necessary to check for NULL in the copy constructor, assignment operator, and destructor.

Model (ProblemDescDB & *problem_db*)

standard constructor for envelope

Used in model instantiations within strategy constructors. Envelope constructor only needs to extract enough data to properly execute get_model, since Model(BaseConstructor, problem_db) builds the actual base class data for the derived models.

References Dakota::abort_handler(), Model::get_model(), and Model::modelRep.

Model (const Model & *model*)

copy constructor

Copy constructor manages sharing of modelRep and incrementing of referenceCount.

References Model::modelRep, and Model::referenceCount.

~Model ( ) `[virtual]`

destructor

Destructor decrements referenceCount and only deletes modelRep when referenceCount reaches zero.

References Model::modelRep, and Model::referenceCount.

Model (BaseConstructor , ProblemDescDB & *problem_db*) `[protected]`

constructor initializing the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor builds the base class data for all inherited models. get_model() instantiates a derived class and the derived class selects this base class constructor in its initialization list (to avoid the recursion of the base class constructor calling get_model() again). Since the letter IS the representation, its representation pointer is set to NULL (an uninitialized pointer causes problems in ~Model).

References Dakota::abort_handler(), Model::currentResponse, Model::fdGradStepSize, Model::fdHessByFnStepSize, Model::fdHessByGradStepSize, ProblemDescDB::get_sa(), Model::gradIdNumerical, Model::gradientType, Model::hessianType, Model::hessIdNumerical, Response::num_functions(), Constraints::num_nonlinear_eq_constraints(), Constraints::num_nonlinear_ineq_constraints(), Model::primaryRespFnSense, Dakota::strbegins(), Dakota::strtolower(), and Model::userDefinedConstraints.

Model (LightWtBaseConstructor , ProblemDescDB & *problem_db,* ParallelLibrary & *parallel_lib*) `[protected]`

constructor initializing base class for recast model instances

This constructor also builds the base class data for inherited models. However, it is used for recast models which are instantiated on the fly. Therefore it only initializes a small subset of attributes.

13.65.3 Member Function Documentation

Model operator= (const Model & *model*)

assignment operator

Assignment operator decrements referenceCount for old modelRep, assigns new modelRep, and increments referenceCount for new modelRep.

References Model::modelRep, and Model::referenceCount.

Iterator & subordinate_iterator ( ) `[virtual]`

return the sub-iterator in nested and surrogate models

return by reference requires use of dummy objects, but is important to allow use of assign_rep() since this operation must be performed on the original envelope object.

Reimplemented in RecastModel, DataFitSurrModel, and NestedModel.

References Dakota::dummy_iterator, Model::modelRep, and Model::subordinate_iterator().

Referenced by NonDExpansion::compute_expansion(), NonDExpansion::compute_print_converged_results(), NonDExpansion::compute_print_iteration_results(), NonDExpansion::finalize_sets(), NonDGlobalReliability::get_best_sample(), NonDPolynomialChaos::increment_order(), NonDExpansion::increment_sets(), NonDPolynomialChaos::increment_specification_sequence(), NonDExpansion::increment_specification_sequence(), DOTOptimizer::initialize(), CONMINOptimizer::initialize(), NCSUOptimizer::initialize(), NLPQLPOptimizer::initialize(), NonDExpansion::initialize_expansion(), NonDExpansion::initialize_sets(), NonDStochCollocation::initialize_u_space_model(), NonDPolynomialChaos::initialize_u_space_model(), NonDExpansion::initialize_u_space_model(), SurrBasedLocalMinimizer::minimize_surrogates(), SurrBasedGlobalMinimizer::minimize_surrogates(), NonDLocalInterval::NonDLocalInterval(), NonDLocalReliability::NonDLocalReliability(), NonDGlobalReliability::optimize_gaussian_process(), NonDExpansion::refine_expansion(), SOLBase::SOLBase(), Model::subordinate_iterator(), RecastModel::subordinate_iterator(), and NonDStochCollocation::update_expansion().

Model & subordinate_model ( ) `[virtual]`

return a single sub-model defined from subModel in nested and recast models and truth_model() in surrogate models; used for a directed dive through model recursions that may bypass some components.

return by reference requires use of dummy objects, but is important to allow use of assign_rep() since this operation must be performed on the original envelope object.

Reimplemented in RecastModel, NestedModel, and SurrogateModel.

References Dakota::dummy_model, Model::modelRep, and Model::subordinate_model().

Referenced by Minimizer::data_transform_model(), NonDGlobalReliability::expected_feasibility(), NonDGlobalReliability::expected_improvement(), SurrogateModel::force_rebuild(), NonDExpansion::initialize_expansion(), Minimizer::initialize_run(), Minimizer::initialize_scaling(), NonDExpansion::initialize_u_space_model(), NonDGlobalReliability::optimize_gaussian_process(), LeastSq::post_run(), COLINOptimizer::post_run(), Optimizer::primary_resp_reducer(), LeastSq::primary_resp_weighter(), LeastSq::print_results(), Optimizer::print_results(), Minimizer::resize_best_resp_array(), Minimizer::resize_best_vars_array(), Minimizer::scale_model(), Model::subordinate_model(), DataFitSurrModel::update_global(), and LeastSq::weight_model().

Model & surrogate_model ( ) `[virtual]`

return the approximation sub-model in surrogate models

return by reference requires use of dummy objects, but is important to allow use of assign_rep() since this operation must be performed on the original envelope object.

Reimplemented in RecastModel, DataFitSurrModel, and HierarchSurrModel.

References Dakota::dummy_model, Model::modelRep, and Model::surrogate_model().

Referenced by NonDAdaptiveSampling::calc_score_delta_y(), NonDAdaptiveSampling::calc_score_topo_alm_hybrid(), NonDAdaptiveSampling::calc_score_topo_avg_persistence(), NonDAdaptiveSampling::calc_score_topo_bottleneck(), SurrBasedLocalMinimizer::find_center_approx(), SurrBasedLocalMinimizer::minimize_surrogates(), SurrBasedGlobalMinimizer::minimize_surrogates(), NonDAdaptiveSampling::output_round_data(), SurrBasedLocalMinimizer::SurrBasedLocalMinimizer(), Model::surrogate_model(), and RecastModel::surrogate_model().

Model & truth_model ( ) [virtual]

return the truth sub-model in surrogate models

return by reference requires use of dummy objects, but is important to allow use of assign_rep() since this operation must be performed on the original envelope object.

Reimplemented in RecastModel, DataFitSurrModel, and HierarchSurrModel.

References Dakota::dummy_model, Model::modelRep, and Model::truth_model().

Referenced by SurrogateModel::force_rebuild(), SurrBasedMinimizer::initialize_graphics(), SurrBasedLocalMinimizer::minimize_surrogates(), SurrBasedGlobalMinimizer::minimize_surrogates(), NonDLocalReliability::NonDLocalReliability(), SurrBasedMinimizer::print_results(), SurrogateModel::subordinate_model(), SurrBasedGlobalMinimizer::SurrBasedGlobalMinimizer(), SurrBasedLocalMinimizer::SurrBasedLocalMinimizer(), Model::truth_model(), and RecastModel::truth_model().

void update_from_subordinate_model (bool *recurse_flag* = `true`) [virtual]

propagate vars/labels/bounds/targets from the bottom up

used only for instantiate-on-the-fly model recursions (all RecastModel instantiations and alternate DataFitSurrModel instantiations). Single, Hierarchical, and Nested Models do not redefine the function since they do not support instantiate-on-the-fly. This means that the recursion will stop as soon as it encounters a Model that was instantiated normally, which is appropriate since ProblemDescDB-constructed Models use top-down information flow and do not require bottom-up updating.

Reimplemented in RecastModel, and DataFitSurrModel.

References Model::modelRep, and Model::update_from_subordinate_model().

Referenced by NonDLocalReliability::initialize_class_data(), NonDExpansion::initialize_expansion(), LeastSq::initialize_run(), Optimizer::initialize_run(), EffGlobalMinimizer::minimize_surrogates_on_model(), NonDGlobalReliability::optimize_gaussian_process(), NonDGlobalInterval::quantify_uncertainty(), NonDLocalInterval::quantify_uncertainty(), Model::update_from_subordinate_model(), DataFitSurrModel::update_from_subordinate_model(), and RecastModel::update_from_subordinate_model().

Interface & derived_interface ( ) [virtual]

return the interface employed by the derived model class, if present: SingleModel::userDefinedInterface, DataFitSurrModel::approxInterface, or NestedModel::optionalInterface

return by reference requires use of dummy objects, but is important to allow use of assign_rep() since this operation must be performed on the original envelope object.

Reimplemented in RecastModel, DataFitSurrModel, NestedModel, and SingleModel.

References Model::derived_interface(), Dakota::dummy_interface, and Model::modelRep.

Referenced by Model::derived_interface(), RecastModel::derived_interface(), and SurrBasedGlobalMinimizer::minimize_surrogates().

short local_eval_synchronization ( ) [virtual]

return derived model synchronization setting

```
SingleModels and HierarchSurrModels redefine this virtual function.  A
default value of "synchronous" prevents asynch local operations for:
```

- NestedModels: a subIterator can support message passing parallelism, but not asynch local.
- DataFitSurrModels: while asynch evals on approximations will work due to some added bookkeeping, avoiding them is preferable.

Reimplemented in RecastModel, NestedModel, and SingleModel.

References Model::local_eval_synchronization(), and Model::modelRep.

Referenced by Model::init_serial(), Model::local_eval_synchronization(), RecastModel::local_eval_synchronization(), and Model::set_ie_asynchronous_mode().

int local_eval_concurrency ( ) [virtual]

return derived model asynchronous evaluation concurrency

SingleModels and HierarchSurrModels redefine this virtual function.

Reimplemented in RecastModel, NestedModel, and SingleModel.

References Model::local_eval_concurrency(), and Model::modelRep.

Referenced by Model::local_eval_concurrency(), RecastModel::local_eval_concurrency(), and Model::set_ie_-asynchronous_mode().

const String & interface_id ( ) const [virtual]

return the interface identifier

return by reference requires use of dummy objects, but is important to allow use of assign_rep() since this operation must be performed on the original envelope object.

Reimplemented in RecastModel, DataFitSurrModel, NestedModel, and SingleModel.

References Dakota::dummy_interface, Interface::interface_id(), Model::interface_id(), and Model::modelRep.

Referenced by DataFitSurrModel::build_global(), Model::compute_response(), DataFitSurrModel::DataFit-SurrModel(), Model::estimate_derivatives(), Model::estimate_message_lengths(), SurrBasedLocalMinimizer::find-_center_approx(), Model::interface_id(), RecastModel::interface_id(), Optimizer::local_objective_recast_retrieve(), SurrBasedLocalMinimizer::minimize_surrogates(), SNLLLeastSq::post_run(), Analyzer::pre_output(), SurrBased-Minimizer::print_results(), LeastSq::print_results(), Optimizer::print_results(), SeqHybridMetaIterator::run_sequential(), DiscrepancyCorrection::search_db(), Model::synchronize(), Model::synchronize_nowait(), Analyzer::update_best(), ConcurrentMetaIterator::update_local_results(), SeqHybridMetaIterator::update_local_results(), and NonDLocal-Reliability::update_mpp_search_data().

bool evaluation_cache ( ) const [virtual]

Indicates the usage of an evaluation cache by the Model.

Only Models including ApplicationInterfaces support an evaluation cache: surrogate, nested, and recast mappings are not stored in the cache. Possible exceptions: HierarchSurrModel, NestedModel::optionalInterface.

Reimplemented in SingleModel.

References Model::evaluation_cache(), and Model::modelRep.

Referenced by DataFitSurrModel::DataFitSurrModel(), and Model::evaluation_cache().

void eval_tag_prefix (const String & *eval_id_str*) [virtual]

set the hierarchical eval ID tag prefix

Derived classes containing additional models or interfaces should implement this function to pass along to their sub Models/Interfaces.

Reimplemented in RecastModel, DataFitSurrModel, NestedModel, HierarchSurrModel, and SingleModel.

References Model::eval_tag_prefix(), and Model::modelRep.

Referenced by HierarchSurrModel::build_approximation(), HierarchSurrModel::derived_asynch_compute_response(), DataFitSurrModel::derived_asynch_compute_response(), HierarchSurrModel::derived_compute_response(), DataFitSurrModel::derived_compute_response(), Model::eval_tag_prefix(), RecastModel::eval_tag_prefix(), and Iterator::eval_tag_prefix().

ModelList & subordinate_models (bool *recurse_flag* = `true`)

return the sub-models in nested and surrogate models

since modelList is built with list insertions (using envelope copies), these models may not be used for model.assign_rep() since this operation must be performed on the original envelope object. They may, however, be used for letter-based operations (including assign_rep() on letter contents such as an interface).

References Model::derived_subordinate_models(), Model::modelList, Model::modelRep, and Model::subordinate_models().

Referenced by DOTOptimizer::initialize(), CONMINOptimizer::initialize(), NCSUOptimizer::initialize(), NLPQLPOptimizer::initialize(), NonDLocalInterval::NonDLocalInterval(), NonDLocalReliability::NonDLocalReliability(), SOLBase::SOLBase(), Model::subordinate_models(), and SurrBasedLocalMinimizer::SurrBasedLocalMinimizer().

void init_communicators (ParLevLIter *pl_iter*, int *max_eval_concurrency*, bool *recurse_flag* = `true`)

allocate communicator partitions for a model and store configuration in modelPCIterMap

The init_communicators() and derived_init_communicators() functions are stuctured to avoid performing the messageLengths estimation more than once. init_communicators() (not virtual) performs the estimation and then forwards the results to derived_init_communicators (virtual) which uses the data in different contexts.

References ParallelLibrary::bcast(), Model::derived_init_communicators(), Model::estimate_message_lengths(), ParallelLibrary::increment_parallel_configuration(), Model::init_communicators(), Model::initCommsBcastFlag, Model::messageLengths, Model::modelPCIter, Model::modelPCIterMap, Model::modelRep, ParallelLibrary::parallel_configuration_iterator(), ParallelLibrary::parallel_level_index(), and Model::parallelLib.

Referenced by EfficientSubspaceMethod::derived_init_communicators(), SurrBasedMinimizer::derived_init_communicators(), NonDGlobalReliability::derived_init_communicators(), NonDLocalInterval::derived_init_communicators(), NonDBayesCalibration::derived_init_communicators(), NonDExpansion::derived_init_communicators(), NonDGlobalInterval::derived_init_communicators(), NonDAdaptImpSampling::derived_init_communicators(), NonDPolynomialChaos::derived_init_communicators(), NonDAdaptiveSampling::derived_init_communicators(), NonDGPImpSampling::derived_init_communicators(), NonDLocalReliability::derived_init_communicators(), HierarchSurrModel::derived_init_communicators(), DataFitSurrModel::derived_init_communicators(), RecastModel::derived_init_communicators(), Iterator::derived_init_communicators(), DataFitSurrModel::derived_set_communicators(), Model::init_communicators(), EfficientSubspaceMethod::reduced_space_uq(), and Model::serve_init().

void init_serial ( )

for cases where init_communicators() will not be called, modify some default settings to behave properly in serial.

The init_serial() and derived_init_serial() functions are stuctured to separate base class (common) operations from derived class (specialized) operations.

References Model::asynchEvalFlag, Model::derived_init_serial(), Model::init_serial(), Model::local_eval_synchronization(), and Model::modelRep.

Referenced by HierarchSurrModel::derived_init_serial(), NestedModel::derived_init_serial(), DataFitSurrModel::derived_init_serial(), RecastModel::derived_init_serial(), and Model::init_serial().

void estimate_message_lengths ( )

estimate messageLengths for a model

This functionality has been pulled out of init_communicators() and defined separately so that it may be used in those cases when messageLengths is needed but model.init_communicators() is not called, e.g., for the master processor in the self-scheduling of a concurrent iterator strategy.

References Response::active_set_derivative_vector(), Response::copy(), Model::currentResponse, Model::currentVariables, Model::estimate_message_lengths(), Model::interface_id(), Model::messageLengths, Model::modelRep, Model::numFns, Model::parallelLib, MPIPackBuffer::reset(), MPIPackBuffer::size(), and ParallelLibrary::world_size().

Referenced by Model::estimate_message_lengths(), Model::init_communicators(), and ConcurrentMetaIterator::pre_run().

void assign_rep (Model * *model_rep*, bool *ref_count_incr* = `true`)

replaces existing letter with a new one

Similar to the assignment operator, the assign_rep() function decrements referenceCount for the old modelRep and assigns the new modelRep. It is different in that it is used for publishing derived class letters to existing envelopes, as opposed to sharing representations among multiple envelopes (in particular, assign_rep is passed a letter object and operator= is passed an envelope object). Letter assignment supports two models as governed by ref_count_incr:

- ref_count_incr = true (default): the incoming letter belongs to another envelope. In this case, increment the reference count in the normal manner so that deallocation of the letter is handled properly.
- ref_count_incr = false: the incoming letter is instantiated on the fly and has no envelope. This case is modeled after get_model(): a letter is dynamically allocated using new and passed into assign_rep, the letter's reference count is not incremented, and the letter is not remotely deleted (its memory management is passed over to the envelope).

References Dakota::abort_handler(), Model::model_rep(), Model::modelRep, and Model::referenceCount.

Referenced by Minimizer::data_transform_model(), EffGlobalMinimizer::EffGlobalMinimizer(), NonDAdaptiveSampling::NonDAdaptiveSampling(), NonDBayesCalibration::NonDBayesCalibration(), NonDGlobalInterval::NonDGlobalInterval(), NonDGlobalReliability::NonDGlobalReliability(), NonDGPImpSampling::NonDGPImpSampling(), NonDLocalInterval::NonDLocalInterval(), NonDLocalReliability::NonDLocalReliability(), NonDPolynomialChaos::NonDPolynomialChaos(), NonDStochCollocation::NonDStochCollocation(), Optimizer::reduce_model(), EfficientSubspaceMethod::reduced_space_uq(), Minimizer::scale_model(), SurrBasedLocalMinimizer::SurrBasedLocalMinimizer(), NonD::transform_model(), and LeastSq::weight_model().

int derivative_concurrency ( ) const

return the gradient concurrency for use in parallel configuration logic

This function assumes derivatives with respect to the active continuous variables. Therefore, concurrency with respect to the inactive continuous variables is not captured.

References Dakota::contains(), Model::derivative_concurrency(), Model::gradIdAnalytic, Model::gradientType, Model::hessianType, Model::hessIdNumerical, Model::intervalType, Model::methodSource, Model::modelRep, and Model::numDerivVars.

Referenced by Model::derivative_concurrency(), HierarchSurrModel::derived_free_communicators(), HierarchSurrModel::derived_init_communicators(), DataFitSurrModel::derived_init_communicators(), HierarchSurrModel::derived_set_communicators(), NonDExpansion::initialize_u_space_model(), Analyzer::num_samples(), and Iterator::update_from_model().

Real initialize_h (Real *x_j*, Real *lb_j*, Real *ub_j*, Real *step_size*, String *step_type*)

function to determine initial finite difference h (before step length adjustment) based on type of step desired

Auxiliary function to determine initial finite difference h (before step length adjustment) based on type of step desired.

Referenced by Model::estimate_derivatives().

Real FDstep1 (Real *x0_j*, Real *lb_j*, Real *ub_j*, Real *h_mag*)

function returning finite-difference step size (affected by bounds)

Auxiliary function to compute forward or first central-difference step size.

References Model::ignoreBounds, and Model::shortStep.

Referenced by Model::estimate_derivatives().

Real FDstep2 (Real *x0_j*, Real *lb_j*, Real *ub_j*, Real *h*)

function returning second central-difference step size (affected by bounds)

Auxiliary function to second central-difference step size, honoring bounds.

References Model::ignoreBounds, and Model::shortStep.

Referenced by Model::estimate_derivatives().

Model * get_model (ProblemDescDB & *problem_db*) `[private]`

Used by the envelope to instantiate the correct letter class.

Used only by the envelope constructor to initialize modelRep to the appropriate derived type, as given by the modelType attribute.

References ProblemDescDB::get_string(), Model::model_type(), and Model::modelType.

Referenced by Model::Model().

int estimate_derivatives (const ShortArray & *map_asv*, const ShortArray & *fd_grad_asv*, const ShortArray & *fd_hess_asv*, const ShortArray & *quasi_hess_asv*, const ActiveSet & *original_set*, const bool *asynch_flag*) `[private]`

evaluate numerical gradients using finite differences. This routine is selected with "method_source dakota" (the default method_source) in the numerical gradient specification.

Estimate derivatives by computing finite difference gradients, finite difference Hessians, and/or quasi-Newton Hessians. The total number of finite difference evaluations is returned for use by synchronize() to track response arrays, and it could be used to improve management of max_function_evaluations within the iterators. ! new logic

References Model::all_continuous_lower_bounds(), Model::all_continuous_upper_bounds(), Model::all_continuous_variable_ids(), Model::all_continuous_variable_types(), Variables::all_continuous_variables(), Model::centralHess, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Variables::continuous_variable_ids(), Model::continuous_variable_ids(), Model::continuous_variable_types(), Variables::continuous_variables(), Response::copy(), Dakota::copy_data(), Model::currentResponse, Model::currentVariables, Dakota::data_pairs, Model::dbCaptureList, Model::dbResponseList, Model::deltaList, ActiveSet::derivative_vector(), Model::derived_asynch_compute_response(), Model::derived_compute_response(), Model::fdGradStepSize, Model::fdGradStepType, Model::fdHessByFnStepSize, Model::fdHessByGradStepSize, Model::fdHessStepType, Model::FDstep1(), Model::FDstep2(),

Dakota::find_index(), Model::finite_difference_lower_bound(), Model::finite_difference_upper_bound(), Response::function_gradients(), Response::function_values(), Model::ignoreBounds, Model::inactive_continuous_lower_bounds(), Model::inactive_continuous_upper_bounds(), Variables::inactive_continuous_variable_ids(), Model::inactive_continuous_variable_ids(), Model::inactive_continuous_variable_types(), Variables::inactive_continuous_variables(), Model::initialize_h(), Model::initialMapList, Model::interface_id(), Model::intervalType, Dakota::lookup_by_val(), Model::numFns, Model::outputLevel, ActiveSet::request_vector(), Model::shortStep, and Model::update_response().

Referenced by Model::asynch_compute_response(), and Model::compute_response().

void synchronize_derivatives (const Variables & *vars,* const IntResponseMap & *fd_responses,* Response & *new_response,* const ShortArray & *fd_grad_asv,* const ShortArray & *fd_hess_asv,* const ShortArray & *quasi_hess_asv,* const ActiveSet & *original_set*) `[private]`

combine results from an array of finite difference response objects (fd_grad_responses) into a single response (new_response)

Merge an array of fd_responses into a single new_response. This function is used both by synchronous compute_response() for the case of asynchronous estimate_derivatives() and by synchronize() for the case where one or more asynch_compute_response() calls has employed asynchronous estimate_derivatives(). !

References Response::active_set(), Model::acv(), Variables::all_continuous_variable_ids(), Model::centralHess, Variables::continuous_variable_ids(), Response::copy(), Model::currentResponse, Model::currentVariables, Model::cv(), Model::dbCaptureList, Model::dbResponseList, Model::deltaList, ActiveSet::derivative_vector(), Dakota::find_index(), Response::function_gradients(), Response::function_values(), Model::icv(), Variables::inactive_continuous_variable_ids(), Model::initialMapList, Model::intervalType, Model::numFns, ActiveSet::request_values(), Response::reset_inactive(), and Model::update_response().

Referenced by Model::compute_response(), and Model::synchronize().

void update_response (const Variables & *vars,* Response & *new_response,* const ShortArray & *fd_grad_asv,* const ShortArray & *fd_hess_asv,* const ShortArray & *quasi_hess_asv,* const ActiveSet & *original_set,* Response & *initial_map_response,* const RealMatrix & *new_fn_grads,* const RealSymMatrixArray & *new_fn_hessians*) `[private]`

overlay results to update a response object

Overlay the initial_map_response with numerically estimated new_fn_grads and new_fn_hessians to populate new_response as governed by asv vectors. Quasi-Newton secant Hessian updates are also performed here, since this is where the gradient data needed for the updates is first consolidated. Convenience function used by estimate_derivatives() for the synchronous case and by synchronize_derivatives() for the asynchronous case.

References Response::active_set_request_vector(), Variables::continuous_variable_ids(), Response::copy(), Model::currentResponse, Model::currentVariables, ActiveSet::derivative_vector(), Response::function_gradients(), Response::function_hessians(), Response::function_values(), Model::hessianType, Model::hessIdQuasi, Response::is_null(), Model::numFns, Model::outputLevel, Model::quasiHessians, ActiveSet::request_vector(), Response::reset_inactive(), Model::supportsEstimDerivs, Model::surrogate_response_mode(), and Model::update_quasi_hessians().

Referenced by Model::estimate_derivatives(), and Model::synchronize_derivatives().

void update_quasi_hessians (const Variables & *vars,* Response & *new_response,* const ActiveSet & *original_set*) `[private]`

perform quasi-Newton Hessian updates

quasi-Newton updates are performed for approximating response function Hessians using BFGS or SR1 formulations. These Hessians are supported only for the active continuous variables, and a check is performed on the DVV prior to invoking the function.

References Dakota::contains(), Variables::continuous_variables(), Dakota::copy_data(), Model::fnGradsPrev, Response::function_gradients(), Model::hessianType, Model::hessIdQuasi, Model::modelType, Model::numDerivVars, Model::numFns, Model::numQuasiUpdates, Model::outputLevel, Model::quasiHessians, Model::quasiHessType, ActiveSet::request_vector(), and Model::xPrev.

Referenced by Model::update_response().

bool manage_asv (const ShortArray & *asv_in,* ShortArray & *map_asv_out,* ShortArray & *fd_grad_asv_out,* ShortArray & *fd_hess_asv_out,* ShortArray & *quasi_hess_asv_out*) [private]

Coordinates usage of estimate_derivatives() calls based on asv_in.

Splits asv_in total request into map_asv_out, fd_grad_asv_out, fd_hess_asv_out, and quasi_hess_asv_out as governed by the responses specification. If the returned use_est_deriv is true, then these asv outputs are used by estimate_derivatives() for the initial map, finite difference gradient evals, finite difference Hessian evals, and quasi-Hessian updates, respectively. If the returned use_est_deriv is false, then only map_asv_out is used.

References Dakota::abort_handler(), Dakota::contains(), Model::gradIdAnalytic, Model::gradIdNumerical, Model::gradientType, Model::hessianType, Model::hessIdAnalytic, Model::hessIdNumerical, Model::hessIdQuasi, Model::intervalType, Model::methodSource, Model::supportsEstimDerivs, and Model::surrogate_response_mode().

Referenced by Model::asynch_compute_response(), and Model::compute_response().

13.65.4 Member Data Documentation

RealVector fdGradStepSize [protected]

relative finite difference step size for numerical gradients

A scalar value (instead of the vector fd_gradient_step_size spec) is used within the iterator hierarchy since this attribute is only used to publish a step size to vendor numerical gradient algorithms.

Referenced by DataFitSurrModel::DataFitSurrModel(), Model::estimate_derivatives(), Model::fd_gradient_step_size(), RecastModel::initialize_data_from_submodel(), and Model::Model().

RealVector fdHessByGradStepSize [protected]

relative finite difference step size for numerical Hessians estimated using first-order differences of gradients

For vendor numerical Hessian algorithms, a scalar value is used.

Referenced by DataFitSurrModel::DataFitSurrModel(), Model::estimate_derivatives(), Model::fd_hessian_by_grad_step_size(), RecastModel::initialize_data_from_submodel(), and Model::Model().

RealVector fdHessByFnStepSize [protected]

relative finite difference step size for numerical Hessians estimated using second-order differences of function values

For vendor numerical Hessian algorithms, a scalar value is used.

Referenced by DataFitSurrModel::DataFitSurrModel(), Model::estimate_derivatives(), Model::fd_hessian_by_fn_step_size(), RecastModel::initialize_data_from_submodel(), and Model::Model().

ProblemDescDB& probDescDB [protected]

class member reference to the problem description database

Iterator and Model cannot use a shallow copy of ProblemDescDB due to circular destruction dependency (reference counts can't get to 0), since ProblemDescDB contains {iterator,model}List.

Referenced by HierarchSurrModel::derived_init_communicators(), NestedModel::derived_init_communicators(), DataFitSurrModel::derived_init_communicators(), NestedModel::derived_init_serial(), Model::problem_description_db(), and NestedModel::update_sub_iterator().

The documentation for this class was generated from the following files:

- DakotaModel.hpp
- DakotaModel.cpp

13.66 MPIManager Class Reference

Class MPIManager to manage Dakota's MPI world, which may be a subset of MPI_COMM_WORLD.

Public Member Functions

- MPIManager ()

  Default constructor; Dakota will not call MPI_Init.
- MPIManager (int &argc, char **&argv)

  Command-line constructor; parses MPI arguments during call to MPI_Init.
- MPIManager (MPI_Comm dakota_mpi_comm)

  Construct on specified MPI_Comm.
- ~MPIManager ()

  destructor: calls finalize if Dakota owns MPI
- MPI_Comm dakota_mpi_comm () const

  get the MPI_Comm on which Dakota is running
- int world_rank () const

  get the rank of this process in Dakota's MPI_Comm
- int world_size () const

  get the size of the MPI_Comm on which Dakota is running
- bool mpirun_flag () const

  true when Dakota is running in MPI mode

Static Public Member Functions

- static bool detect_parallel_launch (int &argc, char **&argv)

  detect parallel launch of Dakota using mpirun/mpiexec/poe/etc. based on command line arguments and environment variables

Private Attributes

- MPI_Comm dakotaMPIComm

  MPI_Comm on which DAKOTA is running.
- int dakotaWorldRank

  rank in MPI_Comm in which DAKOTA is running
- int dakotaWorldSize

  size of MPI_Comm in which DAKOTA is running
- bool mpirunFlag

flag for a parallel mpirun/yod launch

- bool ownMPIFlag

  flag for ownership of MPI_Init/MPI_Finalize

13.66.1 Detailed Description

Class MPIManager to manage Dakota's MPI world, which may be a subset of MPI_COMM_WORLD.

The documentation for this class was generated from the following files:

- MPIManager.hpp
- MPIManager.cpp

13.67 MPIPackBuffer Class Reference

Class for packing MPI message buffers.

Public Member Functions

- MPIPackBuffer (int size_=1024)

  Constructor, which allows the default buffer size to be set.
- ~MPIPackBuffer ()

  Desctructor.
- const char * buf ()

  Returns a pointer to the internal buffer that has been packed.
- int size ()

  The number of bytes of packed data.
- int capacity ()

  the allocated size of Buffer.
- void reset ()

  Resets the buffer index in order to reuse the internal buffer.
- void pack (const int *data, const int num=1)

  *Pack one or more **int's**.*
- void pack (const u_int *data, const int num=1)

  *Pack one or more **unsigned int's**.*
- void pack (const long *data, const int num=1)

  *Pack one or more **long's**.*
- void pack (const u_long *data, const int num=1)

  *Pack one or more **unsigned long's**.*
- void pack (const short *data, const int num=1)

  *Pack one or more **short's**.*
- void pack (const u_short *data, const int num=1)

  *Pack one or more **unsigned short's**.*
- void pack (const char *data, const int num=1)

  *Pack one or more **char's**.*
- void pack (const u_char *data, const int num=1)

*Pack one or more **unsigned char's**.*

- void pack (const double *data, const int num=1)

  *Pack one or more **double's**.*
- void pack (const float *data, const int num=1)

  *Pack one or more **float's**.*
- void pack (const bool *data, const int num=1)

  *Pack one or more **bool's**.*
- void pack (const int &data)

  *Pack a **int**.*
- void pack (const u_int &data)

  *Pack a **unsigned int**.*
- void pack (const long &data)

  *Pack a **long**.*
- void pack (const u_long &data)

  *Pack a **unsigned long**.*
- void pack (const short &data)

  *Pack a **short**.*
- void pack (const u_short &data)

  *Pack a **unsigned short**.*
- void pack (const char &data)

  *Pack a **char**.*
- void pack (const u_char &data)

  *Pack a **unsigned char**.*
- void pack (const double &data)

  *Pack a **double**.*
- void pack (const float &data)

  *Pack a **float**.*
- void pack (const bool &data)

  *Pack a **bool**.*

Protected Member Functions

- void resize (const int newsize)

  Resizes the internal buffer.

Protected Attributes

- char * Buffer

  The internal buffer for packing.
- int Index

  The index into the current buffer.
- int Size

  The total size that has been allocated for the buffer.

13.67.1 Detailed Description

Class for packing MPI message buffers.

A class that provides a facility for packing message buffers using the MPI_Pack facility. The `MPIPackBuffer` class dynamically resizes the internal buffer to contain enough memory to pack the entire object. When deleted, the `MPIPackBuffer` object deletes this internal buffer. This class is based on the Dakota_Version_3_0 version of utilib::PackBuffer from utilib/src/io/PackBuf.[cpp,h]

The documentation for this class was generated from the following files:

- MPIPackBuffer.hpp
- MPIPackBuffer.cpp

13.68 MPIUnpackBuffer Class Reference

Class for unpacking MPI message buffers.

Public Member Functions

- void setup (char *buf_, int size_, bool flag_=false)

  Method that does the setup for the constructors.
- MPIUnpackBuffer ()

  Default constructor.
- MPIUnpackBuffer (int size_)

  Constructor that specifies the size of the buffer.
- MPIUnpackBuffer (char *buf_, int size_, bool flag_=false)

  Constructor that sets the internal buffer to the given array.
- ~MPIUnpackBuffer ()

  Destructor.
- void resize (const int newsize)

  Resizes the internal buffer.
- const char * buf ()

  Returns a pointer to the internal buffer.
- int size ()

  Returns the length of the buffer.
- int curr ()

  Returns the number of bytes that have been unpacked from the buffer.
- void reset ()

  Resets the index of the internal buffer.
- void unpack (int *data, const int num=1)

  *Unpack one or more **int's**.*
- void unpack (u_int *data, const int num=1)

  *Unpack one or more **unsigned int's**.*
- void unpack (long *data, const int num=1)

  *Unpack one or more **long's**.*
- void unpack (u_long *data, const int num=1)

*Unpack one or more **unsigned long's**.*

- void unpack (short *data, const int num=1)

  *Unpack one or more **short's**.*

- void unpack (u_short *data, const int num=1)

  *Unpack one or more **unsigned short's**.*

- void unpack (char *data, const int num=1)

  *Unpack one or more **char's**.*

- void unpack (u_char *data, const int num=1)

  *Unpack one or more **unsigned char's**.*

- void unpack (double *data, const int num=1)

  *Unpack one or more **double's**.*

- void unpack (float *data, const int num=1)

  *Unpack one or more **float's**.*

- void unpack (bool *data, const int num=1)

  *Unpack one or more **bool's**.*

- void unpack (int &data)

  *Unpack a **int**.*

- void unpack (u_int &data)

  *Unpack a **unsigned int**.*

- void unpack (long &data)

  *Unpack a **long**.*

- void unpack (u_long &data)

  *Unpack a **unsigned long**.*

- void unpack (short &data)

  *Unpack a **short**.*

- void unpack (u_short &data)

  *Unpack a **unsigned short**.*

- void unpack (char &data)

  *Unpack a **char**.*

- void unpack (u_char &data)

  *Unpack a **unsigned char**.*

- void unpack (double &data)

  *Unpack a **double**.*

- void unpack (float &data)

  *Unpack a **float**.*

- void unpack (bool &data)

  *Unpack a **bool**.*

Protected Attributes

- char * Buffer

  The internal buffer for unpacking.
- int Index

  The index into the current buffer.
- int Size

  The total size that has been allocated for the buffer.
- bool ownFlag

  If `TRUE`*, then this class owns the internal buffer.*

13.68.1 Detailed Description

Class for unpacking MPI message buffers.

A class that provides a facility for unpacking message buffers using the MPI_Unpack facility. This class is based on the Dakota_Version_3_0 version of utilib::UnPackBuffer from utilib/src/io/PackBuf.[cpp,h]

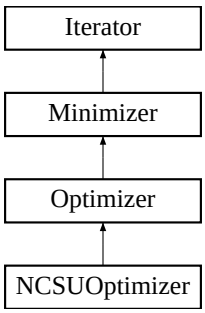
The documentation for this class was generated from the following files:

- MPIPackBuffer.hpp
- MPIPackBuffer.cpp

13.69 NCSUOptimizer Class Reference

Wrapper class for the NCSU DIRECT optimization library.

Inheritance diagram for NCSUOptimizer:



Public Member Functions

- NCSUOptimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NCSUOptimizer (Model &model, const int &max_iter, const int &max_eval, double min_box_size=-1., double vol_box_size=-1., double solution_target=-DBL_MAX)

  alternate constructor for instantiations "on the fly"
- NCSUOptimizer (Model &model)

  alternate constructor for Iterator instantiations by name

- NCSUOptimizer (const RealVector &var_l_bnds, const RealVector &var_u_bnds, const int &max_iter, const int &max_eval, double(∗user_obj_eval)(const RealVector &x), double min_box_size=-1., double vol_box_-size=-1., double solution_target=-DBL_MAX)

  alternate constructor for instantiations "on the fly"

- ∼NCSUOptimizer ()

  destructor

- void find_optimum ()

  Used within the optimizer branch for computing the optimal solution. Redefines the run virtual function for the optimizer branch.

Private Member Functions

- void initialize ()

  shared code among model-based constructors

- void check_inputs ()

  verify problem respects NCSU DIRECT Fortran limits

Static Private Member Functions

- static int objective_eval (int ∗n, double c[ ], double l[ ], double u[ ], int point[ ], int ∗maxI, int ∗start, int ∗maxfunc, double fvec[ ], int iidata[ ], int ∗iisize, double ddata[ ], int ∗idsize, char cdata[ ], int ∗icsize)

  'fep' in Griffin-modified NCSUDirect: computes the value of the objective function (potentially at multiple points, passed by function pointer to NCSUDirect). Include unscaling from DIRECT.

Private Attributes

- short setUpType

  controls iteration mode: SETUP_MODEL (normal usage) or SETUP_USERFUNC (user-supplied functions mode for "on the fly" instantiations). see enum in NCSUOptimizer.cpp NonDGlobalReliability currently uses the model mode. GaussProcApproximation currently uses the user_functions mode.

- Real minBoxSize

  holds the minimum boxsize

- Real volBoxSize

  hold the minimum volume boxsize

- Real solutionTarget

  holds the solution target minimum to drive towards

- RealVector lowerBounds

  holds variable lower bounds passed in for "user_functions" mode.

- RealVector upperBounds

  holds variable upper bounds passed in for "user_functions" mode.

- double(∗ userObjectiveEval)(const RealVector &x)

  holds function pointer for objective function evaluator passed in for "user_functions" mode.

Static Private Attributes

- static NCSUOptimizer * ncsudirectInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.69.1 Detailed Description

Wrapper class for the NCSU DIRECT optimization library.

The NCSUOptimizer class provides a wrapper for a Fortran 77 implementation of the DIRECT algorithm developed at North Carolina State University. It uses a function pointer approach for which passed functions must be either global functions or static member functions. Any attribute used within static member functions must be either local to that function or accessed through a static pointer.

The user input mappings are as follows:

13.69.2 Constructor & Destructor Documentation

NCSUOptimizer (ProblemDescDB & *problem_db*, Model & *model*)

standard constructor

This is the standard constructor with method specification support.

References NCSUOptimizer::check_inputs(), and NCSUOptimizer::initialize().

NCSUOptimizer (Model & *model*, const int & *max_iter*, const int & *max_eval*, double *min_box_size* = `-1.`, double *vol_box_size* = `-1.`, double *solution_target* = `-DBL_MAX`)

alternate constructor for instantiations "on the fly"

This is an alternate constructor for instantiations on the fly using a Model but no ProblemDescDB.

References NCSUOptimizer::check_inputs(), NCSUOptimizer::initialize(), Iterator::maxFunctionEvals, and Iterator::maxIterations.

NCSUOptimizer (Model & *model*)

alternate constructor for Iterator instantiations by name

This is an alternate constructor for Iterator instantiations by name using a Model but no ProblemDescDB.

References NCSUOptimizer::check_inputs(), and NCSUOptimizer::initialize().

NCSUOptimizer (const RealVector & *var_l_bnds*, const RealVector & *var_u_bnds*, const int & *max_iter*, const int & *max_eval*, double(*)(const RealVector &x) *user_obj_eval*, double *min_box_size* = `-1.`, double *vol_box_size* = `-1.`, double *solution_target* = `-DBL_MAX`)

alternate constructor for instantiations "on the fly"

This is an alternate constructor for performing an optimization using the passed in objective function pointer.

References NCSUOptimizer::check_inputs(), Iterator::maxFunctionEvals, and Iterator::maxIterations.

13.69.3 Member Function Documentation

int objective_eval (int ∗ *n*, double *c[ ]*, double *l[ ]*, double *u[ ]*, int *point[ ]*, int ∗ *maxI*, int ∗ *start*, int ∗ *maxfunc*, double *fvec[ ]*, int *iidata[ ]*, int ∗ *iisize*, double *ddata[ ]*, int ∗ *idsize*, char *cdata[ ]*, int ∗ *icsize*) [static], [private]

'fep' in Griffin-modified NCSUDirect: computes the value of the objective function (potentially at multiple points, passed by function pointer to NCSUDirect). Include unscaling from DIRECT.

Modified batch evaluator that accepts multiple points and returns corresponding vector of functions in fvec. Must be used with modified DIRECT src (DIRbatch.f).

References Model::asynch_compute_response(), Model::asynch_flag(), Model::compute_response(), Model::continuous_variables(), Model::current_response(), Response::function_value(), Iterator::iteratedModel, NCSUOptimizer::ncsudirectInstance, Model::primary_response_fn_sense(), NCSUOptimizer::setUpType, Model::synchronize(), and NCSUOptimizer::userObjectiveEval.

Referenced by NCSUOptimizer::find_optimum().

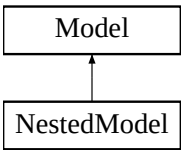
The documentation for this class was generated from the following files:

- NCSUOptimizer.hpp
- NCSUOptimizer.cpp

13.70 NestedModel Class Reference

Derived model class which performs a complete sub-iterator execution within every evaluation of the model.

Inheritance diagram for NestedModel:



Public Member Functions

- NestedModel (ProblemDescDB &problem_db)

  constructor
- ∼NestedModel ()

  destructor

Protected Member Functions

- void derived_compute_response (const ActiveSet &set)

  portion of compute_response() specific to NestedModel
- void derived_asynch_compute_response (const ActiveSet &set)

  portion of asynch_compute_response() specific to NestedModel
- const IntResponseMap & derived_synchronize ()

  portion of synchronize() specific to NestedModel
- Iterator & subordinate_iterator ()

  return subIterator

- Model & subordinate_model ()

  return subModel
- void derived_subordinate_models (ModelList &ml, bool recurse_flag)

  return subModel
- Interface & derived_interface ()

  return optionalInterface
- void surrogate_response_mode (short mode)

  pass a bypass request on to the subModel for any lower-level surrogates
- void component_parallel_mode (short mode)

  update component parallel mode for supporting parallelism in optionalInterface and subModel
- size_t mi_parallel_level_index () const

  return subIteratorSched.miPLIndex
- short local_eval_synchronization ()

  return optionalInterface synchronization setting
- int local_eval_concurrency ()

  return optionalInterface asynchronous evaluation concurrency
- bool derived_master_overload () const

  flag which prevents overloading the master with a multiprocessor evaluation (forwarded to optionalInterface)
- void derived_init_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set up optionalInterface and subModel for parallel operations
- void derived_init_serial ()

  set up optionalInterface and subModel for serial operations.
- void derived_set_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set active parallel configuration within subModel
- void derived_free_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  deallocate communicator partitions for the NestedModel (forwarded to optionalInterface and subModel)
- void serve_run (ParLevLIter pl_iter, int max_eval_concurrency)

  Service optionalInterface and subModel job requests received from the master. Completes when a termination message is received from stop_servers().
- void stop_servers ()

  Executed by the master to terminate server operations for subModel and optionalInterface when iteration on the NestedModel is complete.
- const String & interface_id () const

  return the optionalInterface identifier
- int evaluation_id () const

  Return the current evaluation id for the NestedModel.
- void set_evaluation_reference ()

  set the evaluation counter reference points for the NestedModel (request forwarded to optionalInterface and subModel)
- void fine_grained_evaluation_counters ()

  request fine-grained evaluation reporting within optionalInterface and subModel
- void print_evaluation_summary (std::ostream &s, bool minimal_header=false, bool relative_count=true) const

  print the evaluation summary for the NestedModel (request forwarded to optionalInterface and subModel)

- void eval_tag_prefix (const String &eval_id_str)

  set the hierarchical eval ID tag prefix
- void **initialize_iterator** (int job_index)
- void **pack_parameters_buffer** (MPIPackBuffer &send_buffer, int job_index)
- void **unpack_parameters_buffer** (MPIUnpackBuffer &recv_buffer)
- void **unpack_parameters_initialize** (MPIUnpackBuffer &recv_buffer)
- void **pack_results_buffer** (MPIPackBuffer &send_buffer, int job_index)
- void **unpack_results_buffer** (MPIUnpackBuffer &recv_buffer, int job_index)
- void **update_local_results** (int job_index)

Private Member Functions

- void update_sub_iterator ()

  update subIterator with mapping data and set subIterator-based counts
- void initialize_iterator (const Variables &vars, const ActiveSet &set, int eval_id)

  lower level function shared by initialize_iterator(int) and unpack_parameters_initialize()
- void unpack (MPIUnpackBuffer &recv_buffer, Variables &vars, ActiveSet &set, int &eval_id)

  lower level function shared by unpack_parameters_buffer() and unpack_parameters_initialize()
- void resolve_real_variable_mapping (const String &map1, const String &map2, size_t curr_index, short &inactive_sm_view)

  for a named real mapping, resolve primary index and secondary target
- void resolve_integer_variable_mapping (const String &map1, const String &map2, size_t curr_index, short &inactive_sm_view)

  for a named integer mapping, resolve primary index and secondary target
- void resolve_string_variable_mapping (const String &map1, const String &map2, size_t curr_index, short &inactive_sm_view)

  for a named string mapping, resolve primary index and secondary target
- size_t sm_acv_index_map (size_t pacvm_index, short sacvm_target)

  offset pacvm_index based on sacvm_target to create mapped_index
- size_t sm_adiv_index_map (size_t padivm_index, short sadivm_target)

  offset padivm_index based on sadivm_target to create mapped_index
- size_t sm_adsv_index_map (size_t padsvm_index, short sadsvm_target)

  offset padsvm_index based on sadsvm_target to create mapped_index
- size_t sm_adrv_index_map (size_t padrvm_index, short sadrvm_target)

  offset padrvm_index based on sadrvm_target to create mapped_index
- size_t cv_index_map (size_t cv_index, const Variables &vars)

  offset cv_index to create index into aggregated primary/secondary arrays
- size_t div_index_map (size_t div_index, const Variables &vars)

  offset div_index to create index into aggregated primary/secondary arrays
- size_t dsv_index_map (size_t dsv_index, const Variables &vars)

  offset dsv_index to create index into aggregated primary/secondary arrays
- size_t drv_index_map (size_t drv_index, const Variables &vars)

  offset drv_index to create index into aggregated primary/secondary arrays
- size_t ccv_index_map (size_t ccv_index, const Variables &vars)

offset active complement ccv_index to create index into all continuous arrays

- size_t cdiv_index_map (size_t cdiv_index, const Variables &vars)

  offset active complement cdiv_index to create index into all discrete int arrays
- size_t cdsv_index_map (size_t cdsv_index, const Variables &vars)

  offset active complement cdsv_index to create index into all discrete string arrays
- size_t cdrv_index_map (size_t cdrv_index, const Variables &vars)

  offset active complement cdrv_index to create index into all discrete real arrays
- void real_variable_mapping (const Real &r_var, size_t mapped_index, short svm_target)

  insert r_var into appropriate recipient
- void integer_variable_mapping (const int &i_var, size_t mapped_index, short svm_target)

  insert i_var into appropriate recipient
- void string_variable_mapping (const String &s_var, size_t mapped_index, short svm_target)

  insert s_var into appropriate recipient
- void set_mapping (const ActiveSet &mapped_set, ActiveSet &interface_set, bool &opt_interface_map, ActiveSet &sub_iterator_set, bool &sub_iterator_map)

  define the evaluation requirements for the optionalInterface (interface_set) and the subIterator (sub_iterator_set) from the total model evaluation requirements (mapped_set)
- void response_mapping (const Response &interface_response, const Response &sub_iterator_response, Response &mapped_response)

  combine the response from the optional interface evaluation with the response from the sub-iteration using the primaryCoeffs/secondaryCoeffs mappings to create the total response for the model
- void interface_response_overlay (const Response &opt_interface_response, Response &mapped_response)

  assign the response from the optional interface evaluation within the total response for the model
- void iterator_response_overlay (const Response &sub_iterator_response, Response &mapped_response)

  overlay the sub-iteration response within the total response for the model using the primaryCoeffs/secondaryCoeffs mappings
- Response & find_nested_response (int nested_cntr)

  locate existing or allocate new entry in nestedResponseMap
- void check_response_map (const ShortArray &mapped_asv)

  check function counts for the mapped_asv
- void update_inactive_view (short new_view, short &view)

  update inactive variables view for subIterator based on new_view
- void update_inactive_view (unsigned short type, short &view)

  update inactive variables view for subIterator based on type
- void update_sub_model (const Variables &vars, const Constraints &cons)

  update subModel with current variable values/bounds/labels

Private Attributes

- int nestedModelEvalCntr

  number of calls to derived_compute_response()/ derived_asynch_compute_response()
- bool firstUpdate

  boolean to trigger one-time updates on first call to update_sub_model()
- IntResponseMap nestedResponseMap

used to return a map of nested responses (including subIterator and optionalInterface contributions) for aggregation and rekeying at the base class level

- size_t outerMIPLIndex

  the miPLIndex for the outer parallelism context, prior to any subIterator partitioning
- Iterator subIterator

  the sub-iterator that is executed on every evaluation of this model
- String subMethodPointer

  the sub-method pointer from the nested model specification
- Model subModel

  the sub-model used in sub-iterator evaluations
- PRPQueue subIteratorPRPQueue

  job queue for asynchronous execution of subIterator jobs
- IteratorScheduler subIteratorSched

  scheduling object for concurrent iterator parallelism
- int subIteratorJobCntr

  subIterator job counter since last synchronize()
- IntIntMap subIteratorIdMap

  mapping from subIterator evaluation counter to nested model counter (different when subIterator evaluations do not occur on every nested model evaluation due to variable ASV content)
- size_t numSubIterFns

  number of sub-iterator response functions prior to mapping
- size_t numSubIterMappedIneqCon

  number of top-level inequality constraints mapped from the sub-iteration results
- size_t numSubIterMappedEqCon

  number of top-level equality constraints mapped from the sub-iteration results
- Interface optionalInterface

  the optional interface contributes nonnested response data to the total model response
- String optInterfacePointer

  the optional interface pointer from the nested model specification
- Response optInterfaceResponse

  the response object resulting from optional interface evaluations
- IntIntMap optInterfaceIdMap

  mapping from optionalInterface evaluation counter to nested model counter (different when optionalInterface evaluations do not occur on every nested model evaluation due to variable ASV content)
- size_t numOptInterfPrimary

  number of primary response functions (objective/least squares/generic functions) resulting from optional interface evaluations
- size_t numOptInterfIneqCon

  number of inequality constraints resulting from optional interface evaluations
- size_t numOptInterfEqCon

  number of equality constraints resulting from the optional interface evaluations
- SizetArray active1ACVarMapIndices

  "primary" variable mappings for inserting active continuous currentVariables within all continuous subModel variables. If there are no secondary mappings defined, then the insertions replace the subModel variable values.

- SizetArray active1ADIVarMapIndices

  "primary" variable mappings for inserting active discrete int currentVariables within all discrete int subModel variables. No secondary mappings are defined for discrete int variables, so the active variables replace the subModel variable values.

- SizetArray active1ADSVarMapIndices

  "primary" variable mappings for inserting active discrete string currentVariables within all discrete string subModel variables. No secondary mappings are defined for discrete string variables, so the active variables replace the subModel variable values.

- SizetArray active1ADRVarMapIndices

  "primary" variable mappings for inserting active discrete real currentVariables within all discrete real subModel variables. No secondary mappings are defined for discrete real variables, so the active variables replace the subModel variable values.

- ShortArray active2ACVarMapTargets

  "secondary" variable mappings for inserting active continuous currentVariables into sub-parameters (e.g., distribution parameters for uncertain variables or bounds for continuous design/state variables) within all continuous subModel variables.

- ShortArray active2ADIVarMapTargets

  "secondary" variable mappings for inserting active discrete int currentVariables into sub-parameters (e.g., bounds for discrete design/state variables) within all discrete int subModel variables.

- ShortArray active2ADSVarMapTargets

  "secondary" variable mappings for inserting active discrete string currentVariables into sub-parameters (e.g., bounds for discrete design/state variables) within all discrete string subModel variables.

- ShortArray active2ADRVarMapTargets

  "secondary" variable mappings for inserting active discrete real currentVariables into sub-parameters (e.g., bounds for discrete design/state variables) within all discrete real subModel variables.

- SizetArray complement1ACVarMapIndices

  "primary" variable mappings for inserting the complement of the active continuous currentVariables within all continuous subModel variables

- SizetArray complement1ADIVarMapIndices

  "primary" variable mappings for inserting the complement of the active discrete int currentVariables within all discrete int subModel variables

- SizetArray complement1ADSVarMapIndices

  "primary" variable mappings for inserting the complement of the active discrete string currentVariables within all discrete string subModel variables

- SizetArray complement1ADRVarMapIndices

  "primary" variable mappings for inserting the complement of the active discrete real currentVariables within all discrete real subModel variables

- BitArray extraCVarsData

  flags for updating subModel continuous bounds and labels, one for each active continuous variable in currentVariables

- BitArray extraDIVarsData

  flags for updating subModel discrete int bounds and labels, one for each active discrete int variable in currentVariables

- BitArray extraDSVarsData

  flags for updating subModel discrete string labels, one for each active discrete string variable in currentVariables

- BitArray extraDRVarsData

flags for updating subModel discrete real bounds and labels, one for each active discrete real variable in currentVariables

- RealMatrix primaryRespCoeffs

  "primary" response_mapping matrix applied to the sub-iterator response functions. For OUU, the matrix is applied to UQ statistics to create contributions to the top-level objective functions/least squares/ generic response terms.

- RealMatrix secondaryRespCoeffs

  "secondary" response_mapping matrix applied to the sub-iterator response functions. For OUU, the matrix is applied to UQ statistics to create contributions to the top-level inequality and equality constraints.

- String evalTagPrefix

  cached evalTag Prefix from parents to use at compute_response time

Friends

- class IteratorScheduler

  protect scheduler callback functions from general access

Additional Inherited Members

13.70.1 Detailed Description

Derived model class which performs a complete sub-iterator execution within every evaluation of the model.

The NestedModel class nests a sub-iterator execution within every model evaluation. This capability is most commonly used for optimization under uncertainty, in which a nondeterministic iterator is executed on every optimization function evaluation. The NestedModel also contains an optional interface, for portions of the model evaluation which are independent from the sub-iterator, and a set of mappings for combining sub-iterator and optional interface data into a top level response for the model.

13.70.2 Member Function Documentation

void derived_compute_response (const ActiveSet & *set*) `[protected], [virtual]`

portion of compute_response() specific to NestedModel

Update subModel's inactive variables with active variables from currentVariables, compute the optional interface and sub-iterator responses, and map these to the total model response.

Reimplemented from Model.

References NestedModel::active2ACVarMapTargets, Response::active_set(), ParallelLibrary::bcast(), ParallelLibrary::bcast_hs(), NestedModel::component_parallel_mode(), Model::currentResponse, Model::currentVariables, Interface::eval_tag_prefix(), Iterator::eval_tag_prefix(), NestedModel::evalTagPrefix, Model::hierarchicalTagging, NestedModel::interface_response_overlay(), NestedModel::iterator_response_overlay(), IteratorScheduler::iteratorCommSize, IteratorScheduler::iteratorScheduling, Interface::map(), IteratorScheduler::messagePass, IteratorScheduler::miPLIndex, Model::modelPCIter, NestedModel::nestedModelEvalCntr, NestedModel::optInterfaceResponse, NestedModel::optionalInterface, Model::outputLevel, ParallelLibrary::parallel_configuration_iterator(), Model::parallelLib, IteratorScheduler::peerAssignJobs, Response::reset(), Iterator::response_results(), Iterator::response_results_active_set(), Iterator::run(), IteratorScheduler::run_iterator(), NestedModel::set_mapping(), IteratorScheduler::stop_iterator_servers(), NestedModel::subIterator, NestedModel::subIteratorSched, NestedModel::update_sub_model(), and Model::userDefinedConstraints.

void derived_asynch_compute_response (const ActiveSet & *set*) [protected], [virtual]

portion of asynch_compute_response() specific to NestedModel

Asynchronous execution of subIterator on subModel and, optionally, optionalInterface.

Reimplemented from Model.

References Response::active_set(), Model::currentResponse, Model::currentVariables, Interface::evaluation_id(), Interface::map(), Iterator::method_id(), NestedModel::nestedModelEvalCntr, NestedModel::optInterfaceIdMap, NestedModel::optInterfaceResponse, NestedModel::optionalInterface, Iterator::response_results(), Iterator::response_results_active_set(), NestedModel::set_mapping(), NestedModel::subIterator, NestedModel::subIteratorIdMap, NestedModel::subIteratorJobCntr, and NestedModel::subIteratorPRPQueue.

const IntResponseMap & derived_synchronize ( ) [protected], [virtual]

portion of synchronize() specific to NestedModel

Recovery of asynchronous subIterator executions and, optionally, asynchronous optionalInterface mappings.

Reimplemented from Model.

References NestedModel::component_parallel_mode(), NestedModel::find_nested_response(), NestedModel::interface_response_overlay(), NestedModel::iterator_response_overlay(), Model::modelPCIter, NestedModel::nestedResponseMap, IteratorScheduler::numIteratorJobs, NestedModel::optInterfaceIdMap, NestedModel::optInterfacePointer, NestedModel::optionalInterface, ParallelLibrary::parallel_configuration_iterator(), Model::parallelLib, IteratorScheduler::schedule_iterators(), NestedModel::subIterator, NestedModel::subIteratorIdMap, NestedModel::subIteratorJobCntr, NestedModel::subIteratorPRPQueue, NestedModel::subIteratorSched, and Interface::synch().

short local_eval_synchronization ( ) [inline], [protected], [virtual]

return optionalInterface synchronization setting

Used in setting Model::asynchEvalFlag. subModel synchronization is used for setting asynchEvalFlag within subModel.

Reimplemented from Model.

References Interface::asynch_local_evaluation_concurrency(), Interface::interface_synchronization(), NestedModel::optInterfacePointer, and NestedModel::optionalInterface.

int local_eval_concurrency ( ) [inline], [protected], [virtual]

return optionalInterface asynchronous evaluation concurrency

Used in setting Model::evaluationCapacity. subModel concurrency is used for setting evaluationCapacity within subModel.

Reimplemented from Model.

References Interface::asynch_local_evaluation_concurrency(), NestedModel::optInterfacePointer, and NestedModel::optionalInterface.

bool derived_master_overload ( ) const [inline], [protected], [virtual]

flag which prevents overloading the master with a multiprocessor evaluation (forwarded to optionalInterface)

Derived master overload for subModel is handled separately in subModel.compute_response() within subIterator.run().

Reimplemented from Model.

References Iterator::is_null(), Interface::iterator_eval_dedicated_master(), IteratorScheduler::iteratorScheduling, Interface::multi_proc_eval(), NestedModel::optInterfacePointer, NestedModel::optionalInterface, IteratorScheduler::procsPerIterator, NestedModel::subIterator, and NestedModel::subIteratorSched.

void derived_init_communicators (ParLevLIter *pl_iter,* int *max_eval_concurrency,* bool *recurse_flag* = *true*) [protected], [virtual]

set up optionalInterface and subModel for parallel operations

Asynchronous flags need to be initialized for the subModel. In addition, max_eval_concurrency is the outer level iterator concurrency, not the subIterator concurrency that subModel will see, and recomputing the message-lengths on the subModel is probably not a bad idea either. Therefore, recompute everything on subModel using init_communicators().

Reimplemented from Model.

References Response::active_set(), Model::currentVariables, ProblemDescDB::get_db_method_node(), ProblemDescDB::get_db_model_node(), Interface::init_communicators(), IteratorScheduler::init_evaluation_concurrency(), IteratorScheduler::init_iterator(), IteratorScheduler::init_iterator_parallelism(), Iterator::is_null(), IteratorScheduler::iterator_message_lengths(), IteratorScheduler::iteratorServerId, ProblemDescDB::max_procs_per_mi(), Model::messageLengths, IteratorScheduler::messagePass, ProblemDescDB::min_procs_per_mi(), Model::modelPCIter, IteratorScheduler::numIteratorServers, NestedModel::optInterfacePointer, NestedModel::optionalInterface, ParallelLibrary::parallel_configuration_iterator(), Model::parallelLib, Model::probDescDB, MPIPackBuffer::reset(), Iterator::response_results(), ProblemDescDB::set_db_list_nodes(), ProblemDescDB::set_db_method_node(), ProblemDescDB::set_db_model_nodes(), MPIPackBuffer::size(), NestedModel::subIterator, NestedModel::subIteratorSched, NestedModel::subMethodPointer, NestedModel::subModel, IteratorScheduler::update(), and NestedModel::update_sub_iterator().

int evaluation_id ( ) const [inline], [protected], [virtual]

Return the current evaluation id for the NestedModel.

return the top level nested evaluation count. To get the lower level eval count, the subModel must be explicitly queried. This is consistent with the eval counter definitions in surrogate models.

Reimplemented from Model.

References NestedModel::nestedModelEvalCntr.

size_t cv_index_map (size_t *cv_index,* const Variables & *vars*) [private]

offset cv_index to create index into aggregated primary/secondary arrays

maps index within active continuous variables to index within aggregated active continuous/discrete-int/discrete-string/discrete-real variables.

References SharedVariablesData::aleatory_uncertain_counts(), SharedVariablesData::design_counts(), SharedVariablesData::epistemic_uncertain_counts(), Variables::shared_data(), and SharedVariablesData::view().

Referenced by NestedModel::NestedModel(), and NestedModel::update_sub_model().

size_t div_index_map (size_t *div_index,* const Variables & *vars*) [private]

offset div_index to create index into aggregated primary/secondary arrays

maps index within active discrete int variables to index within aggregated active continuous/discrete-int/discrete-string/discrete-real variables.

References SharedVariablesData::aleatory_uncertain_counts(), Variables::cv(), SharedVariablesData::design_counts(), SharedVariablesData::epistemic_uncertain_counts(), Variables::shared_data(), SharedVariablesData::state_counts(), and SharedVariablesData::view().

Referenced by NestedModel::NestedModel(), and NestedModel::update_sub_model().

size_t dsv_index_map (size_t *dsv_index,* const Variables & *vars*) [private]

offset dsv_index to create index into aggregated primary/secondary arrays

maps index within active discrete string variables to index within aggregated active continuous/discrete-int/discrete-string/discrete-string variables.

References SharedVariablesData::aleatory_uncertain_counts(), Variables::cv(), SharedVariablesData::design_counts(), Variables::div(), SharedVariablesData::epistemic_uncertain_counts(), Variables::shared_data(), SharedVariablesData::state_counts(), and SharedVariablesData::view().

Referenced by NestedModel::NestedModel(), and NestedModel::update_sub_model().

size_t drv_index_map (size_t *drv_index,* const Variables & *vars*) [private]

offset drv_index to create index into aggregated primary/secondary arrays

maps index within active discrete real variables to index within aggregated active continuous/discrete-int/discrete-string/discrete-real variables.

References SharedVariablesData::aleatory_uncertain_counts(), Variables::cv(), SharedVariablesData::design_counts(), Variables::div(), Variables::dsv(), SharedVariablesData::epistemic_uncertain_counts(), Variables::shared_data(), SharedVariablesData::state_counts(), and SharedVariablesData::view().

Referenced by NestedModel::NestedModel(), and NestedModel::update_sub_model().

size_t ccv_index_map (size_t *ccv_index,* const Variables & *vars*) [private]

offset active complement ccv_index to create index into all continuous arrays

maps index within complement of active continuous variables to index within all continuous variables.

References Dakota::abort_handler(), SharedVariablesData::aleatory_uncertain_counts(), SharedVariablesData::design_counts(), SharedVariablesData::epistemic_uncertain_counts(), Variables::shared_data(), and SharedVariablesData::view().

Referenced by NestedModel::NestedModel(), and NestedModel::update_sub_model().

size_t cdiv_index_map (size_t *cdiv_index,* const Variables & *vars*) [private]

offset active complement cdiv_index to create index into all discrete int arrays

maps index within complement of active discrete int variables to index within all discrete int variables.

References Dakota::abort_handler(), SharedVariablesData::aleatory_uncertain_counts(), SharedVariablesData::design_counts(), SharedVariablesData::epistemic_uncertain_counts(), Variables::shared_data(), and SharedVariablesData::view().

Referenced by NestedModel::NestedModel(), and NestedModel::update_sub_model().

size_t cdsv_index_map (size_t *cdsv_index,* const Variables & *vars*) [private]

offset active complement cdsv_index to create index into all discrete string arrays

maps index within complement of active discrete string variables to index within all discrete string variables.

References Dakota::abort_handler(), SharedVariablesData::aleatory_uncertain_counts(), SharedVariablesData::design_counts(), SharedVariablesData::epistemic_uncertain_counts(), Variables::shared_data(), and SharedVariablesData::view().

Referenced by NestedModel::NestedModel(), and NestedModel::update_sub_model().

size_t cdrv_index_map (size_t *cdrv_index*, const Variables & *vars*) [private]

offset active complement cdrv_index to create index into all discrete real arrays

maps index within complement of active discrete real variables to index within all discrete real variables.

References Dakota::abort_handler(), SharedVariablesData::aleatory_uncertain_counts(), SharedVariablesData::design_counts(), SharedVariablesData::epistemic_uncertain_counts(), Variables::shared_data(), and SharedVariablesData::view().

Referenced by NestedModel::NestedModel(), and NestedModel::update_sub_model().

void response_mapping (const Response & *opt_interface_response*, const Response & *sub_iterator_response*, Response & *mapped_response*) [inline], [private]

combine the response from the optional interface evaluation with the response from the sub-iteration using the primaryCoeffs/secondaryCoeffs mappings to create the total response for the model

```
In the OUU case,

optionalInterface fns = {f}, {g} (deterministic primary functions, constraints)
subIterator fns       = {S}      (UQ response statistics)

Problem formulation for mapped functions:
                 minimize    {f} + [W]{S}
                 subject to  {g_l} <= {g}    <= {g_u}
                             {a_l} <= [A]{S} <= {a_u}
                             {g}    == {g_t}
                             [A]{S} == {a_t}
```

where [W] is the primary_mapping_matrix user input (primaryRespCoeffs class attribute), [A] is the secondary_mapping_matrix user input (secondaryRespCoeffs class attribute), {{g_l},{a_l}} are the top level inequality constraint lower bounds, {{g_u},{a_u}} are the top level inequality constraint upper bounds, and {{g_t},{a_t}} are the top level equality constraint targets.

NOTE: optionalInterface/subIterator primary fns (obj/lsq/generic fns) overlap but optionalInterface/subIterator secondary fns (ineq/eq constraints) do not. The [W] matrix can be specified so as to allow

- some purely deterministic primary functions and some combined: [W] filled and [W].num_rows() < {f}.length() [combined first] *or* [W].num_rows() == {f}.length() and [W] contains rows of zeros [combined last]
- some combined and some purely stochastic primary functions: [W] filled and [W].num_rows() > {f}.length()
- separate deterministic and stochastic primary functions: [W].num_rows() > {f}.length() and [W] contains {f}.length() rows of zeros.

If the need arises, could change constraint definition to allow overlap as well: {g_l} <= {g} + [A]{S} <= {g_u} with [A] usage the same as for [W] above.

In the UOO case, things are simpler, just compute statistics of each optimization response function: [W] = [I], {f}/{g}/[A] are empty.

References Response::active_set_request_vector(), NestedModel::check_response_map(), NestedModel::interface_response_overlay(), and NestedModel::iterator_response_overlay().

13.70.3 Member Data Documentation

Model subModel [private]

the sub-model used in sub-iterator evaluations

There are no restrictions on subModel, so arbitrary nestings are possible. This is commonly used to support surrogate-based optimization under uncertainty by having NestedModels contain SurrogateModels and vice versa.

Referenced by NestedModel::component_parallel_mode(), NestedModel::derived_init_communicators(), NestedModel::derived_init_serial(), NestedModel::derived_subordinate_models(), NestedModel::fine_grained_evaluation_counters(), NestedModel::integer_variable_mapping(), NestedModel::NestedModel(), NestedModel::print_evaluation_summary(), NestedModel::real_variable_mapping(), NestedModel::resolve_integer_variable_mapping(), NestedModel::resolve_real_variable_mapping(), NestedModel::resolve_string_variable_mapping(), NestedModel::serve_run(), NestedModel::set_mapping(), NestedModel::sm_acv_index_map(), NestedModel::sm_adiv_index_map(), NestedModel::string_variable_mapping(), NestedModel::subordinate_model(), NestedModel::surrogate_response_mode(), NestedModel::update_inactive_view(), and NestedModel::update_sub_model().

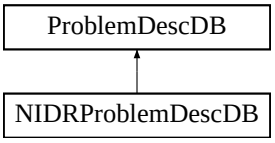
The documentation for this class was generated from the following files:

- NestedModel.hpp
- NestedModel.cpp

13.71 NIDRProblemDescDB Class Reference

The derived input file database utilizing the new IDR parser.

Inheritance diagram for NIDRProblemDescDB:



Public Member Functions

- NIDRProblemDescDB (ParallelLibrary &parallel_lib)

  constructor
- ~NIDRProblemDescDB ()

  destructor
- void derived_parse_inputs (const ProgramOptions &prog_opts)

  parses the input file and populates the problem description database using NIDR.
- void derived_broadcast ()

  perform any data processing that must be coordinated with DB buffer broadcasting (performed prior to broadcasting the DB buffer on rank 0 and after receiving the DB buffer on other processor ranks)
- void derived_post_process ()

  perform any additional data post-processing
- **KWH** (iface_Real)
- **KWH** (iface_Rlit)
- **KWH** (iface_false)
- **KWH** (iface_ilit)
- **KWH** (iface_pint)
- **KWH** (iface_lit)
- **KWH** (iface_start)

- **KWH** (iface_stop)
- **KWH** (iface_str)
- **KWH** (iface_str2D)
- **KWH** (iface_strL)
- **KWH** (iface_true)
- **KWH** (iface_type)
- **KWH** (method_Ii)
- **KWH** (method_Real)
- **KWH** (method_Real01)
- **KWH** (method_RealDL)
- **KWH** (method_RealLlit)
- **KWH** (method_Realp)
- **KWH** (method_Realz)
- **KWH** (method_Ri)
- **KWH** (method_false)
- **KWH** (method_szarray)
- **KWH** (method_ilit2)
- **KWH** (method_ilit2p)
- **KWH** (method_int)
- **KWH** (method_ivec)
- **KWH** (method_lit)
- **KWH** (method_litc)
- **KWH** (method_liti)
- **KWH** (method_litp)
- **KWH** (method_litr)
- **KWH** (method_litz)
- **KWH** (method_nnint)
- **KWH** (method_num_resplevs)
- **KWH** (method_piecewise)
- **KWH** (method_pint)
- **KWH** (method_pintz)
- **KWH** (method_resplevs)
- **KWH** (method_resplevs01)
- **KWH** (method_shint)
- **KWH** (method_sizet)
- **KWH** (method_slit2)
- **KWH** (method_start)
- **KWH** (method_stop)
- **KWH** (method_str)
- **KWH** (method_strL)
- **KWH** (method_true)
- **KWH** (method_tr_final)
- **KWH** (method_type)
- **KWH** (method_usharray)
- **KWH** (method_ushint)
- **KWH** (method_utype)

- **KWH** (method_utype_lit)
- **KWH** (model_Real)
- **KWH** (model_RealDL)
- **KWH** (model_false)
- **KWH** (model_int)
- **KWH** (model_intsetm1)
- **KWH** (model_lit)
- **KWH** (model_order)
- **KWH** (model_pint)
- **KWH** (model_shint)
- **KWH** (model_start)
- **KWH** (model_stop)
- **KWH** (model_str)
- **KWH** (model_strL)
- **KWH** (model_true)
- **KWH** (model_type)
- **KWH** (resp_RealDL)
- **KWH** (resp_RealL)
- **KWH** (resp_false)
- **KWH** (resp_intset)
- **KWH** (resp_ivec)
- **KWH** (resp_lit)
- **KWH** (resp_sizet)
- **KWH** (resp_start)
- **KWH** (resp_stop)
- **KWH** (resp_str)
- **KWH** (resp_strL)
- **KWH** (resp_true)
- **KWH** (env_int)
- **KWH** (env_start)
- **KWH** (env_str)
- **KWH** (env_true)
- **KWH** (var_RealLb)
- **KWH** (var_RealUb)
- **KWH** (var_IntLb)
- **KWH** (var_categorical)
- **KWH** (var_caulbl)
- **KWH** (var_dauilbl)
- **KWH** (var_dauslbl)
- **KWH** (var_daurlbl)
- **KWH** (var_ceulbl)
- **KWH** (var_deuilbl)
- **KWH** (var_deuslbl)
- **KWH** (var_deurlbl)
- **KWH** (var_pintz)
- **KWH** (var_start)

- **KWH** (var_stop)
- **KWH** (var_str)
- **KWH** (var_strL)
- **KWH** (var_true)
- **KWH** (var_newiarray)
- **KWH** (var_newsarray)
- **KWH** (var_newivec)
- **KWH** (var_newrvec)
- **KWH** (var_ivec)
- **KWH** (var_svec)
- **KWH** (var_rvec)
- **KWH** (var_type)

Static Public Member Functions

- static void botch (const char ∗fmt,...)

  print and error message and immediately abort
- static void check_variables (std::list< DataVariables > ∗)

  check each node in a list of DataVariables, first mapping DataVariables members back to flat NIDR arrays if needed.
- static void **check_responses** (std::list< DataResponses > ∗)
- static void make_variable_defaults (std::list< DataVariables > ∗)

  Bounds and initial point check and inferred bounds generation.
- static void **make_response_defaults** (std::list< DataResponses > ∗)
- static void squawk (const char ∗fmt,...)

  print an error message and increment nerr, but continue
- static void warn (const char ∗fmt,...)

  print a warning

Static Public Attributes

- static NIDRProblemDescDB ∗ pDDBInstance

  Pointer to the active object instance used within the static kwhandler functions in order to avoid the need for static data. Only initialized when parsing an input file; will be NULL for cases of direct DB population only.
- static int nerr = 0

  number of parse error encountered

Static Private Member Functions

- static void check_variables_node (void ∗v)

  check a single variables node; input argument v is Var_Info∗
- static int check_driver (const String &an_driver, const StringArray &link_files, const StringArray &copy_-files)

  tokenize and try to validate the presence of an analysis driver, potentially included in the linked or copied template files

Private Attributes

- std::list< void * > VIL

  List of Var_Info pointers, one per Variables instance.

Additional Inherited Members

13.71.1 Detailed Description

The derived input file database utilizing the new IDR parser.

The NIDRProblemDescDB class is derived from ProblemDescDB for use by the NIDR parser in processing DAKOTA input file data. For information on modifying the NIDR input parsing procedures, refer to Dakota/docs/Dev_Spec_Change.dox. For more on the parsing technology, see "Specifying and Reading Program Input with NIDR" by David M. Gay (report SAND2008-2261P, which is available in PDF form as http://dakota.sandia.gov/papers/nidr08.pdf). Source for the routines declared herein is NIDRProblemDescDB.cpp, in which most routines are so short that a description seems unnecessary.

13.71.2 Member Function Documentation

void derived_parse_inputs (const ProgramOptions & *prog_opts*) [virtual]

parses the input file and populates the problem description database using NIDR.

Parse the input file using the Input Deck Reader (IDR) parsing system. IDR populates the IDRProblemDescDB object with the input file data.

Reimplemented from ProblemDescDB.

References Dakota::abort_handler(), NIDRProblemDescDB::botch(), ProblemDescDB::dataMethodList, DataMethodRep::dlDetails, DataMethodRep::dlLib, ProgramOptions::input_file(), ProgramOptions::input_string(), NIDRProblemDescDB::nerr, ProblemDescDB::parallel_library(), ProgramOptions::parser_options(), NIDRProblemDescDB::pDDBInstance, and NIDRProblemDescDB::squawk().

int check_driver (const String & *an_driver*, const StringArray & *link_files*, const StringArray & *copy_files*) [static], [private]

tokenize and try to validate the presence of an analysis driver, potentially included in the linked or copied template files

returns 1 if not found, 2 if found, but not executable, 0 if found (no error) in case we want to return to error on not found...

References WorkdirHelper::find_driver(), NIDRProblemDescDB::squawk(), WorkdirHelper::tokenize_driver(), NIDRProblemDescDB::warn(), and WorkdirHelper::which().

void make_variable_defaults (std::list< DataVariables > * *dvl*) [static]

Bounds and initial point check and inferred bounds generation.

Size arrays for contiguous storage of aggregated uncertain types. For each variable type, call Vgen_* to generate inferred bounds and initial point, repairing initial if needed. size the aggregate arrays for uncertain (design and state are stored separately

References Dakota::DesignAndStateLabelsCheck, NIDRProblemDescDB::squawk(), Dakota::var_mp_drange, Dakota::VLUncertainInt, Dakota::VLUncertainReal, and Dakota::VLUncertainStr.

Referenced by NIDRProblemDescDB::derived_post_process().

The documentation for this class was generated from the following files:

- NIDRProblemDescDB.hpp
- NIDRProblemDescDB.cpp

13.72 NL2Res Struct Reference

Auxiliary information passed to calcr and calcj via ur.

Public Attributes

- Real * r

  residual r = r(x)
- Real * J

  Jacobian J = J(x)
- Real * x

  corresponding parameter vector
- int nf

  function invocation count for r(x)

13.72.1 Detailed Description

Auxiliary information passed to calcr and calcj via ur.

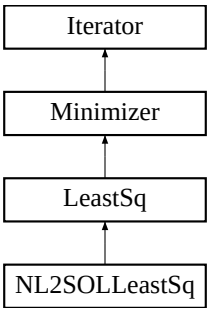
The documentation for this struct was generated from the following file:

- NL2SOLLeastSq.cpp

13.73 NL2SOLLeastSq Class Reference

Wrapper class for the NL2SOL nonlinear least squares library.

Inheritance diagram for NL2SOLLeastSq:



Public Member Functions

- NL2SOLLeastSq (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NL2SOLLeastSq (Model &model)

alternate constructor

- ~NL2SOLLeastSq ()

  destructor

- void minimize_residuals ()

  Used within the least squares branch for minimizing the sum of squares residuals. Redefines the run virtual function for the least squares branch.

Static Private Member Functions

- static void calcr (int *np, int *pp, Real *x, int *nfp, Real *r, int *ui, void *ur, Vf vf)

  evaluator function for residual vector

- static void calcj (int *np, int *pp, Real *x, int *nfp, Real *J, int *ui, void *ur, Vf vf)

  evaluator function for residual Jacobian

Private Attributes

- int auxprt

  `auxilary printing bits (see Dakota Ref Manual): sum of`
  < 1 = x0prt (print initial guess) < 2 = solprt (print final solution) < 4 = statpr (print solution statistics) < 8 = parprt (print nondefault parameters) < 16 = dradpr (print bound constraint drops/adds) < debug/verbose/normal use default = 31 (everything), < quiet uses 3, silent uses 0.

- int outlev

  `frequency of output summary lines in number of iterations`
  < (debug/verbose/normal/quiet use default = 1, silent uses 0)

- Real dltfdj

  `finite-diff step size for computing Jacobian approximation`
  < (`fd_gradient_step_size`)

- Real delta0

  `finite-diff step size for gradient differences for H`
  < (a component of some covariance approximations, if desired) < (`fd_hessian_step_size`)

- Real dltfdc

  `finite-diff step size for function differences for H`
  < (`fd_hessian_step_size`)

- int mxfcal

  function-evaluation limit (`max_function_evaluations`)

- int mxiter

  iteration limit (`max_iterations`)

- Real rfctol

  relative fn convergence tolerance (`convergence_tolerance`)

- Real afctol

  absolute fn convergence tolerance (`absolute_conv_tol`)

- Real xctol

  x-convergence tolerance (`x_conv_tol`)

- Real sctol

  singular convergence tolerance (`singular_conv_tol`)

- Real lmaxs

radius for singular-convergence test (`singular_radius`)

- Real xftol

  false-convergence tolerance (`false_conv_tol`)

- int covreq

  `kind of covariance required (\c covariance):`
  < 1 or -1 ==> sigma^2 H^-1 J^T J H^-1 < 2 or -2 ==> sigma^2 H^-1 < 3 or -3 ==> sigma^2 (J^T J)^-1 < 1 or 2 ==> use gradient diffs to estimate H < -1 or -2 ==> use function diffs to estimate H < default = 0 (no covariance)

- int rdreq

  `whether to compute the regression diagnostic vector`
  < (`regression_diagnostics`)

- Real fprec

  expected response function precision (`function_precision`)

- Real lmax0

  initial trust-region radius (`initial_trust_radius`)

Static Private Attributes

- static NL2SOLLeastSq * nl2solInstance

  pointer to the active object instance used within the static evaluator functions

Additional Inherited Members

13.73.1 Detailed Description

Wrapper class for the NL2SOL nonlinear least squares library.

The NL2SOLLeastSq class provides a wrapper for NL2SOL (TOMS Algorithm 573), in the updated form of Port Library routines dn[fg][b] from Bell Labs; see http://www.netlib.org/port/readme. The Fortran from Port has been turned into C by f2c. NL2SOL uses a function pointer approach for which passed functions must be either global functions or static member functions.

13.73.2 Member Function Documentation

void minimize_residuals ( ) [virtual]

Used within the least squares branch for minimizing the sum of squares residuals. Redefines the run virtual function for the least squares branch.

Details on the following subscript values appear in "Usage Summary for Selected Optimization Routines" by David M. Gay, Computing Science Technical Report No. 153, AT&T Bell Laboratories, 1990. http://netlib.bell-labs.com/cm/cs/cstr/153.ps.gz

Implements LeastSq.

References NL2SOLLeastSq::afctol, NL2SOLLeastSq::auxprt, Iterator::bestResponseArray, Iterator::bestVariablesArray, Minimizer::boundConstraintFlag, NL2SOLLeastSq::calcj(), NL2SOLLeastSq::calcr(), Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), Dakota::copy_data(), NL2SOLLeastSq::covreq, NL2SOLLeastSq::delta0, NL2SOLLeastSq::dltfdc, NL2SOLLeastSq::dltfdj, NL2SOLLeastSq::fprec, LeastSq::get_confidence_intervals(), Model::gradient_type(), Iterator::iteratedModel, NL2SOLLeastSq::lmax0, NL2SOLLeastSq::lmaxs, NL2SOLLeastSq::mxfcal, NL2SOLLeastSq::mxiter, NL2SOLLeastSq::nl2solInstance, Minimizer::numContinuousVars, LeastSq::numLeastSqTerms, NL2SOLLeastSq::outlev, NL2SOLLeastSq::rdreq, NL2SOLLeastSq::rfctol, NL2SOLLeastSq::sctol, Minimizer::speculativeFlag, Minimizer::vendorNumericalGradFlag, NL2SOLLeastSq::xctol, and NL2SOLLeastSq::xftol.

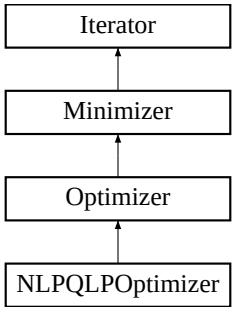
The documentation for this class was generated from the following files:

- NL2SOLLeastSq.hpp
- NL2SOLLeastSq.cpp

13.74 NLPQLPOptimizer Class Reference

Wrapper class for the NLPQLP optimization library, Version 2.0.

Inheritance diagram for NLPQLPOptimizer:



Public Member Functions

- NLPQLPOptimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NLPQLPOptimizer (Model &model)

  alternate constructor
- ~NLPQLPOptimizer ()

  destructor
- void find_optimum ()

  Used within the optimizer branch for computing the optimal solution. Redefines the run virtual function for the optimizer branch.

Protected Member Functions

- void initialize_run ()

  performs run-time set up

Private Member Functions

- void initialize ()

  Shared constructor code.
- void allocate_workspace ()

  Allocates workspace for the optimizer.
- void deallocate_workspace ()

  Releases workspace memory.
- void allocate_constraints ()

  Allocates constraint mappings.

Private Attributes

- int L

  L : Number of parallel systems, i.e. function calls during line search at predetermined iterates. HINT: If only less than 10 parallel function evaluations are possible, it is recommended to apply the serial version by setting L=1.

- int numEqConstraints

  numEqConstraints : Number of equality constraints.

- int MMAX

  MMAX : Row dimension of array DG containing Jacobian of constraints. MMAX must be at least one and greater or equal to M.

- int N

  N : Number of optimization variables.

- int NMAX

  NMAX : Row dimension of C. NMAX must be at least two and greater than N.

- int MNN2

  MNN2 : Must be equal to M+N+N+2.

- double ∗ X

  X(NMAX,L) : Initially, the first column of X has to contain starting values for the optimal solution. On return, X is replaced by the current iterate. In the driving program the row dimension of X has to be equal to NMAX. X is used internally to store L different arguments for which function values should be computed simultaneously.

- double ∗ F

  F(L) : On return, F(1) contains the final objective function value. F is used also to store L different objective function values to be computed from L iterates stored in X.

- double ∗ G

  G(MMAX,L) : On return, the first column of G contains the constraint function values at the final iterate X. In the driving program the row dimension of G has to be equal to MMAX. G is used internally to store L different set of constraint function values to be computed from L iterates stored in X.

- double ∗ DF

  DF(NMAX) : DF contains the current gradient of the objective function. In case of numerical differentiation and a distributed system (L>1), it is recommended to apply parallel evaluations of F to compute DF.

- double ∗ DG

  DG(MMAX,NMAX) : DG contains the gradients of the active constraints (ACTIVE(J)=.true.) at a current iterate X. The remaining rows are filled with previously computed gradients. In the driving program the row dimension of DG has to be equal to MMAX.

- double ∗ U

  U(MNN2) : U contains the multipliers with respect to the actual iterate stored in the first column of X. The first M locations contain the multipliers of the M nonlinear constraints, the subsequent N locations the multipliers of the lower bounds, and the final N locations the multipliers of the upper bounds. At an optimal solution, all multipliers with respect to inequality constraints should be nonnegative.

- double ∗ C

  C(NMAX,NMAX) : On return, C contains the last computed approximation of the Hessian matrix of the Lagrangian function stored in form of an LDL decomposition. C contains the lower triangular factor of an LDL factorization of the final quasi-Newton matrix (without diagonal elements, which are always one). In the driving program, the row dimension of C has to be equal to NMAX.

- double ∗ D

  D(NMAX) : The elements of the diagonal matrix of the LDL decomposition of the quasi-Newton matrix are stored in the one-dimensional array D.

- double ACC

  ACC : The user has to specify the desired final accuracy (e.g. 1.0D-7). The termination accuracy should not be smaller than the accuracy by which gradients are computed.

- double ACCQP

  ACCQP : The tolerance is needed for the QP solver to perform several tests, for example whether optimality conditions are satisfied or whether a number is considered as zero or not. If ACCQP is less or equal to zero, then the machine precision is computed by NLPQLP and subsequently multiplied by 1.0D+4.

- double STPMIN

  STPMIN : Minimum steplength in case of L>1. Recommended is any value in the order of the accuracy by which functions are computed. The value is needed to compute a steplength reduction factor by STPMIN∗∗(1/L-1). If STPMIN<=0, then STPMIN=ACC is used.

- int MAXFUN

  MAXFUN : The integer variable defines an upper bound for the number of function calls during the line search (e.g. 20). MAXFUN is only needed in case of L=1, and must not be greater than 50.

- int MAXIT

  MAXIT : Maximum number of outer iterations, where one iteration corresponds to one formulation and solution of the quadratic programming subproblem, or, alternatively, one evaluation of gradients (e.g. 100).

- int MAX_NM

  MAX_NM : Stack size for storing merit function values at previous iterations for non-monotone line search (e.g. 10). In case of MAX_NM=0, monotone line search is performed.

- double TOL_NM

  TOL_NM : Relative bound for increase of merit function value, if line search is not successful during the very first step. Must be non-negative (e.g. 0.1).

- int IPRINT

  IPRINT : Specification of the desired output level. IPRINT = 0 : No output of the program. IPRINT = 1 : Only a final convergence analysis is given. IPRINT = 2 : One line of intermediate results is printed in each iteration. IPRINT = 3 : More detailed information is printed in each iteration step, e.g. variable, constraint and multiplier values. IPRINT = 4 : In addition to 'IPRINT=3', merit function and steplength values are displayed during the line search.

- int MODE

  MODE : The parameter specifies the desired version of NLPQLP. MODE = 0 : Normal execution (reverse communication!). MODE = 1 : The user wants to provide an initial guess for the multipliers in U and for the Hessian of the Lagrangian function in C and D in form of an LDL decomposition.

- int IOUT

  IOUT : Integer indicating the desired output unit number, i.e. all write-statements start with 'WRITE(IOUT,... '.

- int IFAIL

  IFAIL : The parameter shows the reason for terminating a solution process. Initially IFAIL must be set to zero. On return IFAIL could contain the following values: IFAIL =-2 : Compute gradient values w.r.t. the variables stored in first column of X, and store them in DF and DG. Only derivatives for active constraints ACTIVE(J)=.TRUE. need to be computed. Then call NLPQLP again, see below. IFAIL =-1 : Compute objective fn and all constraint values subject the variables found in the first L columns of X, and store them in F and G. Then call NLPQLP again, see below. IFAIL = 0 : The optimality conditions are satisfied. IFAIL = 1 : The algorithm has been stopped after MAXIT iterations. IFAIL = 2 : The algorithm computed an uphill search direction. IFAIL = 3 : Underflow occurred when determining a new approxi- mation matrix for the Hessian of the Lagrangian. IFAIL = 4 : The line search could not be terminated successfully. IFAIL = 5 : Length of a working array is too short. More detailed error information is obtained with 'IPRINT>0'. IFAIL = 6 : There are false dimensions, for example M>MMAX, N>=NMAX, or MNN2<>M+N+N+2. IFAIL = 7 : The search direction is close to zero, but the current iterate is still infeasible. IFAIL = 8 : The starting point violates a lower or upper bound. IFAIL = 9 : Wrong

input parameter, i.e., MODE, LDL decomposition in D and C (in case of MODE=1), IPRINT, IOUT IFAIL = 10 : Internal inconsistency of the quadratic subproblem, division by zero. IFAIL > 100 : The solution of the quadratic programming subproblem has been terminated with an error message and IFAIL is set to IFQL+100, where IFQL denotes the index of an inconsistent constraint.

- double ∗ WA

  WA(LWA) : WA is a real working array of length LWA.

- int LWA

  LWA : LWA value extracted from NLPQLP20.f.

- int ∗ KWA

  KWA(LKWA) : The user has to provide working space for an integer array.

- int LKWA

  LKWA : LKWA should be at least N+10.

- int ∗ ACTIVE

  ACTIVE(LACTIV) : The logical array shows a user the constraints, which NLPQLP considers to be active at the last computed iterate, i.e. G(J,X) is active, if and only if ACTIVE(J)=.TRUE., J=1,...,M.

- int LACTIVE

  LACTIV : The length LACTIV of the logical array should be at least 2∗M+10.

- int LQL

  LQL : If LQL = .TRUE., the quadratic programming subproblem is to be solved with a full positive definite quasi-Newton matrix. Otherwise, a Cholesky decomposition is performed and updated, so that the subproblem matrix contains only an upper triangular factor.

- int numNlpqlConstr

  total number of constraints seen by NLPQL

- SizetList nonlinIneqConMappingIndices

  a list of indices for referencing the DAKOTA nonlinear inequality constraints used in computing the corresponding NLPQL constraints.

- RealList nonlinIneqConMappingMultipliers

  a list of multipliers for mapping the DAKOTA nonlinear inequality constraints to the corresponding NLPQL constraints.

- RealList nonlinIneqConMappingOffsets

  a list of offsets for mapping the DAKOTA nonlinear inequality constraints to the corresponding NLPQL constraints.

- SizetList linIneqConMappingIndices

  a list of indices for referencing the DAKOTA linear inequality constraints used in computing the corresponding NLPQL constraints.

- RealList linIneqConMappingMultipliers

  a list of multipliers for mapping the DAKOTA linear inequality constraints to the corresponding NLPQL constraints.

- RealList linIneqConMappingOffsets

  a list of offsets for mapping the DAKOTA linear inequality constraints to the corresponding NLPQL constraints.

Additional Inherited Members

13.74.1 Detailed Description

Wrapper class for the NLPQLP optimization library, Version 2.0.

```
AN IMPLEMENTATION OF A SEQUENTIAL QUADRATIC PROGRAMMING
 METHOD FOR SOLVING NONLINEAR OPTIMIZATION PROBLEMS BY
  DISTRIBUTED COMPUTING AND NON-MONOTONE LINE SEARCH
```

This subroutine solves the general nonlinear programming problem

```
minimize     F(X)
subject to   G(J,X)   =  0  ,  J=1,...,ME
             G(J,X)  >=  0  ,  J=ME+1,...,M
             XL  <=  X  <=  XU
```

and is an extension of the code NLPQLD. NLPQLP is specifically tuned to run under distributed systems. A new input parameter L is introduced for the number of parallel computers, that is the number of function calls to be executed simultaneously. In case of L=1, NLPQLP is identical to NLPQLD. Otherwise the line search is modified to allow L parallel function calls in advance. Moreover the user has the opportunity to used distributed function calls for evaluating gradients.

The algorithm is a modification of the method of Wilson, Han, and Powell. In each iteration step, a linearly constrained quadratic programming problem is formulated by approximating the Lagrangian function quadratically and by linearizing the constraints. Subsequently, a one-dimensional line search is performed with respect to an augmented Lagrangian merit function to obtain a new iterate. Also the modified line search algorithm guarantees convergence under the same assumptions as before.

For the new version, a non-monotone line search is implemented which allows to increase the merit function in case of instabilities, for example caused by round-off errors, errors in gradient approximations, etc.

The subroutine contains the option to predetermine initial guesses for the multipliers or the Hessian of the Lagrangian function and is called by reverse communication.

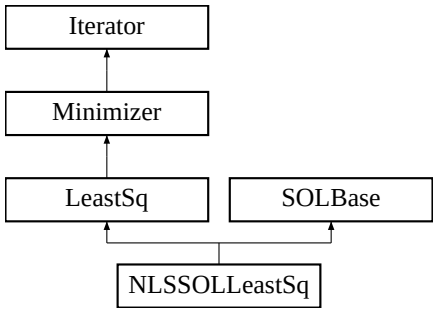
The documentation for this class was generated from the following files:

- NLPQLPOptimizer.hpp
- NLPQLPOptimizer.cpp

13.75 NLSSOLLeastSq Class Reference

Wrapper class for the NLSSOL nonlinear least squares library.

Inheritance diagram for NLSSOLLeastSq:



Public Member Functions

- NLSSOLLeastSq (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NLSSOLLeastSq (Model &model)

alternate constructor

- ~NLSSOLLeastSq ()

  destructor

- void minimize_residuals ()

  Used within the least squares branch for minimizing the sum of squares residuals. Redefines the run virtual function for the least squares branch.

Static Private Member Functions

- static void least_sq_eval (int &mode, int &m, int &n, int &nrowfj, double *x, double *f, double *gradf, int &nstate)

  Evaluator for NLSSOL: computes the values and first derivatives of the least squares terms (passed by function pointer to NLSSOL).

Static Private Attributes

- static NLSSOLLeastSq * nlssolInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.75.1 Detailed Description

Wrapper class for the NLSSOL nonlinear least squares library.

The NLSSOLLeastSq class provides a wrapper for NLSSOL, a Fortran 77 sequential quadratic programming library from Stanford University marketed by Stanford Business Associates. It uses a function pointer approach for which passed functions must be either global functions or static member functions. Any nonstatic attribute used within static member functions must be either local to that function or accessed through a static pointer.

The user input mappings are as follows: `max_function_evaluations` is implemented directly in NLSSOLLeastSq's evaluator functions since there is no NLSSOL parameter equivalent, and `max_iterations`, `convergence_tolerance`, `output` verbosity, `verify_level`, `function_precision`, and `linesearch_tolerance` are mapped into NLSSOL's "Major Iteration Limit", "Optimality Tolerance", "Major Print Level" (`verbose`: Major Print Level = 20; `quiet`: Major Print Level = 10), "Verify Level", "Function Precision", and "Linesearch Tolerance" parameters, respectively, using NLSSOL's npoptn() subroutine (as wrapped by npoptn2() from the npoptn_wrapper.f file). Refer to [Gill, P.E., Murray, W., Saunders, M.A., and Wright, M.H., 1986] for information on NLSSOL's optional input parameters and the npoptn() subroutine.

13.75.2 Constructor & Destructor Documentation

NLSSOLLeastSq (ProblemDescDB & *problem_db*, Model & *model*)

standard constructor

This is the primary constructor. It accepts a Model reference.

References Minimizer::constraintTol, Iterator::convergenceTol, Model::fd_gradient_step_size(), ProblemDescDB::get_int(), ProblemDescDB::get_real(), Model::gradient_type(), Iterator::iteratedModel, Iterator::maxIterations, Iterator::outputLevel, Iterator::probDescDB, SOLBase::set_options(), Minimizer::speculativeFlag, and Minimizer::vendorNumericalGradFlag.

NLSSOLLeastSq (Model & *model*)

alternate constructor

This is an alternate constructor which accepts a Model but does not have a supporting method specification from the ProblemDescDB.

References Minimizer::constraintTol, Iterator::convergenceTol, Model::fd_gradient_step_size(), Model::gradient_type(), Iterator::iteratedModel, Iterator::maxIterations, Iterator::outputLevel, SOLBase::set_options(), Minimizer::speculativeFlag, and Minimizer::vendorNumericalGradFlag.

The documentation for this class was generated from the following files:

- NLSSOLLeastSq.hpp
- NLSSOLLeastSq.cpp

13.76 NoDBBaseConstructor Struct Reference

Dummy struct for overloading constructors used in on-the-fly instantiations without ProblemDescDB support.

Public Member Functions

- NoDBBaseConstructor (int=0)

  C++ structs can have constructors.

13.76.1 Detailed Description

Dummy struct for overloading constructors used in on-the-fly instantiations without ProblemDescDB support.

NoDBBaseConstructor is used to overload the constructor used for on-the-fly instantiations in which ProblemDescDB queries cannot be used. Putting this struct here avoids circular dependencies.

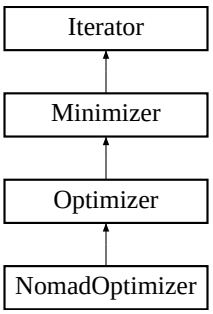
The documentation for this struct was generated from the following file:

- dakota_global_defs.hpp

13.77 NomadOptimizer Class Reference

Wrapper class for NOMAD Optimizer.

Inheritance diagram for NomadOptimizer:



Classes

- class Evaluator

  NOMAD-based Evaluator class.

Public Member Functions

- NomadOptimizer (ProblemDescDB &problem_db, Model &model)

  Constructor.
- NomadOptimizer (Model &model)

  alternate constructor for Iterator instantiations without DB
- ~NomadOptimizer ()

  Destructor.
- void find_optimum ()

  Calls the NOMAD solver.

Private Member Functions

- void load_parameters (Model &model)

  Convenience function for Parameter loading.

Private Attributes

- int numTotalVars

  Total across all types of variables.
- int numNomadNonlinearIneqConstraints

  Number of nonlinear inequality constraints after put into the format required by Nomad.
- int randomSeed

  Parameters passes to Nomad.
- int **maxBlackBoxEvals**
- int **maxIterations**
- std::string **outputFormat**
- std::string **historyFile**
- bool **displayAll**
- Real **epsilon**
- Real **vns**
- NOMAD::Point initialPoint

  Pointer to Nomad initial point.
- NOMAD::Point upperBound

  Pointer to Nomad upper bounds.
- NOMAD::Point lowerBound

  Pointer to Nomad lower bounds.
- std::vector< int > constraintMapIndices

  map from Dakota constraint number to Nomad constraint number
- std::vector< double > constraintMapMultipliers

  multipliers for constraint transformations
- std::vector< double > constraintMapOffsets

  offsets for constraint transformations

Additional Inherited Members

13.77.1 Detailed Description

Wrapper class for NOMAD Optimizer.

NOMAD (is a Nonlinear Optimization by Mesh Adaptive Direct search) is a simulation-based optimization package designed to efficiently explore a design space using Mesh Adaptive Search.

Mesh Adaptive Direct Search uses Meshes, discretizations of the domain space of variables. It generates multiple meshes, and as its name implies, it also adapts the refinement of the meshes in order to find the best solution of a problem.

The objective of each iteration is to find points in a mesh that improves the current solution. If a better solution is not found, the next iteration is done over a finer mesh.

Each iteration is composed of two steps: Search and Poll. The Search step finds any point in the mesh in an attempt to find an improvement; while the Poll step generates trial mesh points surrounding the current best current solution.

The NomadOptimizer is a wrapper for the NOMAD library. It features the following attributes: `max_function_evaluations`, `display_format`, `display_all_evaluations`, `function_precision`, `max_iterations`.

13.77.2 Constructor & Destructor Documentation

NomadOptimizer (ProblemDescDB & *problem_db,* Model & *model*)

Constructor.

```
NOMAD Optimizer Constructor
```

Parameters

model	DAKOTA Model object

References ProblemDescDB::get_bool(), ProblemDescDB::get_int(), ProblemDescDB::get_real(), ProblemDescDB::get_string(), NomadOptimizer::load_parameters(), Iterator::probDescDB, and NomadOptimizer::randomSeed.

13.77.3 Member Function Documentation

void load_parameters (Model & *model*) `[private]`

Convenience function for Parameter loading.

```
This function takes the Parameters provided
```

by the user in the DAKOTA model.

Parameters

model	NOMAD Model object

References Dakota::_NPOS, Dakota::abort_handler(), Minimizer::bigIntBoundSize, Minimizer::bigRealBoundSize, NomadOptimizer::constraintMapIndices, NomadOptimizer::constraintMapMultipliers, NomadOptimizer::constraintMapOffsets, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), Model::discrete_int_lower_bounds(), Model::discrete_int_sets(), Model::discrete_int_upper_bounds(),

Model::discrete_int_variables(), Model::discrete_real_lower_bounds(), Model::discrete_real_upper_bounds(), Model::discrete_real_variables(), Model::discrete_set_int_values(), Model::discrete_set_real_values(), NomadOptimizer::initialPoint, Iterator::iteratedModel, NomadOptimizer::lowerBound, Model::nonlinear_eq_constraint_targets(), Model::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Minimizer::numContinuousVars, Minimizer::numDiscreteIntVars, Minimizer::numDiscreteRealVars, NomadOptimizer::numNomadNonlinearIneqConstraints, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, NomadOptimizer::numTotalVars, Dakota::set_value_to_index(), and NomadOptimizer::upperBound.

Referenced by NomadOptimizer::NomadOptimizer().

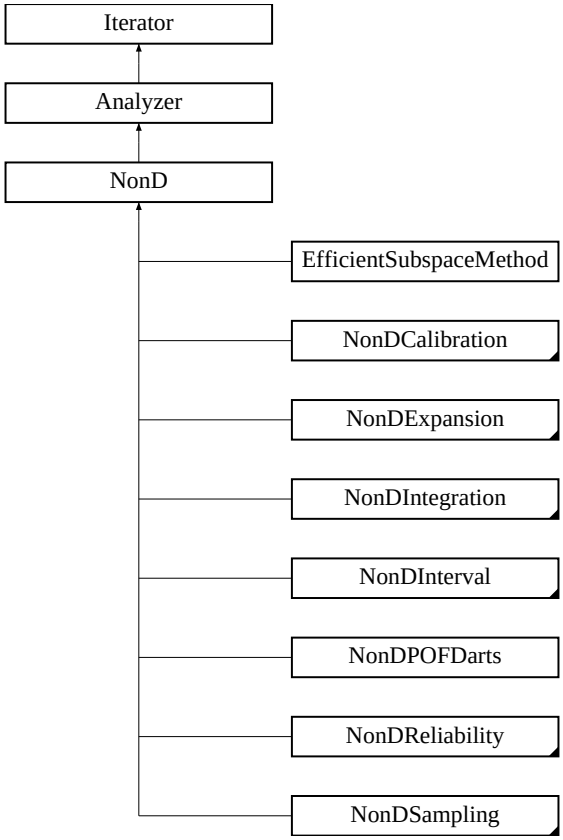
The documentation for this class was generated from the following files:

- NomadOptimizer.hpp
- NomadOptimizer.cpp

13.78 NonD Class Reference

Base class for all nondetermistic iterators (the DAKOTA/UQ branch).

Inheritance diagram for NonD:



Public Member Functions

- void initialize_random_variables (short u_space_type)

initialize natafTransform based on distribution data from iteratedModel

- void initialize_random_variables (const Pecos::ProbabilityTransformation &transform)

  alternate form: initialize natafTransform based on incoming data
- void requested_levels (const RealVectorArray &req_resp_levels, const RealVectorArray &req_prob_levels, const RealVectorArray &req_rel_levels, const RealVectorArray &req_gen_rel_levels, short resp_lev_tgt, short resp_lev_tgt_reduce, bool cdf_flag)

  set requestedRespLevels, requestedProbLevels, requestedRelLevels, requestedGenRelLevels, respLevelTarget, and cdfFlag (used in combination with alternate ctors)
- void distribution_parameter_derivatives (bool dist_param_derivs)

  set distParamDerivs
- bool pdf_output () const

  get pdfOutput
- void pdf_output (bool output)

  set pdfOutput
- Pecos::ProbabilityTransformation & variable_transformation ()

  return natafTransform

Protected Member Functions

- NonD (ProblemDescDB &problem_db, Model &model)

  constructor
- NonD (unsigned short method_name, Model &model)

  alternate constructor for sample generation and evaluation "on the fly"
- NonD (unsigned short method_name, const RealVector &lower_bnds, const RealVector &upper_bnds)

  alternate constructor for sample generation "on the fly"
- ~NonD ()

  destructor
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void initialize_run ()

  utility function to perform common operations prior to pre_run(); typically memory initialization; setting of instance pointers
- void core_run ()
- void finalize_run ()

  utility function to perform common operations following post_run(); deallocation and resetting of instance pointers
- const Response & response_results () const

  return the final statistics from the nondeterministic iteration
- void response_results_active_set (const ActiveSet &set)

  set the active set within finalStatistics
- virtual void quantify_uncertainty ()=0

  Mapping of the core_run() virtual function for the NonD branch.
- virtual void initialize_response_covariance ()

  initializes respCovariance
- virtual void initialize_final_statistics ()

initializes finalStatistics for storing NonD final results

- virtual void update_final_statistics ()

  update finalStatistics::functionValues

- int generate_system_seed ()

  create a system-generated unique seed (when a seed is unspecified)

- void initialize_random_variable_transformation ()

  instantiate natafTransform

- void initialize_random_variable_types (short u_space_type)

  initializes ranVarTypesX and ranVarTypesU within natafTransform

- void initialize_random_variable_parameters ()

  initializes ranVarMeansX, ranVarStdDevsX, ranVarLowerBndsX, ranVarUpperBndsX, and ranVarAddtlParamsX within natafTransform

- void initialize_random_variable_correlations ()

  propagate iteratedModel correlations to natafTransform

- void verify_correlation_support ()

  verify that correlation warping supported by Der Kiureghian & Liu for given variable types

- void initialize_final_statistics_gradients ()

  initializes finalStatistics::functionGradients

- void update_aleatory_final_statistics ()

  update finalStatistics::functionValues from momentStats and computed{Prob,Rel,GenRel,Resp}Levels

- void update_system_final_statistics ()

  update system metrics from component metrics within finalStatistics

- void update_system_final_statistics_gradients ()

  update finalStatistics::functionGradients

- void initialize_distribution_mappings ()

  size computed{Resp,Prob,Rel,GenRel}Levels

- void print_distribution_mappings (std::ostream &s) const

  prints the z/p/beta/beta∗ mappings reflected in {requested,computed}{Resp,Prob,Rel,GenRel}Levels

- void print_system_mappings (std::ostream &s) const

  print system series/parallel mappings for response levels

- void transform_model (Model &x_model, Model &u_model, bool global_bounds=false, Real bound=10.)

  recast x_model from x-space to u-space to create u_model

- void construct_lhs (Iterator &u_space_sampler, Model &u_model, unsigned short sample_type, int num_samples, int seed, const String &rng, bool vary_pattern, short sampling_vars_mode=ACTIVE)

  assign a NonDLHSSampling instance within u_space_sampler

- void archive_allocate_mappings ()

  allocate results array storage for distribution mappings

- void archive_from_resp (size_t fn_index)

  archive the mappings from specified response levels for specified fn

- void archive_to_resp (size_t fn_index)

  archive the mappings to computed response levels for specified fn

Static Protected Member Functions

- static void vars_u_to_x_mapping (const Variables &u_vars, Variables &x_vars)

  static function for RecastModels used for forward mapping of u-space variables from NonD Iterators to x-space variables for Model evaluations
- static void vars_x_to_u_mapping (const Variables &x_vars, Variables &u_vars)

  static function for RecastModels used for inverse mapping of x-space variables from data import to u-space variables for NonD Iterators
- static void set_u_to_x_mapping (const Variables &u_vars, const ActiveSet &u_set, ActiveSet &x_set)

  static function for RecastModels used to map u-space ActiveSets from NonD Iterators to x-space ActiveSets for Model evaluations
- static void resp_x_to_u_mapping (const Variables &x_vars, const Variables &u_vars, const Response &x_response, Response &u_response)

  static function for RecastModels used to map x-space responses from Model evaluations to u-space responses for return to NonD Iterator.

Protected Attributes

- NonD * prevNondInstance

  pointer containing previous value of nondInstance
- Pecos::ProbabilityTransformation natafTransform

  Nonlinear variable transformation that encapsulates the required data for performing transformations from X -> Z -> U and back.
- size_t numContDesVars

  number of continuous design variables (modeled using uniform distribution for All view modes)
- size_t numDiscIntDesVars

  number of discrete integer design variables (modeled using discrete histogram distributions for All view modes)
- size_t numDiscStringDesVars

  number of discrete string design variables (modeled using discrete histogram distributions for All view modes)
- size_t numDiscRealDesVars

  number of discrete real design variables (modeled using discrete histogram distributions for All view modes)
- size_t numDesignVars

  total number of design variables
- size_t numContStateVars

  number of continuous state variables (modeled using uniform distribution for All view modes)
- size_t numDiscIntStateVars

  number of discrete integer state variables (modeled using discrete histogram distributions for All view modes)
- size_t numDiscStringStateVars

  number of discrete string state variables (modeled using discrete histogram distributions for All view modes)
- size_t numDiscRealStateVars

  number of discrete real state variables (modeled using discrete histogram distributions for All view modes)
- size_t numStateVars

  total number of state variables
- size_t numNormalVars

  number of normal uncertain variables (native space)

- size_t numLognormalVars

  number of lognormal uncertain variables (native space)
- size_t numUniformVars

  number of uniform uncertain variables (native space)
- size_t numLoguniformVars

  number of loguniform uncertain variables (native space)
- size_t numTriangularVars

  number of triangular uncertain variables (native space)
- size_t numExponentialVars

  number of exponential uncertain variables (native space)
- size_t numBetaVars

  number of beta uncertain variables (native space)
- size_t numGammaVars

  number of gamma uncertain variables (native space)
- size_t numGumbelVars

  number of gumbel uncertain variables (native space)
- size_t numFrechetVars

  number of frechet uncertain variables (native space)
- size_t numWeibullVars

  number of weibull uncertain variables (native space)
- size_t numHistogramBinVars

  number of histogram bin uncertain variables (native space)
- size_t numPoissonVars

  number of Poisson uncertain variables (native space)
- size_t numBinomialVars

  number of binomial uncertain variables (native space)
- size_t numNegBinomialVars

  number of negative binomial uncertain variables (native space)
- size_t numGeometricVars

  number of geometric uncertain variables (native space)
- size_t numHyperGeomVars

  number of hypergeometric uncertain variables (native space)
- size_t numHistogramPtIntVars

  number of histogram point integer uncertain variables (native space)
- size_t numHistogramPtStringVars

  number of histogram point string uncertain variables (native space)
- size_t numHistogramPtRealVars

  number of histogram point real uncertain variables (native space)
- size_t numContIntervalVars

  number of continuous interval uncertain variables (native space)
- size_t numDiscIntervalVars

  number of discrete interval uncertain variables (native space)
- size_t numDiscSetIntUncVars

number of discrete integer set uncertain variables (native space)

- size_t numDiscSetStringUncVars

  number of discrete integer set uncertain variables (native space)

- size_t numDiscSetRealUncVars

  number of discrete real set uncertain variables (native space)

- size_t numContAleatUncVars

  total number of continuous aleatory uncertain variables (native space)

- size_t numDiscIntAleatUncVars

  total number of discrete integer aleatory uncertain variables (native space)

- size_t numDiscStringAleatUncVars

  total number of discrete string aleatory uncertain variables (native space)

- size_t numDiscRealAleatUncVars

  total number of discrete real aleatory uncertain variables (native space)

- size_t numAleatoryUncVars

  total number of aleatory uncertain variables (native space)

- size_t numContEpistUncVars

  total number of continuous epistemic uncertain variables (native space)

- size_t numDiscIntEpistUncVars

  total number of discrete integer epistemic uncertain variables (native space)

- size_t numDiscStringEpistUncVars

  total number of discrete string epistemic uncertain variables (native space)

- size_t numDiscRealEpistUncVars

  total number of discrete real epistemic uncertain variables (native space)

- size_t numEpistemicUncVars

  total number of epistemic uncertain variables (native space)

- size_t numUncertainVars

  total number of uncertain variables (native space)

- bool epistemicStats

  flag for computing interval-type metrics instead of integrated metrics If any epistemic variables are active in a metric evaluation, then this flag is set.

- RealMatrix momentStats

  moments of response functions (mean, std deviation, skewness, and kurtosis calculated in compute_moments()), indexed as (moment,fn)

- RealVectorArray requestedRespLevels

  requested response levels for all response functions

- RealVectorArray computedProbLevels

  output probability levels for all response functions resulting from requestedRespLevels

- RealVectorArray computedRelLevels

  output reliability levels for all response functions resulting from requestedRespLevels

- RealVectorArray computedGenRelLevels

  output generalized reliability levels for all response functions resulting from requestedRespLevels

- short respLevelTarget

  indicates mapping of z->p (PROBABILITIES), z->beta (RELIABILITIES), or z->beta∗ (GEN_RELIABILITIES)

- short respLevelTargetReduce

  indicates component or system series/parallel failure metrics
- RealVectorArray requestedProbLevels

  requested probability levels for all response functions
- RealVectorArray requestedRelLevels

  requested reliability levels for all response functions
- RealVectorArray requestedGenRelLevels

  requested generalized reliability levels for all response functions
- RealVectorArray computedRespLevels

  output response levels for all response functions resulting from requestedProbLevels, requestedRelLevels, or requestedGenRelLevels
- size_t totalLevelRequests

  total number of levels specified within requestedRespLevels, requestedProbLevels, and requestedRelLevels
- bool cdfFlag

  flag for type of probabilities/reliabilities used in mappings: cumulative/CDF (true) or complementary/CCDF (false)
- bool pdfOutput

  flag for managing output of response probability density functions (PDFs)
- Response finalStatistics

  final statistics from the uncertainty propagation used in strategies: response means, standard deviations, and probabilities of failure
- size_t miPLIndex

  index for the active ParallelLevel within ParallelConfiguration::miPLIters

Static Protected Attributes

- static NonD * nondInstance

  pointer to the active object instance used within static evaluator functions in order to avoid the need for static data

Private Member Functions

- void distribute_levels (RealVectorArray &levels, bool ascending=true)

  convenience function for distributing a vector of levels among multiple response functions if a short-hand specification is employed.
- void distribution_mappings_file (size_t fn_index) const

  Write distribution mappings to a file for a single response.
- void print_distribution_map (size_t fn_index, std::ostream &s) const

  Print distribution mapping for a single response function to ostream.
- unsigned short pecos_to_dakota_variable_type (unsigned short pecos_var_type)

  convert from Pecos To Dakota variable enumeration type for continuous aleatory uncertain variables used in variable transformations

Private Attributes

- bool distParamDerivs

  flags calculation of derivatives with respect to distribution parameters s within resp_x_to_u_mapping() using the chain rule df/dx dx/ds. The default is to calculate derivatives with respect to standard random variables u using the chain rule df/dx dx/du.

13.78.1 Detailed Description

Base class for all nondetermistic iterators (the DAKOTA/UQ branch).

The base class for nondeterministic iterators consolidates uncertain variable data and probabilistic utilities for inherited classes.

13.78.2 Member Function Documentation

void initialize_random_variables (short *u_space_type*)

initialize natafTransform based on distribution data from iteratedModel

Build ProbabilityTransformation::ranVar arrays containing the uncertain variable distribution types and their corresponding means/standard deviations. This function is used when the Model variables are in x-space.

References NonD::initialize_random_variable_correlations(), NonD::initialize_random_variable_parameters(), NonD::initialize_random_variable_transformation(), NonD::initialize_random_variable_types(), NonD::natafTransform, and NonD::verify_correlation_support().

Referenced by NonDExpansion::compute_statistics(), NonDGlobalReliability::importance_sampling(), NonDLocalReliability::initialize_class_data(), NonDExpansion::initialize_expansion(), NonDAdaptImpSampling::NonDAdaptImpSampling(), and NonDGlobalReliability::optimize_gaussian_process().

void initialize_random_variables (const Pecos::ProbabilityTransformation & *transform*)

alternate form: initialize natafTransform based on incoming data

This function is commonly used to publish tranformation data when the Model variables are in a transformed space (e.g., u-space) and ProbabilityTransformation::ranVarTypes et al. may not be generated directly. This allows for the use of inverse transformations to return the transformed space variables to their original states.

References NonD::initialize_random_variable_transformation(), NonD::natafTransform, NonD::numContDesVars, and NonD::numContStateVars.

void initialize_run ( ) `[inline], [protected], [virtual]`

utility function to perform common operations prior to pre_run(); typically memory initialization; setting of instance pointers

Perform initialization phases of run sequence, like allocating memory and setting instance pointers. Commonly used in sub-iterator executions. This is a virtual function; when re-implementing, a derived class must call its nearest parent's initialize_run(), typically *before* performing its own implementation steps.

Reimplemented from Analyzer.

References Analyzer::initialize_run(), NonD::nondInstance, and NonD::prevNondInstance.

void core_run ( ) `[inline], [protected], [virtual]`

Performs a forward uncertainty propagation of parameter distributions into response statistics.

Reimplemented from Iterator.

References Analyzer::bestVarsRespMap, and NonD::quantify_uncertainty().

void finalize_run ( ) `[inline], [protected], [virtual]`

utility function to perform common operations following post_run(); deallocation and resetting of instance pointers

Optional: perform finalization phases of run sequence, like deallocating memory and resetting instance pointers. Commonly used in sub-iterator executions. This is a virtual function; when re-implementing, a derived class must call its nearest parent's finalize_run(), typically *after* performing its own implementation steps.

Reimplemented from Iterator.

References Iterator::finalize_run(), NonD::nondInstance, and NonD::prevNondInstance.

void initialize_final_statistics ( ) [protected], [virtual]

initializes finalStatistics for storing NonD final results

Default definition of virtual function (used by sampling, reliability, and stochastic expansion methods) defines the set of statistical results to include means, standard deviations, and level mappings.

Reimplemented in NonDInterval.

References Dakota::abort_handler(), NonD::cdfFlag, Model::cv(), ActiveSet::derivative_vector(), NonD::epistemicStats, NonD::finalStatistics, Response::function_labels(), Model::inactive_continuous_variable_ids(), Iterator::iteratedModel, Analyzer::numFunctions, NonD::requestedGenRelLevels, NonD::requestedProbLevels, NonD::requestedRelLevels, NonD::requestedRespLevels, NonD::respLevelTarget, NonD::respLevelTargetReduce, and NonD::totalLevelRequests.

Referenced by NonDExpansion::NonDExpansion(), NonDIntegration::NonDIntegration(), NonDReliability::NonDReliability(), NonDSampling::NonDSampling(), and NonD::requested_levels().

void initialize_random_variable_types (short *u_space_type*) [protected]

initializes ranVarTypesX and ranVarTypesU within natafTransform

Build ProbabilityTransformation::ranVar arrays containing the uncertain variable distribution types and their corresponding means/standard deviations. This function is used when the Model variables are in x-space.

References Dakota::abort_handler(), Model::aleatory_distribution_parameters(), Model::cv(), Iterator::iteratedModel, NonD::natafTransform, NonD::numBetaVars, NonD::numContDesVars, NonD::numContIntervalVars, NonD::numContStateVars, NonD::numExponentialVars, NonD::numFrechetVars, NonD::numGammaVars, NonD::numGumbelVars, NonD::numHistogramBinVars, NonD::numLognormalVars, NonD::numLoguniformVars, NonD::numNormalVars, NonD::numTriangularVars, NonD::numUniformVars, and NonD::numWeibullVars.

Referenced by NonDExpansion::initialize(), NonD::initialize_random_variables(), NonDBayesCalibration::NonDBayesCalibration(), NonDIntegration::NonDIntegration(), and NonDReliability::NonDReliability().

void initialize_random_variable_parameters ( ) [protected]

initializes ranVarMeansX, ranVarStdDevsX, ranVarLowerBndsX, ranVarUpperBndsX, and ranVarAddtlParamsX within natafTransform

Build ProbabilityTransformation::ranVar arrays containing the uncertain variable distribution types and their corresponding means/standard deviations. This function is used when the Model variables are in x-space.

References Dakota::abort_handler(), Model::aleatory_distribution_parameters(), Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::cv(), Iterator::iteratedModel, NonD::natafTransform, NonD::numBetaVars, NonD::numContDesVars, NonD::numContIntervalVars, NonD::numContStateVars, NonD::numExponentialVars, NonD::numFrechetVars, NonD::numGammaVars, NonD::numGumbelVars, NonD::numHistogramBinVars, NonD::numLognormalVars, NonD::numLoguniformVars, NonD::numNormalVars, NonD::numTriangularVars, NonD::numUniformVars, and NonD::numWeibullVars.

Referenced by NonDExpansion::initialize_expansion(), NonD::initialize_random_variables(), NonDLocalReliability::mean_value(), NonDLocalReliability::mpp_search(), NonDGlobalReliability::quantify_uncertainty(), and NonDBayesCalibration::quantify_uncertainty().

void print_distribution_mappings (std::ostream & *s*) const [protected]

prints the z/p/beta/beta∗ mappings reflected in {requested,computed}{Resp,Prob,Rel,GenRel}Levels

Print distribution mappings, including to file per response.

References NonD::distribution_mappings_file(), Analyzer::numFunctions, Iterator::outputLevel, NonD::print_distribution_map(), NonD::requestedGenRelLevels, NonD::requestedProbLevels, NonD::requestedRelLevels, NonD::requestedRespLevels, and Dakota::write_precision.

Referenced by NonDExpansion::print_results(), NonDAdaptImpSampling::print_results(), NonDGPImpSampling::print_results(), NonDAdaptiveSampling::print_results(), NonDPOFDarts::print_results(), and NonDSampling::print_statistics().

void vars_u_to_x_mapping (const Variables & *u_vars*, Variables & *x_vars*) [inline], [static], [protected]

static function for RecastModels used for forward mapping of u-space variables from NonD Iterators to x-space variables for Model evaluations

Map the variables from iterator space (u) to simulation space (x).

References Variables::continuous_variables(), Variables::continuous_variables_view(), NonD::natafTransform, and NonD::nondInstance.

Referenced by NonD::transform_model().

void vars_x_to_u_mapping (const Variables & *x_vars*, Variables & *u_vars*) [inline], [static], [protected]

static function for RecastModels used for inverse mapping of x-space variables from data import to u-space variables for NonD Iterators

Map the variables from simulation space (x) to iterator space (u).

References Variables::continuous_variables(), Variables::continuous_variables_view(), NonD::natafTransform, and NonD::nondInstance.

Referenced by NonD::transform_model().

void set_u_to_x_mapping (const Variables & *u_vars*, const ActiveSet & *u_set*, ActiveSet & *x_set*) [static], [protected]

static function for RecastModels used to map u-space ActiveSets from NonD Iterators to x-space ActiveSets for Model evaluations

Define the DVV for x-space derivative evaluations by augmenting the iterator requests to account for correlations.

References Dakota::_NPOS, Variables::all_continuous_variable_ids(), Dakota::contains(), Variables::continuous_variable_ids(), ActiveSet::derivative_vector(), Dakota::find_index(), Variables::inactive_continuous_variable_ids(), NonD::natafTransform, and NonD::nondInstance.

Referenced by NonD::transform_model().

void print_distribution_map (size_t *fn_index*, std::ostream & *s*) const [private]

Print distribution mapping for a single response function to ostream.

Print the distribution mapping for a single response function to the passed output stream

References NonD::cdfFlag, NonD::computedGenRelLevels, NonD::computedProbLevels, NonD::computedRelLevels, NonD::computedRespLevels, Iterator::iteratedModel, NonD::requestedGenRelLevels, NonD::requestedProbLevels, NonD::requestedRelLevels, NonD::requestedRespLevels, NonD::respLevelTarget, Model::response_labels(), and Dakota::write_precision.

Referenced by NonD::distribution_mappings_file(), and NonD::print_distribution_mappings().

The documentation for this class was generated from the following files:

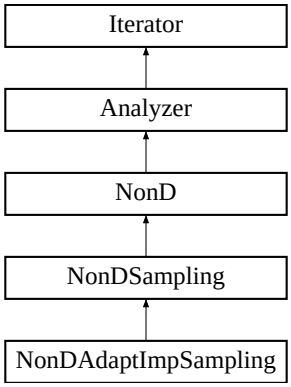
- DakotaNonD.hpp

- DakotaNonD.cpp

13.79 NonDAdaptImpSampling Class Reference

Class for the Adaptive Importance Sampling methods within DAKOTA.

Inheritance diagram for NonDAdaptImpSampling:



Public Member Functions

- NonDAdaptImpSampling (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NonDAdaptImpSampling (Model &model, unsigned short sample_type, int samples, int seed, const String &rng, bool vary_pattern, unsigned short is_type, bool cdf_flag, bool x_space_model, bool use_model_-bounds)

  alternate constructor for on-the-fly instantiations
- ~NonDAdaptImpSampling ()

  destructor
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void quantify_uncertainty ()

  performs adaptive importance sampling and computes probability of failure
- void print_results (std::ostream &s)

  print the final statistics
- void initialize (const RealVectorArray &full_points, bool x_space_data, size_t resp_index, Real initial_prob, Real failure_threshold)

  initializes data needed for importance sampling: an initial set of points around which to sample, a failure threshold, an initial probability to refine, and flags to control transformations

- void initialize (const RealMatrix &full_points, bool x_space_data, size_t resp_index, Real initial_prob, Real failure_threshold)

  initializes data needed for importance sampling: an initial set of points around which to sample, a failure threshold, an initial probability to refine, and flags to control transformations
- void initialize (const RealVector &full_point, bool x_space_data, size_t resp_index, Real initial_prob, Real failure_threshold)

  initializes data needed for importance sampling: an initial point around which to sample, a failure threshold, an initial probability to refine, and flags to control transformations
- Real final_probability ()

  returns the final probability calculated by the importance sampling

Private Member Functions

- void select_rep_points (const RealVectorArray &var_samples_u, const RealVector &fn_samples)

  select representative points from a set of samples
- void converge_statistics (bool cov_flag)

  iteratively generate samples and select representative points until probability and (optionally) coefficient of variation converge
- void generate_samples (RealVectorArray &var_samples_u)

  generate a set of samples based on multimodal sampling density
- void evaluate_samples (const RealVectorArray &var_samples_u, RealVector &fn_samples)

  evaluate the model at the sample points and store the responses
- void calculate_statistics (const RealVectorArray &var_samples_u, const RealVector &fn_samples, size_t total_samples, Real &sum_prob, Real &prob, bool compute_cov, Real &sum_var, Real &cov)

  calculate the probability of exceeding the failure threshold and the coefficent of variation (if requested)
- Real distance (const RealVector &a, const RealVector &b)

  compute Euclidean distance between points a and b
- Real recentered_density (const RealVector &sample_point)

  compute density between a representative point and a sample point, assuming standard normal

Private Attributes

- Model uSpaceModel

  importance sampling is performed in standardized probability space. This u-space model is either passed in (alternate constructor for helper AIS) or constructed using transform_model() (standard constructor for stand-alone AIS)
- unsigned short importanceSamplingType

  integration type (is, ais, mmais) provided by input specification
- bool initLHS

  flag to identify if initial points are generated from an LHS sample
- bool useModelBounds

  flag to control if the sampler should respect the model bounds
- bool invertProb

  flag for inversion of probability values using 1.-p
- int refineSamples

size of sample batch within each refinement iteration

- size_t respFnIndex

  the active response function index in the model to be sampled

- RealVector designPoint

  design subset for which uncertain subset is being sampled

- RealVectorArray initPointsU

  the original set of u-space samples passed in initialize()

- RealVectorArray repPointsU

  the set of representative points in u-space around which to sample

- RealVector repWeights

  the weight associated with each representative point

- Real probEstimate

  the probability estimate that is iteratively refined by importance sampling

- Real failThresh

  the failure threshold (z-bar) for the problem.

Additional Inherited Members

13.79.1 Detailed Description

Class for the Adaptive Importance Sampling methods within DAKOTA.

The NonDAdaptImpSampling implements the multi-modal adaptive importance sampling used for reliability calculations. (eventually we will want to broaden this). Need to add more detail to this description.

13.79.2 Constructor & Destructor Documentation

NonDAdaptImpSampling (ProblemDescDB & *problem_db*, Model & *model*)

standard constructor

This is the primary constructor. It accepts a Model reference. It will perform refinement for all response QOI and all probability levels.

References NonD::initialize_random_variables(), Iterator::iteratedModel, NonDSampling::numSamples, NonDAdaptImpSampling::refineSamples, NonDSampling::statsFlag, NonD::transform_model(), NonDAdaptImpSampling::useModelBounds, and NonDAdaptImpSampling::uSpaceModel.

NonDAdaptImpSampling (Model & *model*, unsigned short *sample_type*, int *refine_samples*, int *refine_seed*, const String & *rng*, bool *vary_pattern*, unsigned short *is_type*, bool *cdf_flag*, bool *x_space_model*, bool *use_model_bounds*)

alternate constructor for on-the-fly instantiations

This is an alternate constructor for instantiations on the fly using a Model but no ProblemDescDB. It will perform refinement for one response QOI and one probability level (passed in initialize()).

References NonD::cdfFlag, Iterator::maxEvalConcurrency, NonDAdaptImpSampling::refineSamples, NonD::transform_model(), NonDAdaptImpSampling::useModelBounds, and NonDAdaptImpSampling::uSpaceModel.

13.79.3 Member Function Documentation

void initialize (const RealVectorArray & *acv_points,* bool *x_space_data,* size_t *resp_index,* Real *initial_prob,* Real *failure_threshold*)

initializes data needed for importance sampling: an initial set of points around which to sample, a failure threshold, an initial probability to refine, and flags to control transformations

Initializes data using a vector array of starting points.

References NonDAdaptImpSampling::designPoint, NonDAdaptImpSampling::failThresh, NonDAdaptImpSampling::initPointsU, NonDAdaptImpSampling::invertProb, NonD::natafTransform, NonD::numContDesVars, NonD::numUncertainVars, NonDAdaptImpSampling::probEstimate, and NonDAdaptImpSampling::respFnIndex.

Referenced by NonDExpansion::compute_statistics(), NonDGlobalReliability::importance_sampling(), and NonDAdaptImpSampling::quantify_uncertainty().

void initialize (const RealMatrix & *acv_points,* bool *x_space_data,* size_t *resp_index,* Real *initial_prob,* Real *failure_threshold*)

initializes data needed for importance sampling: an initial set of points around which to sample, a failure threshold, an initial probability to refine, and flags to control transformations

Initializes data using a matrix of starting points.

References NonDAdaptImpSampling::designPoint, NonDAdaptImpSampling::failThresh, NonDAdaptImpSampling::initPointsU, NonDAdaptImpSampling::invertProb, NonD::natafTransform, NonD::numContDesVars, Analyzer::numContinuousVars, NonD::numUncertainVars, NonDAdaptImpSampling::probEstimate, and NonDAdaptImpSampling::respFnIndex.

void initialize (const RealVector & *acv_point,* bool *x_space_data,* size_t *resp_index,* Real *initial_prob,* Real *failure_threshold*)

initializes data needed for importance sampling: an initial point around which to sample, a failure threshold, an initial probability to refine, and flags to control transformations

Initializes data using only one starting point.

References NonDAdaptImpSampling::designPoint, NonDAdaptImpSampling::failThresh, NonDAdaptImpSampling::initPointsU, NonDAdaptImpSampling::invertProb, NonD::natafTransform, NonD::numContDesVars, NonD::numUncertainVars, NonDAdaptImpSampling::probEstimate, and NonDAdaptImpSampling::respFnIndex.

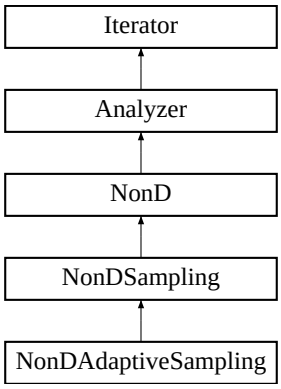
The documentation for this class was generated from the following files:

- NonDAdaptImpSampling.hpp
- NonDAdaptImpSampling.cpp

13.80 NonDAdaptiveSampling Class Reference

Class for testing various Adaptively sampling methods using geometric, statisctical, and topological information of the surrogate.

Inheritance diagram for NonDAdaptiveSampling:



Public Member Functions

- NonDAdaptiveSampling (ProblemDescDB &problem_db, Model &model)

  standard constructor
- ~NonDAdaptiveSampling ()

  alternate constructor for sample generation and evaluation "on the fly" has not been implemented

Protected Member Functions

- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void quantify_uncertainty ()

  perform the GP importance sampling and return probability of failure.
- Real final_probability ()

  returns the probability calculated by the importance sampling
- void print_results (std::ostream &s)

  print the final statistics

Private Member Functions

- void calc_score_alm ()

  Function to compute the ALM scores for the candidate points ALM score is the variance computed by the surrogate at the point.
- void calc_score_delta_x ()

  Function to compute the Distance scores for the candidate points Distance score is the shortest distance between the candidate and an existing training point.
- void calc_score_delta_y ()

  Function to compute the Gradient scores for the candidate points Gradient score is the function value difference between a candidate's surrogate response and its nearest evaluated true response from the training set.

- void calc_score_topo_bottleneck ()

  Function to compute the Bottleneck scores for the candidate points Bottleneck score is computed by determining the bottleneck distance between the persistence diagrams of two approximate Morse-Smale complices. The complices used include one built from only the training data, and another built from the training data and the single candidate.

- void calc_score_topo_avg_persistence (int respFnCount)

  Function to compute the Average Change in Persistence scores for the candidate points Avg_Persistence score is computed as the average change in persistence each point undergoes between two approximate Morse-Smale complices. The complices used include one built from only the training data, and another built from the training data and the single candidate.

- void calc_score_topo_highest_persistence (int respFnCount)

  Function to compute the Highest Persistence scores for the candidate points Highest Persistence score is calculated as a ranking of a set of candidates by constructing an approximate Morse-Smale complex over the entire set of candidates, using their surrogate responses, and the training data, using their true responses, and ranking points based on the most topological significance as measured by their persistence values. In the case where there are no topologically significant points, the point will be chosen randomly TODO: It may be wiser to fall back to a scheme that ranks points based on proximity to extrema, or the most significant extream?

- void calc_score_topo_alm_hybrid (int respFnCount)

  Function to comptue the Hybrid scores for the candidate points Hybrid score is computed the same as Avg_-Persistence score except that instead of computing one score, three scores are computing not only a mean surface, but a mean +/- std. dev. surfaces and then averaging the three separate scores. The hope is that you strike a balance between selecting points in topologically important areas and areas of high uncertainty.

- Real calc_score_alm (int respFnCount, RealVector &test_point)

  Same as the otehr function of the same name, only this allows the user to specify the location of the candidate.

- Real calc_score_delta_x (int respFnCount, RealVector &test_point)

  Same as the otehr function of the same name, only this allows the user to specify the location of the candidate.

- Real calc_score_delta_y (int respFnCount, RealVector &test_point)

  Same as the otehr function of the same name, only this allows the user to specify the location of the candidate.

- Real calc_score_topo_bottleneck (int respFnCount, RealVector &test_point)

  Same as the otehr function of the same name, only this allows the user to specify the location of the candidate.

- Real calc_score_topo_avg_persistence (int respFnCount, RealVector &test_point)

  Same as the otehr function of the same name, only this allows the user to specify the location of the candidate.

- Real calc_score_topo_alm_hybrid (int respFnCount, RealVector &test_point)

  Same as the otehr function of the same name, only this allows the user to specify the location of the candidate.

- Real compute_rmspe ()

  Using the validationSet, compute the RMSE over the surface.

- void compare_complices (int dim, std::ostream &output)

  Using the validationSet, compute the approximate Morse-Smale complices of the true model over the validationSet as well as the surrogate model over the validationSet, and output some topological comparisons.

- void parse_options ()

  Parse misc_options specified in a user input deck.

- RealVectorArray drawNewX (int this_k, int respFnCount=0)

  function to pick the next X value to be evaluated by the Iterated model

- void output_round_data (int round, int respFnCount=0)

  Temporary function for dumping validation data to output files to be visualized in TopoAS.

- void update_amsc (int respFnCount=0)

Update the approximate Morse-Smale complex based on the training points and selected candidates. Uses surrogate function responses.

- void construct_fsu_sampler (Iterator &u_space_sampler, Model &u_model, int num_samples, int seed, unsigned short sample_type)

  Copy of construct_lhs only it allows for the construction of FSU sample designs. This can break the fsu_cvt, so it is not used at the moment, and these designs only affect the initial sample build not the candidate sets constructed at each round.
- void output_for_optimization (int dim)

  This function will write an input deck for a multi-start global optimization run of DAKOTA by extracting all of the local minima off the approximate Morse-Smale complex created from the validation set of the surrogate model.
- Real median (const RealVector &sorted_data)

  compute the median of the sorted values passed in
- void pick_new_candidates ()

  Pick new candidates from Emulator.
- void score_new_candidates ()

  Score New candidates based on the chosen metrics.

Private Attributes

- Iterator gpBuild

  LHS iterator for building the initial GP.
- Iterator gpEval

  LHS iterator for sampling on the GP.
- Iterator gpFinalEval

  LHS iterator for sampling on the final GP.
- Model gpModel

  GP model of response, one approximation per response function.
- int numRounds

  the number of rounds of additions of size batchSize to add to the original set of LHS samples
- int numPtsTotal

  the total number of points
- int numEmulEval

  the number of points evaluated by the GP each iteration
- int numFinalEmulEval

  number of points evaluated on the final GP
- int scoringMethod

  the type of scoring metric to use for sampling
- Real finalProb

  the final calculated probability (p)
- RealVectorArray gpCvars

  Vector to hold the current values of the current sample inputs on the GP.
- RealVectorArray gpMeans

  Vector to hold the current values of the current mean estimates for the sample values on the GP.
- RealVectorArray gpVar

Vector to hold the current values of the current variance estimates for the sample values on the GP.

- RealVector emulEvalScores

  Vector to hold the scored values for the current GP samples.
- RealVector predictionErrors

  Vector to hold the RMSE after each round of adaptively fitting the model.
- RealVectorArray validationSet

  Validation point set used to determine predictionErrors above.
- RealVector yTrue

  True function responses at the values corresponding to validationSet.
- RealVector yModel

  Surrogate function responses at the values corresponding to validationSet.
- int validationSetSize

  Number of points used in the validationSet.
- int batchSize

  Number of points to add each round, default = 1.
- String batchStrategy

  String describing the tpye of batch addition to use. Allowable values are naive, distance, topology.
- String outputDir

  Temporary string for dumping validation files used in TopoAS visualization.
- String scoringMetric

  String describing the method for scoring candidate points. Options are: alm, distance, gradient, highest_persistence, avg_persistence, bottleneck, alm_topo_hybrid Note: alm and alm_topo_hybrid will fail when used with surrogates other than global_kriging as it is based on the variance of the surrogate. At the time of implementation, global_kriging is the only surrogate capable of yielding this information.
- unsigned short sampleDesign

  enum describing the initial sample design. Options are: RANDOM_SAMPLING, FSU_CVT, FSU_HALTON, FSU_HAMMERSLEY
- String approx_type

  String describing type of surrogate is used to fit the data. Options are: global_kriging, global_mars, global_neural_network, global_polynomial, globabl_moving_least_squares, global_radial_basis.
- MS_Complex * AMSC

  The approximate Morse-Smale complex data structure.
- int numKneighbors

  The number of approximate nearest neighbors to use in computing the AMSC.
- bool outputValidationData

  Temporary variable for toggling writing of data files to be used by TopoAS.

Additional Inherited Members

13.80.1 Detailed Description

Class for testing various Adaptively sampling methods using geometric, statisctical, and topological information of the surrogate.

NonDAdaptiveSampling implements an adaptive sampling method based on the work presented in Adaptive Sampling with Topological Scores by Dan Maljovec, Bei Wang, Ana Kupresanin, Gardar Johannesson, Valerio Pascucci, and Peer-Timo Bremer presented in IJUQ (insert issue). The method computes scores based on the topology of the known data and the topology of the surrogate model. A number of alternate adaption strategies are offered as well.

13.80.2 Constructor & Destructor Documentation

NonDAdaptiveSampling (ProblemDescDB & *problem_db,* Model & *model*)

standard constructor

```
This constructor is called for a standard letter-envelope iterator
```

instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

References NonDAdaptiveSampling::AMSC, NonDAdaptiveSampling::approx_type, Iterator::assign_rep(), Model::assign_rep(), NonDAdaptiveSampling::batchSize, NonDAdaptiveSampling::batchStrategy, NonDAdaptiveSampling::construct_fsu_sampler(), NonD::construct_lhs(), ProblemDescDB::get_bool(), ProblemDescDB::get_int(), ProblemDescDB::get_sa(), ProblemDescDB::get_string(), NonDAdaptiveSampling::gpBuild, NonDAdaptiveSampling::gpEval, NonDAdaptiveSampling::gpFinalEval, NonDAdaptiveSampling::gpModel, Model::gradient_type(), Model::hessian_type(), Iterator::iteratedModel, Iterator::maxIterations, NonDAdaptiveSampling::numEmulEval, NonDAdaptiveSampling::numFinalEmulEval, NonDAdaptiveSampling::numKneighbors, NonDAdaptiveSampling::numRounds, NonDSampling::numSamples, NonDAdaptiveSampling::outputDir, Iterator::outputLevel, NonDAdaptiveSampling::outputValidationData, NonDAdaptiveSampling::parse_options(), Iterator::probDescDB, NonDSampling::randomSeed, NonDSampling::rngName, NonDAdaptiveSampling::sampleDesign, NonDAdaptiveSampling::scoringMetric, NonDSampling::vary_pattern(), and NonDSampling::varyPattern.

~NonDAdaptiveSampling ( )

alternate constructor for sample generation and evaluation "on the fly" has not been implemented

destructor

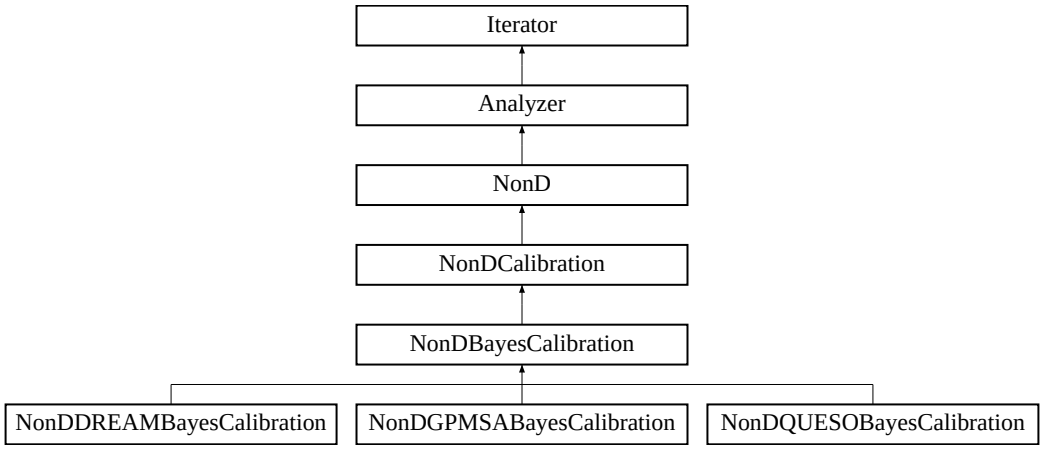
The documentation for this class was generated from the following files:

- NonDAdaptiveSampling.hpp
- NonDAdaptiveSampling.cpp

13.81 NonDBayesCalibration Class Reference

Base class for Bayesian inference: generates posterior distribution on model parameters given experimental data.

Inheritance diagram for NonDBayesCalibration:



Public Member Functions

- NonDBayesCalibration (ProblemDescDB &problem_db, Model &model)

  standard constructor
- ~NonDBayesCalibration ()

  destructor

Protected Member Functions

- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void quantify_uncertainty ()

  Mapping of the core_run() virtual function for the NonD branch.
- const Model & algorithm_space_model () const

Protected Attributes

- Model emulatorModel

  Model instance employed in the likelihood function; provides response function values from Gaussian processes, stochastic expansions (PCE/SC), or direct access to simulations (no surrogate option)
- bool standardizedSpace

  flag indicating use of a variable transformation to standardized probability space
- Iterator stochExpIterator

  NonDPolynomialChaos or NonDStochCollocation instance for defining a PCE/SC-based emulatorModel.
- Iterator lhsIterator

  LHS iterator for generating samples for GP.

Private Attributes

- short emulatorType

  the emulator type: NO_EMULATOR, GP_EMULATOR, PCE_EMULATOR, or SC_EMULATOR

Additional Inherited Members

13.81.1 Detailed Description

Base class for Bayesian inference: generates posterior distribution on model parameters given experimental data.

This class will eventually provide a general-purpose framework for Bayesian inference. In the short term, it only collects shared code between QUESO and GPMSA implementations.

13.81.2 Constructor & Destructor Documentation

NonDBayesCalibration (ProblemDescDB & *problem_db,* Model & *model*)

standard constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

References Iterator::algorithm_space_model(), Iterator::assign_rep(), Model::assign_rep(), NonD::cdfFlag, NonDBayesCalibration::emulatorModel, NonDBayesCalibration::emulatorType, ProblemDescDB::get_bool(), ProblemDescDB::get_int(), ProblemDescDB::get_short(), ProblemDescDB::get_string(), ProblemDescDB::get_usa(), Model::gradient_type(), Model::hessian_type(), NonD::initialize_random_variable_correlations(), NonD::initialize_random_variable_transformation(), NonD::initialize_random_variable_types(), Iterator::iteratedModel, Iterator::iterator_rep(), NonDBayesCalibration::lhsIterator, Iterator::maxEvalConcurrency, Iterator::outputLevel, Iterator::probDescDB, NonD::requested_levels(), NonD::respLevelTarget, NonD::respLevelTargetReduce, NonDBayesCalibration::standardizedSpace, NonDBayesCalibration::stochExpIterator, NonD::transform_model(), and NonD::verify_correlation_support().

13.81.3 Member Function Documentation

const Model & algorithm_space_model ( ) const `[inline], [protected], [virtual]`

default definition that gets redefined in selected derived Minimizers

Reimplemented from Analyzer.

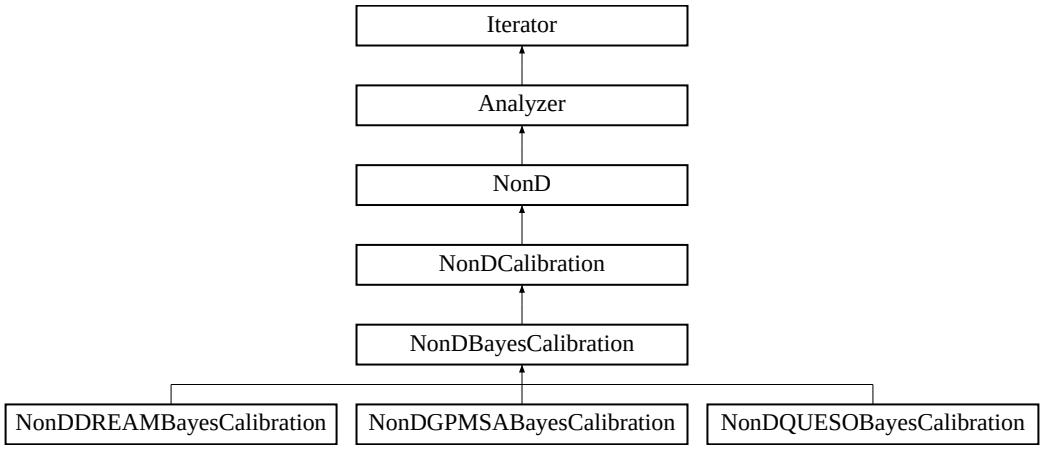
References NonDBayesCalibration::emulatorModel.

The documentation for this class was generated from the following files:

- NonDBayesCalibration.hpp
- NonDBayesCalibration.cpp

13.82 NonDCalibration Class Reference

Inheritance diagram for NonDCalibration:



Public Member Functions

- NonDCalibration (ProblemDescDB &problem_db, Model &model)

  standard constructor
- ~NonDCalibration ()

  destructor

Protected Member Functions

- void set_configuration_vars (Model &model, const RealVector &x)

  set the passed configuration variables into the model's state vars

Protected Attributes

- RealVector expStdDeviations

  1 or numFunctions standard deviations
- String expDataFileName

  filename from which to read experimental data; optionally configuration vars x and standard deviations sigma
- bool expDataFileAnnotated

  whether the data file is in annotated format
- size_t numExperiments

  number of experiments to read from data file
- size_t numExpConfigVars

  number of columns in data file which are state variables
- size_t numExpStdDeviationsRead

  how many sigmas to read from the data file (1 or numFunctions)
- ExperimentData expData

  Container for experimental data to which to calibrate model.

Private Member Functions

- bool find_state_index (unsigned short state_type, UShortMultiArrayConstView variable_types, std::string context_message, size_t &start_index)

  helper function to lookup a state_type enum variable type in the array of variables_types to find its start_index into the all array

Private Attributes

- size_t continuousConfigVars

  number of continuous configuration variables
- size_t discreteIntConfigVars

  number of discrete integer configuration variables
- size_t discreteRealConfigVars

  number of discrete real configuration variables
- size_t continuousConfigStart

  index of configuration variables in all continuous array

- size_t discreteIntConfigStart

  index of configuration variables in all discrete integer array
- size_t discreteRealConfigStart

  index of configuration variables in all discrete real array

Additional Inherited Members

13.82.1 Detailed Description

This class ...

13.82.2 Constructor & Destructor Documentation

NonDCalibration (ProblemDescDB & *problem_db,* Model & *model*)

standard constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

References Dakota::abort_handler(), Model::all_continuous_variable_types(), Model::all_discrete_int_variable_types(), Model::all_discrete_real_variable_types(), NonDCalibration::continuousConfigStart, NonDCalibration::continuousConfigVars, NonDCalibration::discreteIntConfigStart, NonDCalibration::discreteIntConfigVars, NonDCalibration::discreteRealConfigStart, NonDCalibration::discreteRealConfigVars, NonDCalibration::expDataFileName, NonDCalibration::expStdDeviations, NonDCalibration::find_state_index(), ProblemDescDB::get_sizet(), Iterator::iteratedModel, NonDCalibration::numExpConfigVars, Analyzer::numFunctions, and Iterator::probDescDB.

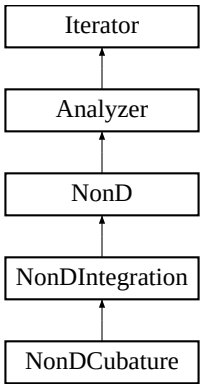
The documentation for this class was generated from the following files:

- NonDCalibration.hpp
- NonDCalibration.cpp

13.83 NonDCubature Class Reference

Derived nondeterministic class that generates N-dimensional numerical cubature points for evaluation of expectation integrals.

Inheritance diagram for NonDCubature:



Public Member Functions

- NonDCubature (Model &model, const Pecos::ShortArray &u_types, unsigned short cub_int_order)
- unsigned short integrand_order () const

  return cubIntOrder

Protected Member Functions

- NonDCubature (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDCubature ()

  destructor
- void initialize_grid (const std::vector< Pecos::BasisPolynomial > &poly_basis)

  initialize integration grid by drawing from polynomial basis settings
- void get_parameter_sets (Model &model)

  Returns one block of samples (ndim ∗ num_samples)
- void sampling_reset (int min_samples, bool all_data_flag, bool stats_flag)
- void increment_grid ()

  increment SSG level/TPQ order
- void increment_grid_preference (const RealVector &dim_pref)
- int num_samples () const

Private Member Functions

- void check_integration (const Pecos::ShortArray &u_types, const Pecos::AleatoryDistParams &adp)

  verify self-consistency of integration specification
- void increment_reference ()

  increment each cubIntOrderRef entry by 1

Private Attributes

- Pecos::CubatureDriver ∗ cubDriver

  convenience pointer to the numIntDriver representation
- unsigned short cubIntOrderRef

  reference point for Pecos::CubatureDriver::cubIntOrder: the original user specification for the number of Gauss points per dimension, plus any refinements posted by increment_grid()
- unsigned short cubIntRule

  the isotropic cubature integration rule

Additional Inherited Members

13.83.1 Detailed Description

Derived nondeterministic class that generates N-dimensional numerical cubature points for evaluation of expectation integrals.

This class is used by NonDPolynomialChaos, but could also be used for general numerical integration of moments. It employs Stroud cubature rules and extensions by D. Xiu.

13.83.2 Constructor & Destructor Documentation

NonDCubature (Model & *model,* const Pecos::ShortArray & *u_types,* unsigned short *cub_int_order*)

This alternate constructor is used for on-the-fly generation and evaluation of numerical cubature points.

References Model::aleatory_distribution_parameters(), NonDCubature::check_integration(), NonDCubature::cubDriver, NonDCubature::cubIntOrderRef, Iterator::iteratedModel, and NonDIntegration::numIntDriver.

NonDCubature (ProblemDescDB & *problem_db,* Model & *model*) `[protected]`

constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification. It is not currently used, as there is not yet a separate nond_cubature method specification.

References Model::aleatory_distribution_parameters(), NonDCubature::check_integration(), NonDIntegration::check_variables(), NonDCubature::cubDriver, NonDCubature::cubIntOrderRef, NonDCubature::cubIntRule, Iterator::iteratedModel, Iterator::maxEvalConcurrency, NonD::natafTransform, and NonDIntegration::numIntDriver.

13.83.3 Member Function Documentation

void sampling_reset (int *min_samples,* bool *all_data_flag,* bool *stats_flag*) `[protected], [virtual]`

used by DataFitSurrModel::build_global() to publish the minimum number of points needed from the cubature routine in order to build a particular global approximation.

Reimplemented from Iterator.

References NonDCubature::cubDriver, and NonDCubature::cubIntOrderRef.

void increment_grid_preference (const RealVector & *dim_pref*) `[inline], [protected], [virtual]`

Should not be used, but pure virtual must be defined.

Reimplemented from NonDIntegration.

References NonDCubature::increment_grid().

int num_samples ( ) const `[inline], [protected], [virtual]`

Return current number of evaluation points. Since the calculation of samples, collocation points, etc. might be costly, provide a default implementation here that backs out from the maxEvalConcurrency.

Reimplemented from Analyzer.

References NonDCubature::cubDriver.

void increment_reference ( ) `[inline], [private]`

increment each cubIntOrderRef entry by 1

cubIntOrderRef is a reference point for CubatureDriver::cubIntOrder, e.g., a lower bound

References NonDCubature::cubIntOrderRef.

Referenced by NonDCubature::increment_grid().

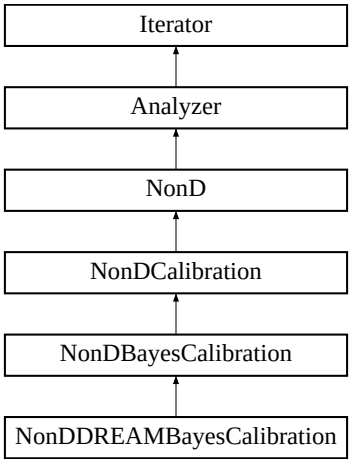
The documentation for this class was generated from the following files:

- NonDCubature.hpp
- NonDCubature.cpp

13.84 NonDDREAMBayesCalibration Class Reference

Bayesian inference using the DREAM approach.

Inheritance diagram for NonDDREAMBayesCalibration:



Public Member Functions

- NonDDREAMBayesCalibration (ProblemDescDB &problem_db, Model &model)

  standard constructor
- ~NonDDREAMBayesCalibration ()

  destructor

Static Public Member Functions

- static void problem_size (int &chain_num, int &cr_num, int &gen_num, int &pair_num, int &par_num)

  initializer for problem size characteristics in DREAM
- static void problem_value (std::string *chain_filename, std::string *gr_filename, double &gr_threshold, int &jumpstep, double limits[ ], int par_num, int &printstep, std::string *restart_read_filename, std::string *restart_write_filename)

  Filename and data initializer for DREAM.
- static double prior_density (int par_num, double zp[ ])

  Compute the prior density at specified point zp.
- static double * prior_sample (int par_num)

  Sample the prior and return an array of parameter values.
- static double sample_likelihood (int par_num, double zp[ ])

  Likelihood function for call-back from DREAM to DAKOTA for evaluation.

Protected Member Functions

- void quantify_uncertainty ()

  redefined from DakotaNonD

Protected Attributes

- Real likelihoodScale

  scale factor for proposal covariance
- int numSamples

  number of samples in the chain (e.g. number of MCMC samples)
- bool calibrateSigmaFlag

  flag to indicate if the sigma terms should be calibrated (default true)
- int randomSeed

  random seed to pass to QUESO
- RealVector paramMins

  lower bounds on calibrated parameters
- RealVector paramMaxs

  upper bounds on calibrated parameters
- int numChains

  number of concurrent chains
- int numGenerations

  number of generations
- int numCR

  number of CR-factors
- int crossoverChainPairs

  number of crossover chain pairs
- Real grThreshold

  threshold for the Gelmin-Rubin statistic
- int jumpStep

  how often to perform a long jump in generations
- std::vector< boost::math::uniform > priorDistributions

  uniform prior PDFs for each variable
- boost::mt19937 rnumGenerator

  random number engine for sampling the prior
- std::vector
  < boost::uniform_real< double > > priorSamplers

  samplers for the uniform prior PDFs for each variable

Private Attributes

- short emulatorType

  the emulator type: NO_EMULATOR, GP_EMULATOR, PCE_EMULATOR, or SC_EMULATOR

Static Private Attributes

- static NonDDREAMBayesCalibration * NonDDREAMInstance

  Pointer to current class instance for use in static callback functions.

Additional Inherited Members

13.84.1 Detailed Description

Bayesian inference using the DREAM approach.

This class performed Bayesian calibration using the DREAM (Markov Chain Monte Carlo acceleration by Differential Evolution) implementation of John Burkhardt (FSU), adapted from that of Guannan Zhang (ORNL)

13.84.2 Constructor & Destructor Documentation

NonDDREAMBayesCalibration (ProblemDescDB & *problem_db,* Model & *model*)

standard constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

References NonDDREAMBayesCalibration::crossoverChainPairs, NonDDREAMBayesCalibration::grThreshold, NonDDREAMBayesCalibration::jumpStep, NonDDREAMBayesCalibration::numChains, NonDDREAMBayes-Calibration::numCR, NonDDREAMBayesCalibration::numGenerations, and NonDDREAMBayesCalibration::num-Samples.

13.84.3 Member Function Documentation

void problem_size (int & *chain_num,* int & *cr_num,* int & *gen_num,* int & *pair_num,* int & *par_num*) `[static]`

initializer for problem size characteristics in DREAM

See documentation in DREAM examples)

References NonDDREAMBayesCalibration::crossoverChainPairs, NonDDREAMBayesCalibration::NonD-DREAMInstance, NonDDREAMBayesCalibration::numChains, Analyzer::numContinuousVars, NonDDREAM-BayesCalibration::numCR, and NonDDREAMBayesCalibration::numGenerations.

void problem_value (std::string * *chain_filename,* std::string * *gr_filename,* double & *gr_threshold,* int & *jumpstep,* double *limits[ ],* int *par_num,* int & *printstep,* std::string * *restart_read_filename,* std::string * *restart_write_filename*) `[static]`

Filename and data initializer for DREAM.

See documentation in DREAM examples)

References NonDDREAMBayesCalibration::grThreshold, NonDDREAMBayesCalibration::jumpStep, Non-DDREAMBayesCalibration::NonDDREAMInstance, NonDDREAMBayesCalibration::numChains, NonDDRE-AMBayesCalibration::paramMaxs, and NonDDREAMBayesCalibration::paramMins.

double prior_density (int *par_num,* double *zp[ ]*) `[static]`

Compute the prior density at specified point zp.

See documentation in DREAM examples)

References NonDDREAMBayesCalibration::NonDDREAMInstance, and NonDDREAMBayesCalibration-::priorDistributions.

double * prior_sample (int *par_num*) [static]

Sample the prior and return an array of parameter values.

See documentation in DREAM examples)

References NonDDREAMBayesCalibration::NonDDREAMInstance, NonDDREAMBayesCalibration::priorSamplers, and NonDDREAMBayesCalibration::rnumGenerator.

double sample_likelihood (int *par_num*, double *zp[]*) [static]

Likelihood function for call-back from DREAM to DAKOTA for evaluation.

Static callback function to evaluate the likelihood

References NonDDREAMBayesCalibration::calibrateSigmaFlag, Model::compute_response(), Model::continuous_variables(), Model::current_response(), NonDBayesCalibration::emulatorModel, NonDDREAMBayesCalibration::emulatorType, NonDCalibration::expData, Response::function_values(), NonDDREAMBayesCalibration::likelihoodScale, NonDDREAMBayesCalibration::NonDDREAMInstance, Analyzer::numContinuousVars, NonDCalibration::numExperiments, Analyzer::numFunctions, Iterator::outputLevel, ExperimentData::scalar_data(), and ExperimentData::scalar_sigma().

void quantify_uncertainty ( ) [protected], [virtual]

redefined from DakotaNonD

Perform the uncertainty quantification

Reimplemented from NonDBayesCalibration.

References NonDDREAMBayesCalibration::calibrateSigmaFlag, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), NonDBayesCalibration::emulatorModel, NonDDREAMBayesCalibration::emulatorType, NonDCalibration::expData, NonDCalibration::expDataFileAnnotated, NonDCalibration::expDataFileName, NonD::generate_system_seed(), Iterator::iterator_rep(), ExperimentData::load_scalar(), NonDDREAMBayesCalibration::NonDDREAMInstance, Analyzer::numContinuousVars, NonDCalibration::numExpConfigVars, NonDCalibration::numExperiments, NonDCalibration::numExpStdDeviationsRead, Analyzer::numFunctions, NonDDREAMBayesCalibration::numSamples, Iterator::outputLevel, NonDDREAMBayesCalibration::paramMaxs, NonDDREAMBayesCalibration::paramMins, NonDDREAMBayesCalibration::priorDistributions, NonDDREAMBayesCalibration::priorSamplers, NonDBayesCalibration::quantify_uncertainty(), NonDDREAMBayesCalibration::randomSeed, NonDDREAMBayesCalibration::rnumGenerator, ExperimentData::scalar_sigma(), NonDBayesCalibration::standardizedSpace, and NonDBayesCalibration::stochExpIterator.

13.84.4 Member Data Documentation

Real likelihoodScale [protected]

scale factor for proposal covariance

scale factor for likelihood

Referenced by NonDDREAMBayesCalibration::sample_likelihood().

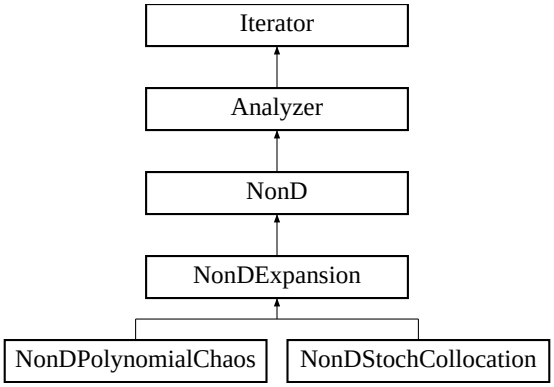
The documentation for this class was generated from the following files:

- NonDDREAMBayesCalibration.hpp
- NonDDREAMBayesCalibration.cpp

13.85 NonDExpansion Class Reference

Base class for polynomial chaos expansions (PCE) and stochastic collocation (SC)

Inheritance diagram for NonDExpansion:



Public Member Functions

- NonDExpansion (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NonDExpansion (unsigned short method_name, Model &model, short exp_coeffs_approach, short u_space_type, bool piecewise_basis, bool use_derivs)

  alternate constructor
- ~NonDExpansion ()

  destructor
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void quantify_uncertainty ()

  perform a forward uncertainty propagation using PCE/SC methods
- void print_results (std::ostream &s)

  print the final statistics
- const Model & algorithm_space_model () const

Protected Member Functions

- virtual void resolve_inputs (short &u_space_type, short &data_order)

  perform error checks and mode overrides
- virtual void initialize_u_space_model ()

  initialize uSpaceModel polynomial approximations with PCE/SC data
- virtual void initialize_expansion ()

  initialize random variable definitions and final stats arrays
- virtual void compute_expansion ()

form the expansion by calling uSpaceModel.build_approximation()

- virtual void increment_order ()

  uniformly increment the expansion order (PCE only)
- virtual void increment_specification_sequence ()

  increment the input specification sequence (PCE only)
- virtual void update_expansion ()

  update an expansion; avoids overhead in compute_expansion()
- virtual void print_coefficients (std::ostream &s)

  print expansion coefficients, as supported by derived instance
- virtual void archive_coefficients ()

  archive expansion coefficients, as supported by derived instance
- virtual Real compute_covariance_metric ()

  compute 2-norm of change in response covariance
- virtual Real compute_final_statistics_metric ()

  compute 2-norm of change in final statistics
- void initialize_response_covariance ()

  set covarianceControl defaults and shape respCovariance
- void update_final_statistics ()

  update function values within finalStatistics
- void update_final_statistics_gradients ()

  update function gradients within finalStatistics
- void initialize (short u_space_type)

  common constructor code for initialization of natafTransform
- void refine_expansion ()

  refine the reference expansion found by compute_expansion() using uniform/adaptive p-/h-refinement strategies
- void construct_cubature (Iterator &u_space_sampler, Model &g_u_model, unsigned short cub_int_order)

  assign a NonDCubature instance within u_space_sampler
- void construct_quadrature (Iterator &u_space_sampler, Model &g_u_model, const UShortArray &quad_order_seq, const RealVector &dim_pref)

  assign a NonDQuadrature instance within u_space_sampler based on a quad_order specification
- void construct_quadrature (Iterator &u_space_sampler, Model &g_u_model, int filtered_samples, const RealVector &dim_pref)

  assign a NonDQuadrature instance within u_space_sampler that generates a filtered tensor product sample set
- void construct_quadrature (Iterator &u_space_sampler, Model &g_u_model, int random_samples, int seed, const UShortArray &quad_order_seq, const RealVector &dim_pref)

  assign a NonDQuadrature instance within u_space_sampler that samples randomly from a tensor product multi-index
- void construct_sparse_grid (Iterator &u_space_sampler, Model &g_u_model, const UShortArray &ssg_level_seq, const RealVector &ssg_dim_pref)

  assign a NonDSparseGrid instance within u_space_sampler
- void construct_expansion_sampler ()

  construct the expansionSampler operating on uSpaceModel
- void compute_statistics ()

  calculate analytic and numerical statistics from the expansion
- void archive_moments ()

  archive the central moments (numerical and expansion) to ResultsDB

Protected Attributes

- Model uSpaceModel

  Model representing the approximate response function in u-space, after u-space recasting and orthogonal polynomial data fit recursions.
- short expansionCoeffsApproach

  method for collocation point generation and subsequent calculation of the expansion coefficients
- short expansionBasisType

  type of expansion basis: DEFAULT_BASIS or Pecos::{NODAL,HIERARCHICAL}_INTERPOLANT for SC or Pecos::{TENSOR_PRODUCT,TOTAL_ORDER,ADAPTED}_BASIS for PCE regression
- size_t numUncertainQuant

  number of invocations of quantify_uncertainty()
- int numSamplesOnModel

  number of truth samples performed on g_u_model to form the expansion
- int numSamplesOnExpansion

  number of approximation samples performed on the polynomial expansion in order to estimate probabilities
- bool nestedRules

  flag for indicating state of `nested` *and* `non_nested` *overrides of default rule nesting, which depends on the type of integration driver*
- bool piecewiseBasis

  flag for `piecewise` *specification, indicating usage of local basis polynomials within the stochastic expansion*
- bool useDerivs

  flag for `use_derivatives` *specification, indicating usage of derivative data (with respect to expansion variables) to enhance the calculation of the stochastic expansion.*
- short refineType

  refinement type: NO_REFINEMENT, P_REFINEMENT, or H_REFINEMENT
- short refineControl

  refinement control: NO_CONTROL, UNIFORM_CONTROL, LOCAL_ADAPTIVE_CONTROL, DIMENSION_ADAPTIVE_CONTROL_SOBOL, DIMENSION_ADAPTIVE_CONTROL_DECAY, or DIMENSION_ADAPTIVE_CONTROL_GENERALIZED
- unsigned short softConvLimit

  number of consecutive iterations within tolerance required to indicate soft convergence
- RealSymMatrix respCovariance

  symmetric matrix of analytic response covariance (full response covariance option)
- RealVector respVariance

  vector of response variances (diagonal response covariance option)
- RealVector initialPtU

  stores the initial variables data in u-space

Private Member Functions

- void reduce_total_sobol_sets (RealVector &avg_sobol)

  compute average of total Sobol' indices (from VBD) across the response set for use as an anisotropy indicator
- void reduce_decay_rate_sets (RealVector &min_decay)

  compute minimum of spectral coefficient decay rates across the response set for use as an anisotropy indicator

- void initialize_sets ()

  initialization of adaptive refinement using generalized sparse grids
- Real increment_sets ()

  perform an adaptive refinement increment using generalized sparse grids
- void finalize_sets (bool converged_within_tol)

  finalization of adaptive refinement using generalized sparse grids
- void compute_covariance ()

  calculate the response covariance (diagonal or full matrix)
- void compute_diagonal_variance ()

  calculate respVariance or diagonal terms respCovariance(i,i)
- void compute_off_diagonal_covariance ()

  calculate respCovariance(i,j) for j<i
- void print_moments (std::ostream &s)

  print expansion and numerical moments
- void print_covariance (std::ostream &s)

  print respCovariance
- void print_sobol_indices (std::ostream &s)

  print global sensitivity indices
- void print_local_sensitivity (std::ostream &s)

  print local sensitivities evaluated at initialPtU
- void compute_print_increment_results ()

  manage print of results following a refinement increment
- void compute_print_iteration_results (bool initialize)

  manage print of results following a refinement increment
- void compute_print_converged_results (bool print_override=false)

  manage print of results following convergence of iterative refinement

Private Attributes

- short ruleNestingOverride

  user override of default rule nesting: NO_NESTING_OVERRIDE, NESTED, or NON_NESTED
- short ruleGrowthOverride

  user override of default rule growth: NO_GROWTH_OVERRIDE, RESTRICTED, or UNRESTRICTED
- Iterator expansionSampler

  Iterator used for sampling on the uSpaceModel to generate approximate probability/reliability/response level statistics. Currently this is an LHS sampling instance, but AIS could also be used.
- Iterator importanceSampler

  Iterator used to refine the approximate probability estimates generated by the expansionSampler using importance sampling.
- bool expSampling

  flag to indicate calculation of numerical statistics by sampling on the expansion
- bool impSampling

  flag to use LHS sampling or MMAIS sampling on the expansion
- RealMatrix expGradsMeanX

derivative of the expansion with respect to the x-space variables evaluated at the means (used as uncertainty importance metrics)

- bool vbdFlag

  flag indicating the activation of variance-bsaed decomposition for computing Sobol' indices

- unsigned short vbdOrderLimit

  limits the order of interactions within the component Sobol' indices

- Real vbdDropTol

  tolerance for omitting output of small VBD indices

- short covarianceControl

  enumeration for controlling response covariance calculation and output: {DEFAULT,DIAGONAL,FULL}_COVARIANCE

Additional Inherited Members

13.85.1 Detailed Description

Base class for polynomial chaos expansions (PCE) and stochastic collocation (SC)

The NonDExpansion class provides a base class for methods that use polynomial expansions to approximate the effect of parameter uncertainties on response functions of interest.

13.85.2 Member Function Documentation

const Model & algorithm_space_model ( ) const `[inline], [virtual]`

default definition that gets redefined in selected derived Minimizers

Reimplemented from Analyzer.

References NonDExpansion::uSpaceModel.

void increment_specification_sequence ( ) `[protected], [virtual]`

increment the input specification sequence (PCE only)

default implementation is overridden by PCE

Reimplemented in NonDPolynomialChaos.

References NonDIntegration::increment_specification_sequence(), Iterator::iterator_rep(), Model::subordinate_iterator(), and NonDExpansion::uSpaceModel.

Referenced by NonDPolynomialChaos::increment_specification_sequence(), and NonDExpansion::quantify_uncertainty().

Real compute_covariance_metric ( ) `[protected], [virtual]`

compute 2-norm of change in response covariance

computes the default refinement metric based on change in respCovariance

Reimplemented in NonDStochCollocation.

References NonDExpansion::compute_covariance(), NonDExpansion::covarianceControl, NonDExpansion::respCovariance, and NonDExpansion::respVariance.

Referenced by NonDStochCollocation::compute_covariance_metric(), NonDExpansion::increment_sets(), and NonDExpansion::refine_expansion().

Real compute_final_statistics_metric ( ) [protected], [virtual]

compute 2-norm of change in final statistics

computes a "goal-oriented" refinement metric employing finalStatistics

Reimplemented in NonDStochCollocation.

References NonDExpansion::compute_statistics(), NonD::finalStatistics, Response::function_values(), Analyzer::numFunctions, NonD::requestedGenRelLevels, NonD::requestedProbLevels, NonD::requestedRelLevels, and NonD::requestedRespLevels.

Referenced by NonDStochCollocation::compute_final_statistics_metric(), and NonDExpansion::increment_sets().

void compute_statistics ( ) [protected]

calculate analytic and numerical statistics from the expansion

Calculate analytic and numerical statistics from the expansion and log results within final_stats for use in OUU.

References Dakota::abort_handler(), ResultsManager::active(), Iterator::active_set(), Response::active_set_derivative_vector(), Response::active_set_request_vector(), Iterator::all_responses(), Iterator::all_samples(), Model::approximation_data(), Model::approximations(), NonD::archive_allocate_mappings(), NonDExpansion::archive_coefficients(), NonD::archive_from_resp(), NonDExpansion::archive_moments(), NonD::archive_to_resp(), NonD::cdfFlag, PecosApproximation::compute_component_effects(), PecosApproximation::compute_moments(), NonDExpansion::compute_off_diagonal_covariance(), PecosApproximation::compute_total_effects(), NonD::computedGenRelLevels, NonD::computedProbLevels, NonD::computedRelLevels, NonD::computedRespLevels, Model::continuous_variable_ids(), Model::continuous_variable_labels(), Model::continuous_variables(), Dakota::copy_data(), NonDExpansion::covarianceControl, Model::current_variables(), PecosApproximation::expansion_coefficient_flag(), NonDExpansion::expansionSampler, NonDExpansion::expGradsMeanX, NonDExpansion::expSampling, NonDAdaptImpSampling::final_probability(), NonD::finalStatistics, Response::function_gradient(), Response::function_value(), Response::function_values(), NonDExpansion::importanceSampler, NonDExpansion::impSampling, Iterator::initial_points(), NonDAdaptImpSampling::initialize(), NonD::initialize_distribution_mappings(), NonD::initialize_random_variables(), NonDExpansion::initialPtU, ResultsManager::insert(), Iterator::iteratedModel, Iterator::iterator_rep(), PecosApproximation::mean_gradient(), Iterator::methodPCIter, NonD::miPLIndex, PecosApproximation::moments(), NonD::nataf-Transform, NonD::numContDesVars, NonD::numContEpistUncVars, Analyzer::numContinuousVars, NonD::numContStateVars, Analyzer::numFunctions, NonDExpansion::numSamplesOnExpansion, Iterator::outputLevel, ActiveSet::request_vector(), NonD::requestedGenRelLevels, NonD::requestedProbLevels, NonD::requestedRelLevels, NonD::requestedRespLevels, NonDExpansion::respCovariance, NonD::respLevelTarget, Model::response_labels(), Iterator::response_results(), NonDExpansion::respVariance, Iterator::resultsDB, Iterator::resultsNames, Iterator::run(), Iterator::run_identifier(), Iterator::subIteratorFlag, NonD::totalLevelRequests, NonDSampling::update_final_statistics(), NonDExpansion::uSpaceModel, PecosApproximation::variance_gradient(), and NonDExpansion::vbdFlag.

Referenced by NonDStochCollocation::compute_final_statistics_metric(), NonDExpansion::compute_final_statistics_metric(), NonDExpansion::compute_print_converged_results(), NonDExpansion::compute_print_increment_results(), and NonDExpansion::compute_print_iteration_results().

13.85.3 Member Data Documentation

bool useDerivs [protected]

flag for `use_derivatives` specification, indicating usage of derivative data (with respect to expansion variables) to enhance the calculation of the stochastic expansion.

This is part of the method specification since the instantiation of the global data fit surrogate is implicit with no user specification. This behavior is distinct from the usage of response derivatives with respect to auxilliary

variables (design, epistemic) for computing derivatives of aleatory expansion statistics with respect to these variables.

Referenced by NonDExpansion::compute_expansion(), NonDStochCollocation::initialize_u_space_model(), NonDPolynomialChaos::initialize_u_space_model(), NonDStochCollocation::resolve_inputs(), NonDPolynomialChaos::resolve_inputs(), NonDPolynomialChaos::terms_ratio_to_samples(), and NonDPolynomialChaos::terms_samples_to_ratio().

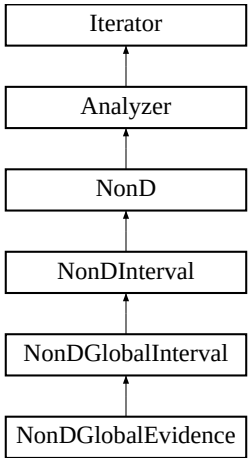
The documentation for this class was generated from the following files:

- NonDExpansion.hpp
- NonDExpansion.cpp

13.86 NonDGlobalEvidence Class Reference

Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.

Inheritance diagram for NonDGlobalEvidence:



Public Member Functions

- NonDGlobalEvidence (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDGlobalEvidence ()

  destructor
- void initialize ()

  perform any required initialization
- void set_cell_bounds ()

  set the optimization variable bounds for each cell
- void get_best_sample (bool maximize, bool eval_approx)

  determine truthFnStar and approxFnStar
- void post_process_cell_results (bool maximize)

  post-process a cell minimization/maximization result
- void post_process_response_fn_results ()

post-process the interval computed for a response function

- void post_process_final_results ()

  perform final post-processing

Additional Inherited Members

13.86.1 Detailed Description

Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.

The NonDEvidence class implements the propagation of epistemic uncertainty using Dempster-Shafer theory of evidence. In this approach, one assigns a set of basic probability assignments (BPA) to intervals defined for the uncertain variables. Input interval combinations are calculated, along with their BPA. Currently, the response function is evaluated at a set of sample points, then a response surface is constructed which is sampled extensively to find the minimum and maximum within each input interval cell, corresponding to the belief and plausibility within that cell, respectively. This data is then aggregated to calculate cumulative distribution functions for belief and plausibility.

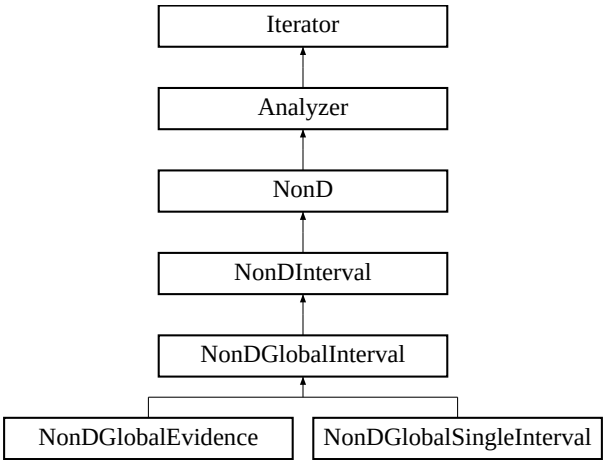
The documentation for this class was generated from the following files:

- NonDGlobalEvidence.hpp
- NonDGlobalEvidence.cpp

13.87 NonDGlobalInterval Class Reference

Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.

Inheritance diagram for NonDGlobalInterval:



Public Member Functions

- NonDGlobalInterval (ProblemDescDB &problem_db, Model &model)

  constructor

- ~NonDGlobalInterval ()

destructor

- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void quantify_uncertainty ()

  Performs an optimization to determine interval bounds for an entire function or interval bounds on a particular statistical estimator.
- const Model & algorithm_space_model () const

Protected Member Functions

- virtual void initialize ()

  perform any required initialization
- virtual void set_cell_bounds ()

  set the optimization variable bounds for each cell
- virtual void get_best_sample (bool maximize, bool eval_approx)

  determine truthFnStar and approxFnStar
- virtual void post_process_cell_results (bool maximize)

  post-process a cell minimization/maximization result
- virtual void post_process_response_fn_results ()

  post-process the interval computed for a response function
- virtual void post_process_final_results ()

  perform final post-processing
- void post_process_run_results (bool maximize)

  post-process an optimization execution: output results, update convergence controls, and update GP approximation
- void evaluate_response_star_truth ()

  evaluate the truth response at the optimal variables solution and update the GP with the new data

Protected Attributes

- Iterator daceIterator

  LHS iterator for constructing initial GP for all response functions.
- Model fHatModel

  GP model of response, one approximation per response function.
- Iterator intervalOptimizer

  optimizer for solving surrogate-based subproblem: NCSU DIRECT optimizer for maximizing expected improvement or mixed EA if discrete variables.
- Model intervalOptModel

  recast model which formulates the surrogate-based optimization subproblem (recasts as design problem; may assimilate mean and variance to enable max(expected improvement))
- Real approxFnStar

  approximate response corresponding to minimum/maximum truth response
- Real truthFnStar

  minimum/maximum truth response function value

Static Private Member Functions

- static void EIF_objective_min (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used as the objective function in the Expected Improvement Function (EIF) for minimizing the GP
- static void EIF_objective_max (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used as the objective function in the Expected Improvement Function (EIF) for maximizing the GP
- static void extract_objective (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used to extract the active objective function when optimizing for an interval lower or upper bound (non-EIF formulations). The sense of the optimization is set separately.

Private Attributes

- const int seedSpec

  the user seed specification (default is 0)
- int numSamples

  the number of samples used in the surrogate
- String rngName

  name of the random number generator
- bool gpModelFlag

  flag indicating use of GP surrogate emulation
- bool eifFlag

  flag indicating use of maximized expected improvement for GP iterate selection
- unsigned short improvementConvergeCntr

  counter for number of successive iterations that the iteration improvement is less than the convergenceTol
- unsigned short improvementConvergeLimit

  counter for number of successive iterations that the iteration improvement is less than the convergenceTol
- Real distanceTol

  tolerance for L_2 change in optimal solution
- unsigned short distanceConvergeCntr

  counter for number of successive iterations that the L_2 change in optimal solution is less than the convergenceTol
- unsigned short distanceConvergeLimit

  counter for number of successive iterations that the L_2 change in optimal solution is less than the convergenceTol
- RealVector prevCVStar

  stores previous optimal point for continuous variables; used for assessing convergence
- IntVector prevDIVStar

  stores previous optimal point for discrete integer variables; used for assessing convergence
- RealVector prevDRVStar

  stores previous optimal point for discrete real variables; used for assessing convergence
- Real prevFnStar

  stores previous solution value for assessing convergence
- size_t sbIterNum

  surrogate-based minimization/maximization iteration count

- bool boundConverged

  flag indicating convergence of a minimization or maximization cycle

- bool allResponsesPerIter

  flag for maximal response extraction (all response values obtained on each function call)

- short dataOrder

  order of the data used for surrogate construction, in ActiveSet request vector 3-bit format; user may override responses spec

Static Private Attributes

- static NonDGlobalInterval ∗ nondGIInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.87.1 Detailed Description

Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.

The NonDGlobalInterval class supports global nongradient-based optimization apporaches to determining interval bounds for epistemic UQ. The interval bounds may be on the entire function in the case of pure interval analysis (e.g. intervals on input = intervals on output), or the intervals may be on statistics of an "inner loop" aleatory analysis such as intervals on means, variances, or percentile levels. The preliminary implementation will use a Gaussian process surrogate to determine interval bounds.

13.87.2 Member Function Documentation

const Model & algorithm_space_model ( ) const `[inline], [virtual]`

default definition that gets redefined in selected derived Minimizers

Reimplemented from Analyzer.

References NonDGlobalInterval::fHatModel.

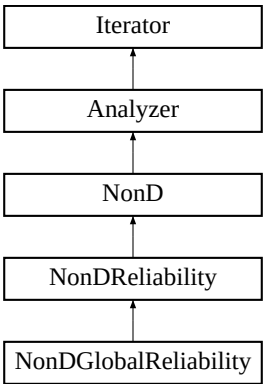
The documentation for this class was generated from the following files:

- NonDGlobalInterval.hpp
- NonDGlobalInterval.cpp

13.88 NonDGlobalReliability Class Reference

Class for global reliability methods within DAKOTA/UQ.

Inheritance diagram for NonDGlobalReliability:



Public Member Functions

- NonDGlobalReliability (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDGlobalReliability ()

  destructor
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void quantify_uncertainty ()

  performs an uncertainty propagation using analytical reliability methods which solve constrained optimization problems to obtain approximations of the cumulative distribution function of response
- void print_results (std::ostream &s)

  print the approximate mean, standard deviation, and importance factors when using the mean value method or the CDF/CCDF information when using MPP-search-based reliability methods

Private Member Functions

- void optimize_gaussian_process ()

  construct the GP using EGO/SKO
- void importance_sampling ()

  perform multimodal adaptive importance sampling on the GP
- void get_best_sample ()

  determine current best solution from among sample data for expected imporovement function in Performance Measure Approach (PMA)
- Real constraint_penalty (const Real &constraint, const RealVector &c_variables)

  calculate the penalty to be applied to the PMA constraint value
- Real expected_improvement (const RealVector &expected_values, const Variables &recast_vars)

  expected improvement function for the GP
- Real expected_feasibility (const RealVector &expected_values, const Variables &recast_vars)

  expected feasibility function for the GP

Static Private Member Functions

- static void EIF_objective_eval (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used as the objective function in the Expected Improvement (EIF) problem formulation for PMA
- static void EFF_objective_eval (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used as the objective function in the Expected Feasibility (EFF) problem formulation for RIA

Private Attributes

- Real fnStar

  minimum penalized response from among true function evaluations
- short meritFunctionType

  type of merit function used to penalize sample data
- Real lagrangeMult

  Lagrange multiplier for standard Lagrangian merit function.
- Real augLagrangeMult

  Lagrange multiplier for augmented Lagrangian merit function.
- Real penaltyParameter

  penalty parameter for augmented Lagrangian merit funciton
- Real lastConstraintViolation

  constraint violation at last iteration, used to determine if the current iterate should be accepted (must reduce violation)
- bool lastIterateAccepted

  flag to determine if last iterate was accepted this controls update of parameters for augmented Lagrangian merit fn
- short dataOrder

  order of the data used for surrogate construction, in ActiveSet request vector 3-bit format; user may override responses spec

Static Private Attributes

- static NonDGlobalReliability ∗ nondGlobRelInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.88.1 Detailed Description

Class for global reliability methods within DAKOTA/UQ.

The NonDGlobalReliability class implements EGO/SKO for global MPP search, which maximizes an expected improvement function derived from Gaussian process models. Once the limit state has been characterized, a multimodal importance sampling approach is used to compute probabilities.

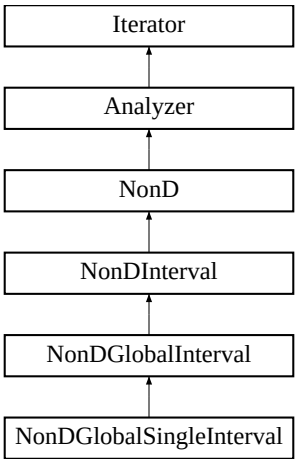
The documentation for this class was generated from the following files:

- NonDGlobalReliability.hpp
- NonDGlobalReliability.cpp

13.89 NonDGlobalSingleInterval Class Reference

Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.

Inheritance diagram for NonDGlobalSingleInterval:



Public Member Functions

- NonDGlobalSingleInterval (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDGlobalSingleInterval ()

  destructor

Protected Member Functions

- void initialize ()

  perform any required initialization
- void post_process_cell_results (bool maximize)

  post-process a cell minimization/maximization result
- void get_best_sample (bool maximize, bool eval_approx)

  determine truthFnStar and approxFnStar

Private Attributes

- size_t statCntr

  counter for finalStatistics

Additional Inherited Members

13.89.1 Detailed Description

Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.

The NonDGlobalSingleInterval class supports global nongradient-based optimization apporaches to determining interval bounds for epistemic UQ. The interval bounds may be on the entire function in the case of pure interval analysis (e.g. intervals on input = intervals on output), or the intervals may be on statistics of an "inner loop" aleatory analysis such as intervals on means, variances, or percentile levels. The preliminary implementation will use a Gaussian process surrogate to determine interval bounds.

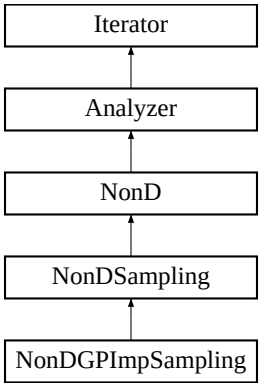
The documentation for this class was generated from the following files:

- NonDGlobalSingleInterval.hpp
- NonDGlobalSingleInterval.cpp

13.90 NonDGPImpSampling Class Reference

Class for the Gaussian Process-based Importance Sampling method.

Inheritance diagram for NonDGPImpSampling:



Public Member Functions

- NonDGPImpSampling (ProblemDescDB &problem_db, Model &model)

  standard constructor
- ~NonDGPImpSampling ()

  destructor
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void quantify_uncertainty ()

  perform the GP importance sampling and return probability of failure.
- void print_results (std::ostream &s)

  print the final statistics
- Real final_probability ()

  returns the probability calculated by the importance sampling

Private Member Functions

- RealVector calcExpIndicator (const int respFnCount, const Real respThresh)

  function to calculate the expected indicator probabilities
- Real calcExpIndPoint (const int respFnCount, const Real respThresh, const RealVector this_mean, const RealVector this_var)

  function to calculate the expected indicator probabilities for one point
- void calcRhoDraw ()

  function to update the rhoDraw data, adding x values and rho draw values
- RealVector drawNewX (int this_k)

  function to pick the next X value to be evaluated by the Iterated model

Private Attributes

- Iterator gpBuild

  LHS iterator for building the initial GP.
- Iterator gpEval

  LHS iterator for sampling on the GP.
- Model gpModel

  GP model of response, one approximation per response function.
- Iterator sampleRhoOne

  LHS iterator for sampling from the rhoOneDistribution.
- int numPtsAdd

  the number of points added to the original set of LHS samples
- int numPtsTotal

  the total number of points
- int numEmulEval

  the number of points evaluated by the GP each iteration
- Real finalProb

  the final calculated probability (p)
- RealVectorArray gpCvars

  Vector to hold the current values of the current sample inputs on the GP.
- RealVectorArray gpMeans

  Vector to hold the current values of the current mean estimates for the sample values on the GP.
- RealVectorArray gpVar

  Vector to hold the current values of the current variance estimates for the sample values on the GP.
- RealVector expIndicator

  Vector to hold the expected indicator values for the current GP samples.
- RealVector rhoDraw

  Vector to hold the rhoDraw values for the current GP samples.
- RealVector normConst

  Vector to hold the normalization constant calculated for each point added.
- RealVector indicator

  IntVector to hold indicator for actual simulation values vs. threshold.

- RealVectorArray xDrawThis

  xDrawThis, appended to locally to hold the X values of emulator points chosen
- RealVector expIndThis

  expIndThis, appended locally to hold the expected indicator
- RealVector rhoDrawThis

  rhoDrawThis, appended locally to hold the rhoDraw density for calculating draws
- RealVector rhoMix

  rhoMix, mixture density
- RealVector rhoOne

  rhoOne, original importance density

Additional Inherited Members

13.90.1 Detailed Description

Class for the Gaussian Process-based Importance Sampling method.

The NonDGPImpSampling implements a method developed by Keith Dalbey that uses a Gaussian process surrogate in the calculation of the importance density. Specifically, the mean and variance of the GP prediction are used to calculate an expected value that a particular point fails, and that is used as part of the computation of the "draw distribution." The normalization constants and the mixture distribution used are defined in (need to get SAND report).

13.90.2 Constructor & Destructor Documentation

NonDGPImpSampling (ProblemDescDB & *problem_db*, Model & *model*)

standard constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

References Iterator::assign_rep(), Model::assign_rep(), NonD::construct_lhs(), ProblemDescDB::get_bool(), ProblemDescDB::get_int(), ProblemDescDB::get_string(), NonDGPImpSampling::gpBuild, NonDGPImpSampling::gpEval, NonDGPImpSampling::gpModel, Model::gradient_type(), Model::hessian_type(), Iterator::iteratedModel, Iterator::maxIterations, NonDGPImpSampling::numEmulEval, NonDGPImpSampling::numPtsAdd, NonDSampling::numSamples, Iterator::outputLevel, Iterator::probDescDB, NonDSampling::randomSeed, NonDSampling::rngName, NonDGPImpSampling::sampleRhoOne, NonDSampling::samplingVarsMode, NonDSampling::statsFlag, NonDSampling::vary_pattern(), and NonDSampling::varyPattern.

13.90.3 Member Function Documentation

void quantify_uncertainty ( ) `[virtual]`

perform the GP importance sampling and return probability of failure.

Calculate the failure probabilities for specified probability levels using Gaussian process based importance sampling.

Implements NonD.

References Model::acv(), Iterator::all_responses(), Analyzer::all_samples(), Iterator::all_samples(), Model::append_approximation(), Model::approximation_data(), Model::approximation_variances(), Model::build_approximation(), NonDGPImpSampling::calcExpIndicator(), NonDGPImpSampling::calcExpIndPoint(), NonDGPImpSampling::calcRhoDraw(), NonD::cdfFlag, Model::compute_response(), NonD::computedProbLevels, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), Model::current_response(),

Model::current_variables(), NonDGPImpSampling::drawNewX(), Model::evaluation_id(), NonDGPImpSampling::expIndicator, NonDGPImpSampling::expIndThis, NonDGPImpSampling::finalProb, Response::function_values(), NonDGPImpSampling::gpCvars, NonDGPImpSampling::gpEval, NonDGPImpSampling::gpMeans, NonDGPImpSampling::gpModel, NonDGPImpSampling::gpVar, NonDGPImpSampling::indicator, NonD::initialize_distribution_mappings(), Iterator::iteratedModel, Iterator::methodPCIter, NonD::miPLIndex, NonDGPImpSampling::normConst, NonDGPImpSampling::numEmulEval, Analyzer::numFunctions, NonDGPImpSampling::numPtsAdd, NonDGPImpSampling::numPtsTotal, NonDSampling::numSamples, Iterator::outputLevel, Model::pop_approximation(), NonD::requestedRespLevels, NonDGPImpSampling::rhoDraw, NonDGPImpSampling::rhoDrawThis, NonDGPImpSampling::rhoMix, NonDGPImpSampling::rhoOne, Iterator::run(), NonDGPImpSampling::sampleRhoOne, and NonDGPImpSampling::xDrawThis.

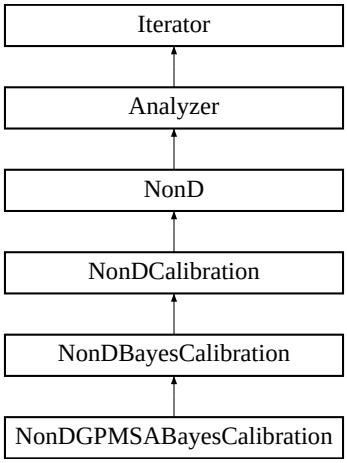
The documentation for this class was generated from the following files:

- NonDGPImpSampling.hpp
- NonDGPImpSampling.cpp

13.91 NonDGPMSABayesCalibration Class Reference

Generates posterior distribution on model parameters given experiment data.

Inheritance diagram for NonDGPMSABayesCalibration:



Public Member Functions

- NonDGPMSABayesCalibration (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDGPMSABayesCalibration ()

  destructor

Public Attributes

- String rejectionType

  Rejection type (standard or delayed, in the DRAM framework)
- String metropolisType

Metropolis type (hastings or adaptive, in the DRAM framework)

- int numSamples

  number of samples in the chain (e.g. number of MCMC samples)
- int emulatorSamples

  number of samples of the simulation to construct the GP
- RealVector proposalCovScale

  scale factor for proposal covariance
- Real likelihoodScale

  scale factor for likelihood
- bool calibrateSigmaFlag

  flag to indicated if the sigma terms should be calibrated (default true)
- String approxImportFile

  name of file from which to import build points to build GP
- bool approxImportAnnotated

  annotate flag
- bool approxImportActiveOnly

  import active variables only

Protected Member Functions

- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void quantify_uncertainty ()

  performs a forward uncertainty propagation by using GPM/SA to generate a posterior distribution on parameters given a set of simulation parameter/response data, a set of experimental data, and additional variables to be specified here.

Protected Attributes

- int randomSeed

  print the final statistics

Private Attributes

- Iterator lhsIter

  LHS iterator for generating samples for GP.

Static Private Attributes

- static NonDGPMSABayesCalibration * NonDGPMSAInstance

  Pointer to current class instance for use in static callback functions.

Additional Inherited Members

13.91.1 Detailed Description

Generates posterior distribution on model parameters given experiment data.

This class provides a wrapper for the functionality provided in the Los Alamos National Laboratory code called GPM/SA (Gaussian Process Models for Simulation Analysis). Although this is a code that provides input/output mapping, it DOES NOT provide the mapping that we usually think of in the NonDeterministic class hierarchy in DAKOTA, where uncertainty in parameter inputs are mapped to uncertainty in simulation responses. Instead, this class takes a pre-existing set of simulation data as well as experimental data, and maps priors on input parameters to posterior distributions on those input parameters, according to a likelihood function. The goal of the MCMC sampling is to produce posterior values of parameter estimates which will produce simulation response values that "match well" to the experimental data. The MCMC is an integral part of the calibration. The data structures in GPM/SA are fairly detailed and nested. Part of this prototyping exercise is to determine what data structures need to be specified and initialized in DAKOTA and sent to GPM/SA, and what data structures will be returned.

13.91.2 Constructor & Destructor Documentation

NonDGPMSABayesCalibration (ProblemDescDB & *problem_db,* Model & *model*)

constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

References Iterator::assign_rep(), NonDGPMSABayesCalibration::emulatorSamples, ProblemDescDB::get_string(), Iterator::iteratedModel, NonDGPMSABayesCalibration::lhsIter, Iterator::probDescDB, and NonDGPMSABayesCalibration::randomSeed.

13.91.3 Member Function Documentation

void quantify_uncertainty ( ) `[protected], [virtual]`

performs a forward uncertainty propagation by using GPM/SA to generate a posterior distribution on parameters given a set of simulation parameter/response data, a set of experimental data, and additional variables to be specified here.

Perform the uncertainty quantification

Reimplemented from NonDBayesCalibration.

References Iterator::all_responses(), Analyzer::all_samples(), Iterator::all_samples(), NonDGPMSABayesCalibration::approxImportAnnotated, NonDGPMSABayesCalibration::approxImportFile, NonDGPMSABayesCalibration::calibrateSigmaFlag, ExperimentData::config_vars(), Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), NonDGPMSABayesCalibration::emulatorSamples, NonDCalibration::expData, NonDCalibration::expDataFileAnnotated, NonDCalibration::expDataFileName, Iterator::iteratedModel, NonDGPMSABayesCalibration::lhsIter, ExperimentData::load_scalar(), Iterator::methodPCIter, NonDGPMSABayesCalibration::metropolisType, NonD::miPLIndex, NonDGPMSABayesCalibration::NonDGPMSAInstance, NonDCalibration::numExpConfigVars, NonDCalibration::numExperiments, NonDCalibration::numExpStdDeviationsRead, Analyzer::numFunctions, NonDGPMSABayesCalibration::numSamples, NonD::numUncertainVars, Iterator::outputLevel, NonDGPMSABayesCalibration::rejectionType, Iterator::run(), and ExperimentData::scalar_data().

13.91.4 Member Data Documentation

int randomSeed `[protected]`

print the final statistics

random seed to pass to QUESO

Referenced by NonDGPMSABayesCalibration::NonDGPMSABayesCalibration().

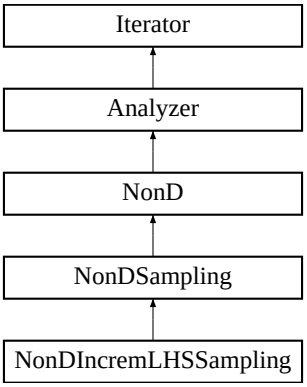
The documentation for this class was generated from the following files:

- NonDGPMSABayesCalibration.hpp
- NonDGPMSABayesCalibration.cpp

13.92 NonDIncremLHSSampling Class Reference

Performs icremental LHS sampling for uncertainty quantification.

Inheritance diagram for NonDIncremLHSSampling:



Public Member Functions

- NonDIncremLHSSampling (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDIncremLHSSampling ()

  destructor
- void quantify_uncertainty ()

  performs a forward uncertainty propagation by using LHS to generate a set of parameter samples, performing function evaluations on these parameter samples, and computing statistics on the ensemble of results.
- void print_results (std::ostream &s)

  print the final statistics

Static Protected Member Functions

- static bool rank_sort (const int &x, const int &y)

  sort algorithm to compute ranks for rank correlations

Private Attributes

- int previousSamples

  number of samples in previous LHS run
- bool varBasedDecompFlag

  flags computation of VBD

Static Private Attributes

- static RealArray rawData

  static data used by static rank_sort() fn

Additional Inherited Members

13.92.1 Detailed Description

Performs icremental LHS sampling for uncertainty quantification.

The Latin Hypercube Sampling (LHS) package from Sandia Albuquerque's Risk and Reliability organization provides comprehensive capabilities for Monte Carlo and Latin Hypercube sampling within a broad array of user-specified probabilistic parameter distributions. The icremental LHS sampling capability allows one to supplement an initial sample of size n to size 2n while maintaining the correct stratification of the 2n samples and also maintaining the specified correlation structure. The icremental version of LHS will return a sample of size n, which when combined with the original sample of size n, allows one to double the size of the sample.

13.92.2 Constructor & Destructor Documentation

NonDIncremLHSSampling (ProblemDescDB & *problem_db,* Model & *model*)

constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

13.92.3 Member Function Documentation

void quantify_uncertainty ( ) `[virtual]`

performs a forward uncertainty propagation by using LHS to generate a set of parameter samples, performing function evaluations on these parameter samples, and computing statistics on the ensemble of results.

Generate incremental samples. Loop over the set of samples and compute responses. Compute statistics on the set of responses if statsFlag is set.

Implements NonD.

References Dakota::abort_handler(), Model::aleatory_distribution_parameters(), Analyzer::allResponses, Analyzer::allSamples, NonDSampling::compute_statistics(), Dakota::copy_data(), Dakota::data_pairs, Analyzer::evaluate_parameter_sets(), NonDSampling::get_parameter_sets(), Iterator::iteratedModel, NonD::numBetaVars, NonD::numBinomialVars, Analyzer::numContinuousVars, NonD::numExponentialVars, NonD::numFrechetVars, NonD::numGammaVars, NonD::numGeometricVars, NonD::numGumbelVars, NonD::numHistogramBinVars, NonD::numHistogramPtIntVars, NonD::numHistogramPtRealVars, NonD::numHistogramPtStringVars, NonD::numHypergeomVars, NonD::numLognormalVars, NonD::numLoguniformVars, NonD::numNegBinomialVars, NonD::numNormalVars, NonD::numPoissonVars, NonDSampling::numSamples, NonD::numTriangularVars, NonD::numUniformVars, NonD::numWeibullVars, NonDIncremLHSSampling::previousSamples, NonDIncremLHSSampling-

::rank_sort(), NonDIncremLHSSampling::rawData, NonDSampling::sampleRanks, NonDSampling::sampleRanksMode, NonDSampling::samplesRef, NonDSampling::sampleType, Iterator::submethod_enum_to_string(), and NonDSampling::varyPattern.

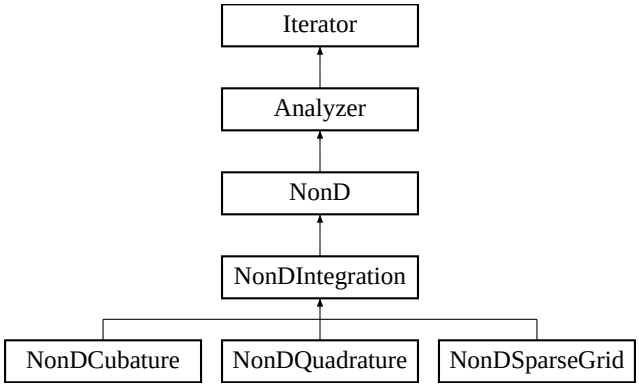
The documentation for this class was generated from the following files:

- NonDIncremLHSSampling.hpp
- NonDIncremLHSSampling.cpp

13.93 NonDIntegration Class Reference

Derived nondeterministic class that generates N-dimensional numerical integration points for evaluation of expectation integrals.

Inheritance diagram for NonDIntegration:



Public Member Functions

- virtual void initialize_grid (const std::vector< Pecos::BasisPolynomial > &poly_basis)=0

  initialize integration grid by drawing from polynomial basis settings
- virtual void increment_grid ()=0

  increment SSG level/TPQ order
- virtual void increment_grid_preference (const RealVector &dim_pref)

  increment SSG level/TPQ order and update anisotropy
- virtual void increment_grid_weights (const RealVector &aniso_wts)

  increment SSG level/TPQ order and update anisotropy
- virtual void increment_specification_sequence ()

  increment sequenceIndex and update active orders/levels
- const Pecos::IntegrationDriver & driver () const

  return numIntDriver

Static Public Member Functions

- static void dimension_preference_to_anisotropic_order (unsigned short scalar_order_spec, const RealVector &dim_pref_spec, size_t num_v, UShortArray &aniso_order)

convert scalar_order_spec and vector dim_pref_spec to vector aniso_order

- static void anisotropic_order_to_dimension_preference (const UShortArray &aniso_order, unsigned short &scalar_order, RealVector &dim_pref)

  convert vector aniso_order to scalar_order and vector dim_pref

Protected Member Functions

- NonDIntegration (ProblemDescDB &problem_db, Model &model)

  constructor

- NonDIntegration (unsigned short method_name, Model &model)

  alternate constructor for instantiations "on the fly"

- NonDIntegration (unsigned short method_name, Model &model, const RealVector &dim_pref)

  alternate constructor for instantiations "on the fly"

- ~NonDIntegration ()

  destructor

- void quantify_uncertainty ()

  Mapping of the core_run() virtual function for the NonD branch.

- void check_variables (const Pecos::ShortArray &x_types)

  verify self-consistency of variables data

- void print_points_weights (const String &tabular_name)

  output integration points and weights to a tabular file

Protected Attributes

- Pecos::IntegrationDriver numIntDriver

  Pecos utlity class for managing interface to tensor-product grids and VPISparseGrid utilities for Smolyak sparse grids and cubature.

- size_t numIntegrations

  counter for number of integration executions for this object

- size_t sequenceIndex

  index into NonDQuadrature::quadOrderSpec and NonDSparseGrid::ssgLevelSpec that defines the current instance of several possible refinement levels

- RealVector dimPrefSpec

  the user specification for anisotropic dimension preference

Additional Inherited Members

13.93.1 Detailed Description

Derived nondeterministic class that generates N-dimensional numerical integration points for evaluation of expectation integrals.

This class provides a base class for shared code among NonDQuadrature and NonDSparseGrid.

13.93.2 Constructor & Destructor Documentation

NonDIntegration (ProblemDescDB & *problem_db,* Model & *model*) [protected]

constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification. It is not currently used, as there are not yet separate nond_quadrature/nond_sparse_grid method specifications.

References Dakota::abort_handler(), NonD::initialize_final_statistics(), NonD::initialize_random_variable_correlations(), NonD::initialize_random_variable_transformation(), NonD::initialize_random_variable_types(), Analyzer::numDiscreteIntVars, Analyzer::numDiscreteRealVars, Analyzer::numDiscreteStringVars, and NonD::verify_correlation_support().

NonDIntegration (unsigned short *method_name,* Model & *model*) [protected]

alternate constructor for instantiations "on the fly"

This alternate constructor is used for on-the-fly generation and evaluation of numerical integration points.

NonDIntegration (unsigned short *method_name,* Model & *model,* const RealVector & *dim_pref*) [protected]

alternate constructor for instantiations "on the fly"

This alternate constructor is used for on-the-fly generation and evaluation of numerical integration points.

13.93.3 Member Function Documentation

void dimension_preference_to_anisotropic_order (unsigned short *scalar_order_spec,* const RealVector & *dim_pref_spec,* size_t *num_v,* UShortArray & *aniso_order*) [static]

convert scalar_order_spec and vector dim_pref_spec to vector aniso_order

Converts a scalar order specification and a vector anisotropic dimension preference into an anisotropic order vector. It is used for initialization and does not enforce a reference lower bound (see also NonDQuadrature::update_anisotropic_order()).

Referenced by NonDPolynomialChaos::increment_specification_sequence(), NonDQuadrature::initialize_dimension_quadrature_order(), and NonDPolynomialChaos::NonDPolynomialChaos().

void anisotropic_order_to_dimension_preference (const UShortArray & *aniso_order,* unsigned short & *scalar_order,* RealVector & *dim_pref*) [static]

convert vector aniso_order to scalar_order and vector dim_pref

Converts a vector anisotropic order into a scalar order and vector anisotropic dimension preference.

Referenced by NonDPolynomialChaos::NonDPolynomialChaos().

void check_variables (const Pecos::ShortArray & *x_types*) [protected]

verify self-consistency of variables data

Virtual function called from probDescDB-based constructors and from NonDIntegration::quantify_uncertainty()

References Dakota::abort_handler(), NonD::numContAleatUncVars, NonD::numContDesVars, NonD::numContEpistUncVars, Analyzer::numContinuousVars, and NonD::numContStateVars.

Referenced by NonDCubature::NonDCubature(), NonDQuadrature::NonDQuadrature(), NonDSparseGrid::NonDSparseGrid(), and NonDIntegration::quantify_uncertainty().

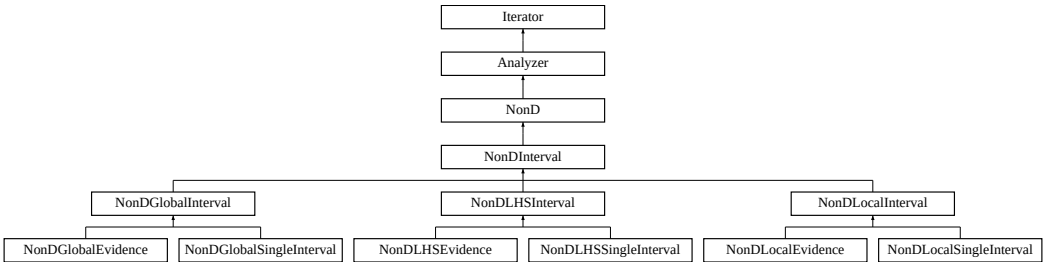
The documentation for this class was generated from the following files:

- NonDIntegration.hpp
- NonDIntegration.cpp

13.94 NonDInterval Class Reference

Base class for interval-based methods within DAKOTA/UQ.

Inheritance diagram for NonDInterval:



Public Member Functions

- NonDInterval (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDInterval ()

  destructor
- void print_results (std::ostream &s)

  print the cumulative distribution functions for belief and plausibility

Protected Member Functions

- void initialize_final_statistics ()

  initialize finalStatistics for belief/plausibility results sets
- void compute_evidence_statistics ()

  method for computing belief and plausibility values for response levels or vice-versa
- void calculate_cells_and_bpas ()

  computes the interval combinations (cells) and their bpas replaces CBPIIC_F77 from wrapper calculate_basic_-prob_intervals()
- void calculate_cbf_cpf (bool complementary=true)

  function to compute (complementary) distribution functions on belief and plausibility replaces CCBFPF_F77 from wrapper calculate_cum_belief_plaus()

Protected Attributes

- bool singleIntervalFlag

  flag for SingleInterval derived class
- RealVectorArray ccBelFn

  Storage array to hold CCBF values.

- RealVectorArray ccPlausFn

  Storage array to hold CCPF values.
- RealVectorArray ccBelVal

  Storage array to hold CCB response values.
- RealVectorArray ccPlausVal

  Storage array to hold CCP response values.
- RealVectorArray cellContLowerBounds

  Storage array to hold cell lower bounds for continuous variables.
- RealVectorArray cellContUpperBounds

  Storage array to hold cell upper bounds for continuous variables.
- IntVectorArray cellIntRangeLowerBounds

  Storage array to hold cell lower bounds for discrete int range variables.
- IntVectorArray cellIntRangeUpperBounds

  Storage array to hold cell upper bounds for discrete int range variables.
- IntVectorArray cellIntSetBounds

  Storage array to hold cell values for discrete integer set variables.
- IntVectorArray cellRealSetBounds

  Storage array to hold cell value for discrete real set variables.
- RealVectorArray cellFnLowerBounds

  Storage array to hold cell min.
- RealVectorArray cellFnUpperBounds

  Storage array to hold cell max.
- RealVector cellBPA

  Storage array to hold cell bpa.
- size_t respFnCntr

  response function counter
- size_t cellCntr

  cell counter
- size_t numCells

  total number of interval combinations

Additional Inherited Members

13.94.1 Detailed Description

Base class for interval-based methods within DAKOTA/UQ.

The NonDInterval class implements the propagation of epistemic uncertainty using either pure interval propagation or Dempster-Shafer theory of evidence. In the latter approach, one assigns a set of basic probability assignments (BPA) to intervals defined for the uncertain variables. Input interval combinations are calculated, along with their BPA. Currently, the response function is evaluated at a set of sample points, then a response surface is constructed which is sampled extensively to find the minimum and maximum within each input interval cell, corresponding to the belief and plausibility within that cell, respectively. This data is then aggregated to calculate cumulative distribution functions for belief and plausibility.

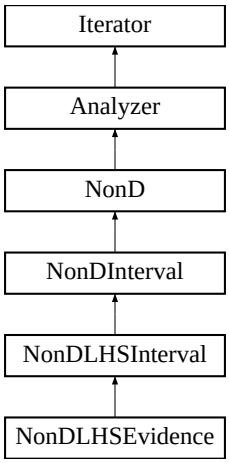
The documentation for this class was generated from the following files:

- NonDInterval.hpp
- NonDInterval.cpp

13.95 NonDLHSEvidence Class Reference

Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.

Inheritance diagram for NonDLHSEvidence:



Public Member Functions

- NonDLHSEvidence (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDLHSEvidence ()

  destructor
- void initialize ()

  perform any required initialization
- void post_process_samples ()

  post-process the output from executing lhsSampler

Additional Inherited Members

13.95.1 Detailed Description

Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.

The NonDEvidence class implements the propagation of epistemic uncertainty using Dempster-Shafer theory of evidence. In this approach, one assigns a set of basic probability assignments (BPA) to intervals defined for the uncertain variables. Input interval combinations are calculated, along with their BPA. Currently, the response function is evaluated at a set of sample points, then a response surface is constructed which is sampled extensively to find the minimum and maximum within each input interval cell, corresponding to the belief and plausibility within that cell, respectively. This data is then aggregated to calculate cumulative distribution functions for belief and plausibility.

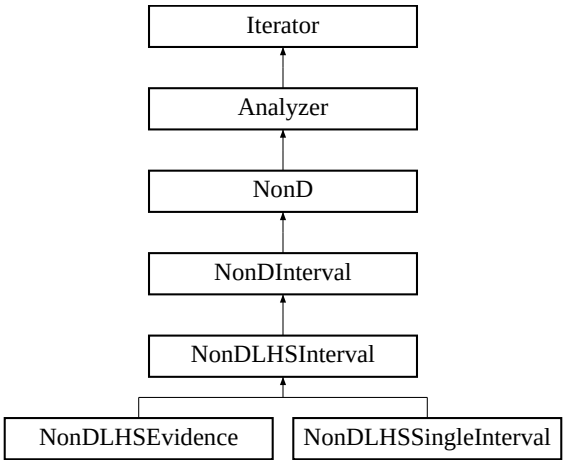
The documentation for this class was generated from the following files:

- NonDLHSEvidence.hpp
- NonDLHSEvidence.cpp

13.96 NonDLHSInterval Class Reference

Class for the LHS-based interval methods within DAKOTA/UQ.

Inheritance diagram for NonDLHSInterval:



Public Member Functions

- NonDLHSInterval (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDLHSInterval ()

  destructor
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void quantify_uncertainty ()

  performs an epistemic uncertainty propagation using LHS samples

Protected Member Functions

- virtual void initialize ()

  perform any required initialization
- virtual void post_process_samples ()=0

  post-process the output from executing lhsSampler

Protected Attributes

- Iterator lhsSampler

  the LHS sampler instance
- const int seedSpec

  the user seed specification (default is 0)
- int numSamples

  the number of samples used
- String rngName

  name of the random number generator

Additional Inherited Members

13.96.1 Detailed Description

Class for the LHS-based interval methods within DAKOTA/UQ.

The NonDLHSInterval class implements the propagation of epistemic uncertainty using LHS-based methods.

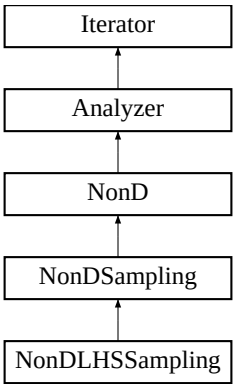
The documentation for this class was generated from the following files:

- NonDLHSInterval.hpp
- NonDLHSInterval.cpp

13.97 NonDLHSSampling Class Reference

Performs LHS and Monte Carlo sampling for uncertainty quantification.

Inheritance diagram for NonDLHSSampling:



Public Member Functions

- NonDLHSSampling (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NonDLHSSampling (Model &model, unsigned short sample_type, int samples, int seed, const String &rng, bool vary_pattern=true, short sampling_vars_mode=ACTIVE)

  alternate constructor for sample generation and evaluation "on the fly"

- NonDLHSSampling (unsigned short sample_type, int samples, int seed, const String &rng, const RealVector &lower_bnds, const RealVector &upper_bnds)

  alternate constructor for sample generation "on the fly"

- ~NonDLHSSampling ()

  destructor

Protected Member Functions

- void pre_run ()

  generate LHS samples in non-VBD cases

- void post_input ()

  read tabular data for post-run mode

- void quantify_uncertainty ()

  perform the evaluate parameter sets portion of run

- void post_run (std::ostream &s)

  generate statistics for LHS runs in non-VBD cases

- void print_results (std::ostream &s)

  print the final statistics

Private Attributes

- size_t numResponseFunctions

  number of response functions; used to distinguish NonD from opt/NLS usage

- bool varBasedDecompFlag

  flags computation of variance-based decomposition indices

Additional Inherited Members

13.97.1 Detailed Description

Performs LHS and Monte Carlo sampling for uncertainty quantification.

The Latin Hypercube Sampling (LHS) package from Sandia Albuquerque's Risk and Reliability organization provides comprehensive capabilities for Monte Carlo and Latin Hypercube sampling within a broad array of user-specified probabilistic parameter distributions. It enforces user-specified rank correlations through use of a mixing routine. The NonDLHSSampling class provides a C++ wrapper for the LHS library and is used for performing forward propagations of parameter uncertainties into response statistics.

13.97.2 Constructor & Destructor Documentation

NonDLHSSampling (ProblemDescDB & *problem_db*, Model & *model*)

standard constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

NonDLHSSampling (Model & *model*, unsigned short *sample_type*, int *samples*, int *seed*, const String & *rng*, bool *vary_pattern* = `true`, short *sampling_vars_mode* = `ACTIVE`)

alternate constructor for sample generation and evaluation "on the fly"

This alternate constructor is used for generation and evaluation of Model-based sample sets. A set_db_list_-nodes has not been performed so required data must be passed through the constructor. It's purpose is to avoid the need for a separate LHS specification within methods that use LHS sampling.

NonDLHSSampling (unsigned short *sample_type*, int *samples*, int *seed*, const String & *rng*, const RealVector & *lower_bnds*, const RealVector & *upper_bnds*)

alternate constructor for sample generation "on the fly"

This alternate constructor is used by ConcurrentStrategy for generation of uniform, uncorrelated sample sets. It is *not* a letter-envelope instantiation and a set_db_list_nodes has not been performed. It is called with all needed data passed through the constructor and is designed to allow more flexibility in variables set definition (i.e., relax connection to a variables specification and allow sampling over parameter sets such as multiobjective weights). In this case, a Model is not used and the object must only be used for sample generation (no evaluation).

References NonDSampling::get_parameter_sets().

13.97.3 Member Function Documentation

void quantify_uncertainty ( ) `[protected], [virtual]`

perform the evaluate parameter sets portion of run

Loop over the set of samples and compute responses. Compute statistics on the set of responses if statsFlag is set.

Implements NonD.

References NonDSampling::allDataFlag, Analyzer::evaluate_parameter_sets(), Iterator::iteratedModel, Analyzer-::numContinuousVars, Analyzer::numDiscreteIntVars, Analyzer::numDiscreteRealVars, NonDLHSSampling::num-ResponseFunctions, NonDSampling::numSamples, NonDSampling::statsFlag, NonDLHSSampling::varBasedDecomp-Flag, and Analyzer::variance_based_decomp().

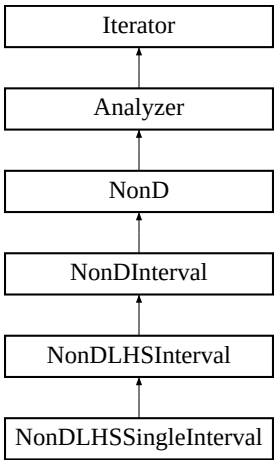
The documentation for this class was generated from the following files:

- NonDLHSSampling.hpp
- NonDLHSSampling.cpp

13.98 NonDLHSSingleInterval Class Reference

Class for pure interval propagation using LHS.

Inheritance diagram for NonDLHSSingleInterval:



Public Member Functions

- NonDLHSSingleInterval (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDLHSSingleInterval ()

  destructor

Protected Member Functions

- void initialize ()

  perform any required initialization
- void post_process_samples ()

  post-process the output from executing lhsSampler

Private Attributes

- size_t statCntr

  counter for finalStatistics

Additional Inherited Members

13.98.1 Detailed Description

Class for pure interval propagation using LHS.

The NonDSingleInterval class implements the propagation of epistemic uncertainty using ...

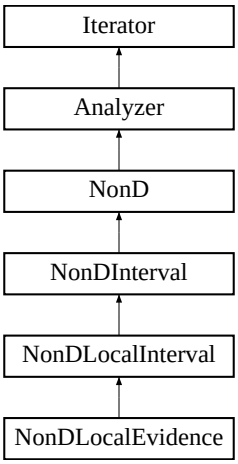
The documentation for this class was generated from the following files:

- NonDLHSSingleInterval.hpp
- NonDLHSSingleInterval.cpp

13.99 NonDLocalEvidence Class Reference

Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.

Inheritance diagram for NonDLocalEvidence:



Public Member Functions

- NonDLocalEvidence (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDLocalEvidence ()

  destructor

Protected Member Functions

- void initialize ()

  perform any required initialization
- void set_cell_bounds ()

  set the optimization variable bounds for each cell
- void truncate_to_cell_bounds (RealVector &initial_pt)

  truncate initial_pt to respect current cell lower/upper bounds
- void post_process_cell_results (bool maximize)

  post-process a cell minimization/maximization result
- void post_process_response_fn_results ()

  post-process the interval computed for a response function
- void post_process_final_results ()

  perform final post-processing

Additional Inherited Members

13.99.1 Detailed Description

Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.

The NonDEvidence class implements the propagation of epistemic uncertainty using Dempster-Shafer theory of evidence. In this approach, one assigns a set of basic probability assignments (BPA) to intervals defined for the uncertain variables. Input interval combinations are calculated, along with their BPA. Currently, the response function is evaluated at a set of sample points, then a response surface is constructed which is sampled extensively to find the minimum and maximum within each input interval cell, corresponding to the belief and plausibility within that cell, respectively. This data is then aggregated to calculate cumulative distribution functions for belief and plausibility.

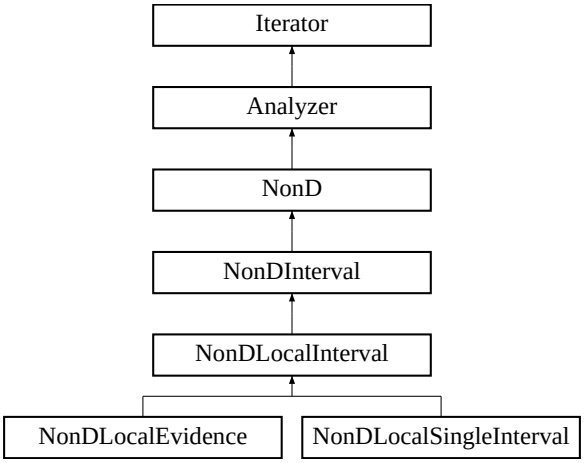
The documentation for this class was generated from the following files:

- NonDLocalEvidence.hpp
- NonDLocalEvidence.cpp

13.100 NonDLocalInterval Class Reference

Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.

Inheritance diagram for NonDLocalInterval:



Public Member Functions

- NonDLocalInterval (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDLocalInterval ()

  destructor
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

derived class contributions to setting the communicators associated with this Iterator instance

- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance

- void quantify_uncertainty ()

  Performs a gradient-based optimization to determine interval bounds for an entire function or interval bounds on a particular statistical estimator.

- unsigned short uses_method () const

  return name of active optimizer method

- void method_recourse ()

  perform an MPP optimizer method switch due to a detected conflict

Protected Member Functions

- virtual void initialize ()

  perform any required initialization

- virtual void set_cell_bounds ()

  set the optimization variable bounds for each cell

- virtual void truncate_to_cell_bounds (RealVector &initial_pt)

  truncate initial_pt to respect current cell lower/upper bounds

- virtual void post_process_cell_results (bool maximize)

  post-process a cell minimization/maximization result

- virtual void post_process_response_fn_results ()

  post-process the interval computed for a response function

- virtual void post_process_final_results ()

  perform final post-processing

Protected Attributes

- Iterator minMaxOptimizer

  local gradient-based optimizer

- Model minMaxModel

  recast model which extracts the active objective function

Static Private Member Functions

- static void extract_objective (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used to extract the active objective function when optimizing for an interval lower or upper bound

Private Attributes

- bool npsolFlag

  flag representing the gradient-based optimization algorithm selection (NPSOL SQP or OPT++ NIP)

Static Private Attributes

- static NonDLocalInterval * nondLIInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.100.1 Detailed Description

Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.

The NonDLocalInterval class supports local gradient-based optimization apporaches to determining interval bounds for epistemic UQ. The interval bounds may be on the entire function in the case of pure interval analysis (e.g. intervals on input = intervals on output), or the intervals may be on statistics of an "inner loop" aleatory analysis such as intervals on means, variances, or percentile levels.

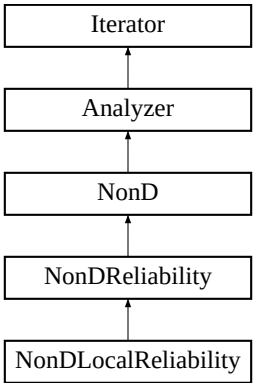
The documentation for this class was generated from the following files:

- NonDLocalInterval.hpp
- NonDLocalInterval.cpp

13.101 NonDLocalReliability Class Reference

Class for the reliability methods within DAKOTA/UQ.

Inheritance diagram for NonDLocalReliability:



Public Member Functions

- NonDLocalReliability (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDLocalReliability ()

  destructor
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

derived class contributions to setting the communicators associated with this Iterator instance

- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance

- void quantify_uncertainty ()

  performs an uncertainty propagation using analytical reliability methods which solve constrained optimization problems to obtain approximations of the cumulative distribution function of response

- void print_results (std::ostream &s)

  print the approximate mean, standard deviation, and importance factors when using the mean value method or the CDF/CCDF information when using MPP-search-based reliability methods

- unsigned short uses_method () const

  return name of active MPP optimizer

- void method_recourse ()

  perform an MPP optimizer method switch due to a detected conflict

Private Member Functions

- void initial_taylor_series ()

  convenience function for performing the initial limit state Taylor-series approximation

- void mean_value ()

  convenience function for encapsulating the simple Mean Value computation of approximate statistics and importance factors

- void mpp_search ()

  convenience function for encapsulating the reliability methods that employ a search for the most probable point (AMV, AMV+, FORM, SORM)

- void initialize_class_data ()

  convenience function for initializing class scope arrays

- void initialize_level_data ()

  convenience function for initializing/warm starting MPP search data for each response function prior to level 0

- void initialize_mpp_search_data ()

  convenience function for initializing/warm starting MPP search data for each z/p/beta level for each response function

- void update_mpp_search_data (const Variables &vars_star, const Response &resp_star)

  convenience function for updating MPP search data for each z/p/beta level for each response function

- void update_level_data ()

  convenience function for updating z/p/beta level data and final statistics following MPP convergence

- void update_pma_maximize (const RealVector &mpp_u, const RealVector &fn_grad_u, const RealSymMatrix &fn_hess_u)

  update pmaMaximizeG from prescribed probabilities or prescribed generalized reliabilities by inverting second-order integrations

- void update_limit_state_surrogate ()

  convenience function for passing the latest variables/response data to the data fit embedded within uSpaceModel

- void assign_mean_data ()

  update mostProbPointX/U, computedRespLevel, fnGradX/U, and fnHessX/U from ranVarMeansX/U, fnValsMeanX, fnGradsMeanX, and fnHessiansMeanX

- void dg_ds_eval (const RealVector &x_vars, const RealVector &fn_grad_x, RealVector &final_stat_grad)

convenience function for evaluating dg/ds

- Real dp2_dbeta_factor (Real beta, bool cdf_flag)

  compute factor for derivative of second-order probability with respect to reliability index (from differentiating BREITUNG or HOHENRACK expressions)

- Real signed_norm (const RealVector &mpp_u, const RealVector &fn_grad_u, bool cdf_flag)

  convert norm of mpp_u (u-space solution) to a signed reliability index

- Real signed_norm (Real norm_mpp_u)

  convert norm of u-space vector to a signed reliability index

- Real signed_norm (Real norm_mpp_u, const RealVector &mpp_u, const RealVector &fn_grad_u, bool cdf_flag)

  shared helper function

- Real probability (Real beta)

  Convert reliability to probability using a first-order integration.

- Real probability (bool cdf_flag, const RealVector &mpp_u, const RealVector &fn_grad_u, const RealSymMatrix &fn_hess_u)

  Convert computed reliability to probability using either a first-order or second-order integration.

- Real probability (Real beta, bool cdf_flag, const RealVector &mpp_u, const RealVector &fn_grad_u, const RealSymMatrix &fn_hess_u)

  Convert provided reliability to probability using either a first-order or second-order integration.

- Real reliability (Real p)

  Convert probability to reliability using the inverse of a first-order integration.

- Real reliability (Real p, bool cdf_flag, const RealVector &mpp_u, const RealVector &fn_grad_u, const RealSymMatrix &fn_hess_u)

  Convert probability to reliability using the inverse of a first-order or second-order integration.

- bool reliability_residual (const Real &p, const Real &beta, const RealVector &kappa, Real &res)

  compute the residual for inversion of second-order probability corrections using Newton's method (called by reliability(p))

- Real reliability_residual_derivative (const Real &p, const Real &beta, const RealVector &kappa)

  compute the residual derivative for inversion of second-order probability corrections using Newton's method (called by reliability(p))

- void principal_curvatures (const RealVector &mpp_u, const RealVector &fn_grad_u, const RealSymMatrix &fn_hess_u, RealVector &kappa_u)

  Compute the kappaU vector of principal curvatures from fnHessU.

- void scale_curvature (Real beta, bool cdf_flag, const RealVector &kappa, RealVector &scaled_kappa)

  scale copy of principal curvatures by -1 if needed; else take a view

Static Private Member Functions

- static void RIA_objective_eval (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used as the objective function in the Reliability Index Approach (RIA) problem formulation. This equality-constrained optimization problem performs the search for the most probable point (MPP) with the objective function of (norm u)^2.

- static void RIA_constraint_eval (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

static function used as the constraint function in the Reliability Index Approach (RIA) problem formulation. This equality-constrained optimization problem performs the search for the most probable point (MPP) with the constraint of G(u) = response level.

- static void PMA_objective_eval (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used as the objective function in the Performance Measure Approach (PMA) problem formulation. This equality-constrained optimization problem performs the search for the most probable point (MPP) with the objective function of G(u).

- static void PMA_constraint_eval (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used as the constraint function in the first-order Performance Measure Approach (PMA) problem formulation. This optimization problem performs the search for the most probable point (MPP) with the equality constraint of (norm u)^2 = (beta-bar)^2.

- static void PMA2_constraint_eval (const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  static function used as the constraint function in the second-order Performance Measure Approach (PMA) problem formulation. This optimization problem performs the search for the most probable point (MPP) with the equality constraint of beta∗ = beta∗-bar.

- static void PMA2_set_mapping (const Variables &recast_vars, const ActiveSet &recast_set, ActiveSet &sub_model_set)

  static function used to augment the sub-model ASV requests for second-order PMA

Private Attributes

- Real computedRespLevel

  output response level calculated

- Real computedRelLevel

  output reliability level calculated for RIA and 1st-order PMA

- Real computedGenRelLevel

  output generalized reliability level calculated for 2nd-order PMA

- RealVector fnGradX

  actual x-space gradient for current function from most recent response evaluation

- RealVector fnGradU

  u-space gradient for current function updated from fnGradX and Jacobian dx/du

- RealSymMatrix fnHessX

  actual x-space Hessian for current function from most recent response evaluation

- RealSymMatrix fnHessU

  u-space Hessian for current function updated from fnHessX and Jacobian dx/du

- RealVector kappaU

  principal curvatures derived from eigenvalues of orthonormal transformation of fnHessU

- RealVector fnValsMeanX

  response function values evaluated at mean x

- RealMatrix fnGradsMeanX

  response function gradients evaluated at mean x

- RealSymMatrixArray fnHessiansMeanX

  response function Hessians evaluated at mean x

- RealVector ranVarMeansU

  vector of means for all uncertain random variables in u-space
- bool initialPtUserSpec

  flag indicating user specification of (any portion of) initialPtU
- RealVector initialPtUSpec

  user specification or default initial guess for local optimization
- RealVector initialPtU

  current starting point for MPP searches in u-space
- RealVector mostProbPointX

  location of MPP in x-space
- RealVector mostProbPointU

  location of MPP in u-space
- RealVectorArray prevMPPULev0

  array of converged MPP's in u-space for level 0. Used for warm-starting initialPtU within RBDO.
- RealMatrix prevFnGradDLev0

  matrix of limit state sensitivities w.r.t. inactive/design variables for level 0. Used for warm-starting initialPtU within RBDO.
- RealMatrix prevFnGradULev0

  matrix of limit state sensitivities w.r.t. active/uncertain variables for level 0. Used for warm-starting initialPtU within RBDO.
- RealVector prevICVars

  previous design vector. Used for warm-starting initialPtU within RBDO.
- ShortArray prevCumASVLev0

  accumulation (using |=) of all previous design ASV's from requested finalStatistics. Used to detect availability of prevFnGradDLev0 data for warm-starting initialPtU within RBDO.
- bool npsolFlag

  flag representing the optimization MPP search algorithm selection (NPSOL SQP or OPT++ NIP)
- bool warmStartFlag

  flag indicating the use of warm starts
- bool nipModeOverrideFlag

  flag indicating the use of move overrides within OPT++ NIP
- bool curvatureDataAvailable

  flag indicating that sufficient data (i.e., fnGradU, fnHessU, mostProbPointU) is available for computing principal curvatures
- bool kappaUpdated

  track when kappaU requires updating via principal_curvatures()
- short integrationOrder

  integration order (1 or 2) provided by `integration` *specification*
- short secondOrderIntType

  type of second-order integration: Breitung, Hohenbichler-Rackwitz, or Hong
- Real curvatureThresh

  cut-off value for 1/sqrt() term in second-order probability corrections.
- short taylorOrder

  order of Taylor series approximations (1 or 2) in MV/AMV/AMV+ derived from hessian type

- RealMatrix impFactor

  importance factors predicted by MV
- int npsolDerivLevel

  derivative level for NPSOL executions (1 = analytic grads of objective fn, 2 = analytic grads of constraints, 3 = analytic grads of both).
- unsigned short warningBits

  set of warnings accumulated during execution

Static Private Attributes

- static NonDLocalReliability ∗ nondLocRelInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.101.1 Detailed Description

Class for the reliability methods within DAKOTA/UQ.

The NonDLocalReliability class implements the following reliability methods through the support of different limit state approximation and integration options: mean value (MVFOSM/MVSOSM), advanced mean value method (AMV, AMV^2) in x- or u-space, iterated advanced mean value method (AMV+, AMV^2+) in x- or u-space, two-point adaptive nonlinearity approximation (TANA) in x- or u-space, first order reliability method (FORM), and second order reliability method (SORM). All options except mean value employ an optimizer (currently NPSOL SQP or OPT++ NIP) to solve an equality-constrained optimization problem for the most probable point (MPP). The MPP search may be formulated as the reliability index approach (RIA) for mapping response levels to reliabilities/probabilities or as the performance measure approach (PMA) for performing the inverse mapping of reliability/probability levels to response levels.

13.101.2 Member Function Documentation

void RIA_objective_eval (const Variables & *sub_model_vars,* const Variables & *recast_vars,* const Response & *sub_model_response,* Response & *recast_response*) `[static], [private]`

static function used as the objective function in the Reliability Index Approach (RIA) problem formulation. This equality-constrained optimization problem performs the search for the most probable point (MPP) with the objective function of (norm u)^2.

This function recasts a G(u) response set (already transformed and approximated in other recursions) into an RIA objective function.

References Response::active_set_request_vector(), Variables::continuous_variables(), Response::function_gradient_view(), Response::function_hessian_view(), and Response::function_value().

Referenced by NonDLocalReliability::mpp_search().

void RIA_constraint_eval (const Variables & *sub_model_vars,* const Variables & *recast_vars,* const Response & *sub_model_response,* Response & *recast_response*) `[static], [private]`

static function used as the constraint function in the Reliability Index Approach (RIA) problem formulation. This equality-constrained optimization problem performs the search for the most probable point (MPP) with the constraint of G(u) = response level.

This function recasts a G(u) response set (already transformed and approximated in other recursions) into an RIA equality constraint.

References Response::active_set_request_vector(), Response::function_gradient(), Response::function_gradient_view(), Response::function_hessian(), Response::function_value(), NonDLocalReliability::nondLocRelInstance, NonDReliability::requestedTargetLevel, and NonDReliability::respFnCount.

Referenced by NonDLocalReliability::mpp_search().

void PMA_objective_eval (const Variables & *sub_model_vars,* const Variables & *recast_vars,* const Response & *sub_model_response,* Response & *recast_response*) `[static], [private]`

static function used as the objective function in the Performance Measure Approach (PMA) problem formulation. This equality-constrained optimization problem performs the search for the most probable point (MPP) with the objective function of G(u).

This function recasts a G(u) response set (already transformed and approximated in other recursions) into an PMA objective function.

References Response::active_set_request_vector(), Variables::continuous_variables(), NonDLocalReliability::curvatureDataAvailable, Response::function_gradient(), Response::function_gradient_view(), Response::function_hessian(), Response::function_hessian_view(), Response::function_value(), NonDLocalReliability::integrationOrder, NonDLocalReliability::kappaUpdated, NonDReliability::mppSearchType, NonDLocalReliability::nondLocRelInstance, NonDReliability::pmaMaximizeG, NonDReliability::respFnCount, and NonDLocalReliability::update_pma_maximize().

Referenced by NonDLocalReliability::mpp_search().

void PMA_constraint_eval (const Variables & *sub_model_vars,* const Variables & *recast_vars,* const Response & *sub_model_response,* Response & *recast_response*) `[static], [private]`

static function used as the constraint function in the first-order Performance Measure Approach (PMA) problem formulation. This optimization problem performs the search for the most probable point (MPP) with the equality constraint of (norm u)^2 = (beta-bar)^2.

This function recasts a G(u) response set (already transformed and approximated in other recursions) into a first-order PMA equality constraint on reliability index beta.

References Response::active_set_request_vector(), Variables::continuous_variables(), Response::function_gradient_view(), Response::function_hessian_view(), Response::function_value(), NonDLocalReliability::nondLocRelInstance, and NonDReliability::requestedTargetLevel.

Referenced by NonDLocalReliability::mpp_search().

void PMA2_constraint_eval (const Variables & *sub_model_vars,* const Variables & *recast_vars,* const Response & *sub_model_response,* Response & *recast_response*) `[static], [private]`

static function used as the constraint function in the second-order Performance Measure Approach (PMA) problem formulation. This optimization problem performs the search for the most probable point (MPP) with the equality constraint of beta∗ = beta∗-bar.

This function recasts a G(u) response set (already transformed and approximated in other recursions) into a second-order PMA equality constraint on generalized reliability index beta-star.

References Dakota::abort_handler(), Response::active_set_request_vector(), NonD::cdfFlag, NonDLocalReliability::computedGenRelLevel, NonDLocalReliability::computedRelLevel, Variables::continuous_variables(), NonDLocalReliability::dp2_dbeta_factor(), NonDLocalReliability::fnGradU, NonDLocalReliability::fnHessU, Response::function_gradient_view(), Response::function_hessian(), Response::function_value(), NonDLocalReliability::mostProbPointU, NonDReliability::mppSearchType, NonDLocalReliability::nondLocRelInstance, NonDLocalReliability::probability(),

NonDLocalReliability::reliability(), NonDReliability::requestedTargetLevel, NonDReliability::respFnCount, and NonDLocalReliability::signed_norm().

Referenced by NonDLocalReliability::mpp_search().

void initial_taylor_series ( ) [private]

convenience function for performing the initial limit state Taylor-series approximation

An initial first- or second-order Taylor-series approximation is required for MV/AMV/AMV+/TANA or for the case where momentStats (from MV) are required within finalStatistics for subIterator usage of NonDLocalReliability.

References Response::active_set_request_vector(), Iterator::activeSet, Model::component_parallel_mode(), Model::compute_response(), Model::continuous_variables(), Model::current_response(), NonD::finalStatistics, NonDLocalReliability::fnGradsMeanX, NonDLocalReliability::fnHessiansMeanX, NonDLocalReliability::fnValsMeanX, Response::function_gradients(), Response::function_hessians(), Response::function_values(), Model::hessian_type(), Iterator::iteratedModel, NonD::momentStats, NonDReliability::mppSearchType, NonD::natafTransform, Analyzer::numFunctions, NonD::numUncertainVars, ActiveSet::request_vector(), NonD::requestedGenRelLevels, NonD::requestedProbLevels, NonD::requestedRelLevels, NonD::requestedRespLevels, Iterator::subIteratorFlag, NonDLocalReliability::taylorOrder, and NonDReliability::uSpaceModel.

Referenced by NonDLocalReliability::mean_value(), and NonDLocalReliability::mpp_search().

void initialize_class_data ( ) [private]

convenience function for initializing class scope arrays

Initialize class-scope arrays and perform other start-up activities, such as evaluating median limit state responses.

References Response::active_set_derivative_vector(), NonD::finalStatistics, NonDReliability::importanceSampler, NonD::initialize_random_variables(), NonDReliability::integrationRefinement, Iterator::iterator_rep(), NonDReliability::mppModel, NonD::natafTransform, Analyzer::numFunctions, NonDReliability::numRelAnalyses, NonD::numUncertainVars, NonDLocalReliability::prevCumASVLev0, NonDLocalReliability::prevFnGradDLev0, NonDLocalReliability::prevFnGradULev0, NonDLocalReliability::prevMPPULev0, NonDLocalReliability::ranVarMeansU, Iterator::subIteratorFlag, Model::update_from_subordinate_model(), and NonDLocalReliability::warmStartFlag.

Referenced by NonDLocalReliability::mpp_search().

void initialize_level_data ( ) [private]

convenience function for initializing/warm starting MPP search data for each response function prior to level 0

For a particular response function prior to the first z/p/beta level, initialize/warm-start optimizer initial guess (initialPtU), expansion point (mostProbPointX/U), and associated response data (computedRespLevel, fnGradX/U, and fnHessX/U).

References Iterator::activeSet, NonDLocalReliability::assign_mean_data(), Model::component_parallel_mode(), Model::compute_response(), NonDLocalReliability::computedRespLevel, Model::continuous_variable_ids(), Model::continuous_variables(), Dakota::copy_data(), Model::current_response(), NonDLocalReliability::curvatureDataAvailable, NonDLocalReliability::fnGradU, NonDLocalReliability::fnGradX, NonDLocalReliability::fnHessU, NonDLocalReliability::fnHessX, Response::function_gradient_copy(), Response::function_hessian(), Response::function_value(), Model::inactive_continuous_variables(), NonDLocalReliability::initialPtU, NonDLocalReliability::initialPtUSpec, Iterator::iteratedModel, NonDLocalReliability::kappaUpdated, NonDLocalReliability::mostProbPointU, NonDLocalReliability::mostProbPointX, NonDReliability::mppSearchType, NonD::natafTransform, NonDReliability::numRelAnalyses, NonD::numUncertainVars, NonDLocalReliability::prevCumASVLev0, NonDLocalReliability::prevFnGradDLev0, NonDLocalReliability::prevFnGradULev0, NonDLocalReliability::prevICVars, NonDLocalReliability::prevMPPULev0, ActiveSet::request_value(), ActiveSet::request_values(), NonD::requestedResp-

Levels, NonDReliability::respFnCount, Iterator::subIteratorFlag, Model::surrogate_function_indices(), NonDLocalReliability::taylorOrder, NonDLocalReliability::update_limit_state_surrogate(), NonDReliability::uSpaceModel, and NonDLocalReliability::warmStartFlag.

Referenced by NonDLocalReliability::mpp_search().

void initialize_mpp_search_data ( ) [private]

convenience function for initializing/warm starting MPP search data for each z/p/beta level for each response function

For a particular response function at a particular z/p/beta level, warm-start or reset the optimizer initial guess (initialPtU), expansion point (mostProbPointX/U), and associated response data (computedRespLevel, fnGradX/-U, and fnHessX/U).

References NonDLocalReliability::assign_mean_data(), NonD::computedGenRelLevels, NonD::computedRelLevels, NonDLocalReliability::fnGradU, Model::hessian_type(), NonDLocalReliability::initialPtU, NonDLocalReliability::initialPtUSpec, NonDLocalReliability::integrationOrder, Iterator::iteratedModel, NonDReliability::levelCount, NonDLocalReliability::mostProbPointU, NonDReliability::mppSearchType, NonD::numUncertainVars, NonD::requestedProbLevels, NonD::requestedRelLevels, NonD::requestedRespLevels, NonDReliability::requestedTargetLevel, NonDReliability::respFnCount, NonDLocalReliability::taylorOrder, and NonDLocalReliability::warmStartFlag.

Referenced by NonDLocalReliability::mpp_search().

void update_mpp_search_data (const Variables & *vars_star,* const Response & *resp_star*) [private]

convenience function for updating MPP search data for each z/p/beta level for each response function

Includes case-specific logic for updating MPP search data for the AMV/AMV+/TANA/NO_APPROX methods.

References Response::active_set(), Response::active_set_request_vector(), Iterator::activeSet, NonDReliability::approxConverged, NonDReliability::approxIters, Model::component_parallel_mode(), Model::compute_response(), NonDLocalReliability::computedRelLevel, NonDLocalReliability::computedRespLevel, Model::continuous_variable_ids(), Variables::continuous_variables(), Model::continuous_variables(), Iterator::convergenceTol, Variables::copy(), Dakota::copy_data(), Model::current_response(), Model::current_variables(), NonDLocalReliability::curvatureDataAvailable, Dakota::data_pairs, NonD::finalStatistics, NonDLocalReliability::fnGradU, NonDLocalReliability::fnGradX, NonDLocalReliability::fnHessU, NonDLocalReliability::fnHessX, Response::function_gradient_copy(), Response::function_hessian(), Response::function_value(), Response::function_values(), NonDLocalReliability::initialPtU, NonDLocalReliability::integrationOrder, Model::interface_id(), Iterator::iteratedModel, NonDLocalReliability::kappaUpdated, NonDReliability::levelCount, Dakota::lookup_by_val(), Iterator::maxIterations, NonDLocalReliability::mostProbPointU, NonDLocalReliability::mostProbPointX, NonDReliability::mppSearchType, NonD::natafTransform, Analyzer::numFunctions, NonD::numNormalVars, NonD::numUncertainVars, NonDReliability::pmaMaximizeG, ActiveSet::request_value(), ActiveSet::request_values(), ActiveSet::request_vector(), NonD::requestedProbLevels, NonD::requestedRelLevels, NonD::requestedRespLevels, NonDReliability::requestedTargetLevel, NonDReliability::respFnCount, NonDLocalReliability::signed_norm(), NonDReliability::statCount, NonDLocalReliability::taylorOrder, NonDLocalReliability::update_limit_state_surrogate(), NonDLocalReliability::update_pma_maximize(), NonDReliability::uSpaceModel, NonDLocalReliability::warmStartFlag, and NonDLocalReliability::warningBits.

Referenced by NonDLocalReliability::mpp_search().

void update_level_data ( ) [private]

convenience function for updating z/p/beta level data and final statistics following MPP convergence

Updates computedRespLevels/computedProbLevels/computedRelLevels, finalStatistics, warm start, and graphics data.

References Response::active_set_derivative_vector(), Response::active_set_request_vector(), Graphics::add_datapoint(), NonD::cdfFlag, NonDLocalReliability::computedGenRelLevel, NonD::computedGenRelLevels, NonD::computedProbLevels, NonDLocalReliability::computedRelLevel, NonD::computedRelLevels, NonDLocalReliability::computedRespLevel, NonD::computedRespLevels, NonDLocalReliability::dg_ds_eval(), NonDLocalReliability::dp2_dbeta_factor(), NonD::finalStatistics, NonDLocalReliability::fnGradU, NonDLocalReliability::fnGradX, NonDLocalReliability::fnHessU, Response::function_gradient(), OutputManager::graphics(), NonDLocalReliability::integrationOrder, NonDReliability::levelCount, NonDLocalReliability::mostProbPointU, NonDLocalReliability::mostProbPointX, Graphics::new_dataset(), Analyzer::numFunctions, NonD::numUncertainVars, ParallelLibrary::output_manager(), Iterator::parallelLib, NonDLocalReliability::prevCumASVLev0, NonDLocalReliability::prevFnGradDLev0, NonDLocalReliability::prevFnGradULev0, NonDLocalReliability::prevMPPULev0, NonDLocalReliability::probability(), NonDLocalReliability::reliability(), NonD::requestedGenRelLevels, NonD::requestedProbLevels, NonD::requestedRelLevels, NonD::requestedRespLevels, NonDReliability::respFnCount, NonD::respLevelTarget, NonD::respLevelTargetReduce, NonDReliability::statCount, Iterator::subIteratorFlag, NonD::totalLevelRequests, and NonDLocalReliability::warmStartFlag.

Referenced by NonDLocalReliability::mpp_search().

void dg_ds_eval (const RealVector & *x_vars,* const RealVector & *fn_grad_x,* RealVector & *final_stat_grad*) [private]

convenience function for evaluating dg/ds

Computes dg/ds where s = design variables. Supports potentially overlapping cases of design variable augmentation and insertion.

References Response::active_set_derivative_vector(), Iterator::activeSet, Model::all_continuous_variable_ids(), Model::component_parallel_mode(), Model::compute_response(), Dakota::contains(), Model::continuous_variable_ids(), Model::continuous_variables(), Dakota::copy_data(), Model::current_response(), ActiveSet::derivative_vector(), NonD::finalStatistics, Response::function_gradient_copy(), Response::function_gradients(), Model::inactive_continuous_variable_ids(), Iterator::iteratedModel, NonDReliability::mppSearchType, NonD::natafTransform, Iterator::primaryACVarMapIndices, ActiveSet::request_value(), ActiveSet::request_values(), NonDReliability::respFnCount, Iterator::secondaryACVarMapTargets, and NonDReliability::uSpaceModel.

Referenced by NonDLocalReliability::mean_value(), NonDLocalReliability::mpp_search(), and NonDLocalReliability::update_level_data().

Real dp2_dbeta_factor (Real *beta,* bool *cdf_flag*) [private]

compute factor for derivative of second-order probability with respect to reliability index (from differentiating BREITUNG or HOHENRACK expressions)

Compute sensitivity of second-order probability w.r.t. beta for use in derivatives of p_2 or beta∗ w.r.t. auxilliary parameters s (design, epistemic) or derivatives of beta∗ w.r.t. u in PMA2_constraint_eval().

References Dakota::abort_handler(), NonDLocalReliability::curvatureDataAvailable, NonDLocalReliability::curvatureThresh, NonDLocalReliability::kappaU, NonD::numUncertainVars, NonDLocalReliability::probability(), NonDLocalReliability::scale_curvature(), NonDLocalReliability::secondOrderIntType, and NonDLocalReliability::warningBits.

Referenced by NonDLocalReliability::PMA2_constraint_eval(), and NonDLocalReliability::update_level_data().

Real probability (Real *beta,* bool *cdf_flag,* const RealVector & *mpp_u,* const RealVector & *fn_grad_u,* const RealSymMatrix & *fn_hess_u*) [private]

Convert provided reliability to probability using either a first-order or second-order integration.

Converts beta into a probability using either first-order (FORM) or second-order (SORM) integration. The SORM calculation first calculates the principal curvatures at the MPP (using the approach in Ch. 8 of Haldar &

Mahadevan), and then applies correction formulations from the literature (Breitung, Hohenbichler-Rackwitz, or Hong).

References NonDLocalReliability::curvatureDataAvailable, NonDLocalReliability::curvatureThresh, NonDAdaptImpSampling::final_probability(), NonDReliability::importanceSampler, NonDLocalReliability::integrationOrder, NonDReliability::integrationRefinement, Iterator::iterator_rep(), NonDLocalReliability::kappaU, NonDLocalReliability::kappaUpdated, Iterator::methodPCIter, NonD::miPLIndex, NonD::numUncertainVars, Iterator::outputLevel, NonDLocalReliability::principal_curvatures(), NonDLocalReliability::probability(), NonDReliability::requestedTargetLevel, NonDReliability::respFnCount, Iterator::run(), NonDLocalReliability::scale_curvature(), NonDLocalReliability::secondOrderIntType, NonDLocalReliability::warningBits, and Dakota::write_precision.

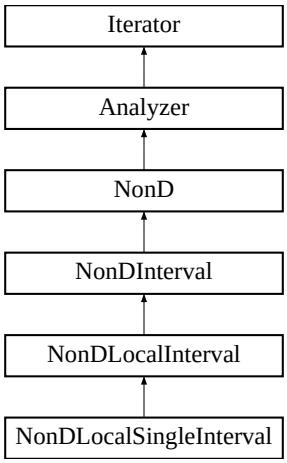
The documentation for this class was generated from the following files:

- NonDLocalReliability.hpp
- NonDLocalReliability.cpp

13.102 NonDLocalSingleInterval Class Reference

Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.

Inheritance diagram for NonDLocalSingleInterval:



Public Member Functions

- NonDLocalSingleInterval (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDLocalSingleInterval ()

  destructor

Protected Member Functions

- void initialize ()

  perform any required initialization
- void post_process_cell_results (bool maximize)

  post-process a cell minimization/maximization result

Private Attributes

- size_t statCntr

  counter for finalStatistics

Additional Inherited Members

13.102.1 Detailed Description

Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification.

The NonDLocalSingleInterval class supports local gradient-based optimization apporaches to determining interval bounds for epistemic UQ. The interval bounds may be on the entire function in the case of pure interval analysis (e.g. intervals on input = intervals on output), or the intervals may be on statistics of an "inner loop" aleatory analysis such as intervals on means, variances, or percentile levels.

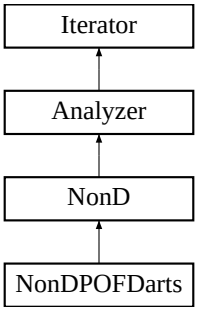
The documentation for this class was generated from the following files:

- NonDLocalSingleInterval.hpp
- NonDLocalSingleInterval.cpp

13.103 NonDPOFDarts Class Reference

Base class for POF Dart methods within DAKOTA/UQ.

Inheritance diagram for NonDPOFDarts:



Public Member Functions

- NonDPOFDarts (ProblemDescDB &problem_db, Model &model)

  constructor

- ~NonDPOFDarts ()

  destructor

- void quantify_uncertainty ()

  perform POFDart analysis and return probability of failure

Protected Member Functions

- void initiate_random_number_generator (unsigned long x)

  POF Darts Methods.
- double **generate_a_random_number** ()
- void **init_pof_darts** ()
- void **exit_pof_darts** ()
- void **execute** (size_t kd)
- void print_results (std::ostream &s)

  print the final statistics
- void **classical_dart_throwing_games** (size_t game_index)
- void **line_dart_throwing_games** (size_t game_index)
- bool **valid_dart** (double *x)
- bool **valid_line_flat** (size_t flat_dim, double *flat_dart)
- void **add_point** (double *x)
- void **compute_response** (double *x)
- void **retrieve_neighbors** (size_t ipoint, bool update_point_neighbors)
- void **update_global_L** ()
- void **assign_sphere_radius_POF** (size_t isample)
- void **shrink_big_spheres** ()
- double **area_triangle** (double x1, double y1, double x2, double y2, double x3, double y3)
- void **initialize_surrogates** ()
- void **add_surrogate_data** (const Variables &vars, const Response &resp)
- void **build_surrogate** (size_t fn_index)
- double **eval_surrogate** (size_t fn_index, double *vin)
- void **estimate_pof_surrogate** ()
- bool **trim_line_using_Hyperplane** (size_t num_dim, double *st, double *end, double *qH, double *nH)
- bool **VPS_execute** ()
- void **VPS_adjust_extend_neighbors_of_all_points** ()
- void **VPS_extend_neighbors** (size_t ipoint)
- void **VPS_retrieve_poly_coefficients_for_all_points** ()
- void **VPS_retrieve_poly_coefficients** (size_t ipoint, size_t function_index)
- void **estimate_pof_VPS** ()
- void **VPS_destroy_global_containers** ()
- void **retrieve_permutations** (size_t &m, size_t **&perm, size_t num_dim, size_t upper_bound, bool include_origin, bool force_sum_constraint, size_t sum_constraint)
- double **vec_pow_vec** (size_t num_dim, double *vec_a, size_t *vec_b)
- bool **Cholesky** (int n, double **A, double **LD)
- void **Cholesky_solver** (int n, double **LD, double *b, double *x)
- void **GMRES** (size_t n, double **A, double *b, double *x, double eps)
- double **f_true** (double *x)
- void **plot_vertices_2d** (bool plot_true_function, bool plot_suurogate)
- void **plot_neighbors** ()

Protected Attributes

- int **samples**
- int **seed**
- int **emulatorOrder**
- int **emulatorSamples**
- short **emulatorType**
- String **lipschitzType**
- double **Q** [1220]
- int **indx**
- double **cc**
- double **c**
- double **zc**
- double **zx**
- double **zy**
- size_t **qlen**
- bool **_eval_error**
- size_t **_n_dim**
- double * **_xmin**
- double * **_xmax**
- double **_diag**
- double **_failure_threshold**
- double **_num_darts**
- double **_num_successive_misses_p**
- double **_num_successive_misses_m**
- double **_max_num_successive_misses**
- double **_accepted_void_ratio**
- size_t **_num_inserted_points**
- size_t **_total_budget**
- double ** **_sample_points**
- size_t ** **_sample_neighbors**
- double * **_sample_vsize**
- double **_max_vsize**
- double * **_dart**
- size_t **_flat_dim**
- size_t * **_line_flat**
- size_t **_num_flat_segments**
- double * **_line_flat_start**
- double * **_line_flat_end**
- double * **_line_flat_length**
- double * **_Lip**
- double ** **_fval**
- size_t **_active_response_function**
- SharedApproxData **sharedData**
- std::vector< Approximation > **gpApproximations**
- Variables **gpEvalVars**
- bool **_use_vor_surrogate**

- bool **_use_local_L**
- size_t **_vps_order**
- size_t **_vps_num_poly_terms**
- size_t **_num_GMRES**
- double * **_vps_dfar**
- size_t ** **_vps_t**
- double *** **_vps_c**
- size_t ** **_vps_ext_neighbors**

Additional Inherited Members

13.103.1 Detailed Description

Base class for POF Dart methods within DAKOTA/UQ.

The NonDPOFDart class implements the calculation of a failure probability for a specified threshold for a specified response function using the concepts developed by Mohamed Ebeida. The approach works by throwing down a number of Poisson disk samples of varying radii, and identifying each disk as either in the failure or safe region. The center of each disk represents a "true" function evaluation. kd-darts are used to place additional points, in such a way to target the failure region. When the disks cover the space sufficiently, Monte Carlo methods or a box volume approach is used to calculate both the lower and upper bounds on the failure probability.

13.103.2 Constructor & Destructor Documentation

NonDPOFDarts (ProblemDescDB & *problem_db*, Model & *model*)

constructor

/*if (!_use_vor_surrogate)*/ initialize_surrogates(); // initialize one GP surrogate per function

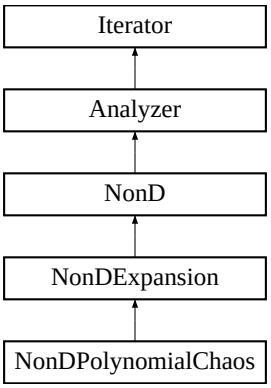
The documentation for this class was generated from the following files:

- NonDPOFDarts.hpp
- NonDPOFDarts.cpp

13.104 NonDPolynomialChaos Class Reference

Nonintrusive polynomial chaos expansion approaches to uncertainty quantification.

Inheritance diagram for NonDPolynomialChaos:



Public Member Functions

- NonDPolynomialChaos (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NonDPolynomialChaos (Model &model, short exp_coeffs_approach, unsigned short num_int_level, short u_space_type, bool piecewise_basis, bool use_derivs)

  alternate constructor
- ~NonDPolynomialChaos ()

  destructor

Protected Member Functions

- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void resolve_inputs (short &u_space_type, short &data_order)

  perform error checks and mode overrides
- void initialize_u_space_model ()

  initialize uSpaceModel polynomial approximations with PCE/SC data
- void increment_specification_sequence ()

  increment the input specification sequence (PCE only)
- void compute_expansion ()

  form or import an orthogonal polynomial expansion using PCE methods
- void increment_order ()

  uniformly increment the order of the polynomial chaos expansion
- void print_coefficients (std::ostream &s)

  print the PCE coefficient array for the orthogonal basis
- void archive_coefficients ()

  archive the PCE coefficient array for the orthogonal basis

Private Member Functions

- int terms_ratio_to_samples (size_t num_exp_terms, Real colloc_ratio, Real terms_order)

  convert number of expansion terms and collocation ratio to a number of collocation samples
- Real terms_samples_to_ratio (size_t num_exp_terms, int samples, Real terms_order)

  convert number of expansion terms and number of collocation samples to a collocation ratio
- void order_to_dim_preference (const UShortArray &order, unsigned short &p, RealVector &dim_pref)

  convert an isotropic/anisotropic expansion_order vector into a scalar plus a dimension preference vector

Private Attributes

- String expansionExportFile

  filename for export of chaos coefficients
- String expansionImportFile

  filename for import of chaos coefficients
- Real collocRatio

  factor applied to terms^termsOrder in computing number of regression points, either user specified or inferred
- Real termsOrder

  exponent applied to number of expansion terms for computing number of regression points
- int randomSeed

  seed for random number generator used for regression with LHS and sub-sampled tensor grids
- bool tensorRegression

  option for regression PCE using a filtered set tensor-product points
- bool crossValidation

  flag for use of cross-validation for selection of parameter settings in regression approaches
- RealVector noiseTols

  noise tolerance for compressive sensing algorithms; vector form used in cross-validation
- Real l2Penalty

  L2 penalty for LASSO algorithm (elastic net variant)
- unsigned short numAdvance

  number of frontier expansions per iteration with the ADAPTED_BASIS_EXPANDING_FRONT approach
- UShortArray expOrderSeqSpec

  user specification for expansion_order (array for multifidelity)
- RealVector dimPrefSpec

  user specification for dimension_preference
- SizetArray collocPtsSeqSpec

  user specification for collocation_points (array for multifidelity)
- SizetArray expSamplesSeqSpec

  user specification for expansion_samples (array for multifidelity)
- size_t sequenceIndex

  sequence index for {expOrder,collocPts,expSamples}SeqSpec
- RealMatrix pceGradsMeanX

  derivative of the PCE with respect to the x-space variables evaluated at the means (used as uncertainty importance metrics)
- bool normalizedCoeffOutput

  user request for use of normalization when outputting PCE coefficients

Additional Inherited Members

13.104.1 Detailed Description

Nonintrusive polynomial chaos expansion approaches to uncertainty quantification.

The NonDPolynomialChaos class uses a polynomial chaos expansion (PCE) approach to approximate the effect of parameter uncertainties on response functions of interest. It utilizes the OrthogPolyApproximation class to manage multiple types of orthogonal polynomials within a Wiener-Askey scheme to PCE. It supports PCE coefficient estimation via sampling, quadrature, point-collocation, and file import.

13.104.2 Constructor & Destructor Documentation

NonDPolynomialChaos (ProblemDescDB & *problem_db*, Model & *model*)

standard constructor

This constructor is called for a standard letter-envelope iterator instantiation using the ProblemDescDB.

References Dakota::abort_handler(), NonDIntegration::anisotropic_order_to_dimension_preference(), Model::assign_rep(), NonDPolynomialChaos::collocPtsSeqSpec, NonDPolynomialChaos::collocRatio, ParallelLibrary::command_line_check(), NonDExpansion::construct_cubature(), NonDExpansion::construct_expansion_sampler(), NonD::construct_lhs(), NonDExpansion::construct_quadrature(), NonDExpansion::construct_sparse_grid(), NonDIntegration::dimension_preference_to_anisotropic_order(), NonDPolynomialChaos::dimPrefSpec, NonDExpansion::expansionBasisType, NonDExpansion::expansionCoeffsApproach, NonDPolynomialChaos::expansionImportFile, NonDPolynomialChaos::expOrderSeqSpec, NonDPolynomialChaos::expSamplesSeqSpec, ProblemDescDB::get_bool(), ProblemDescDB::get_real(), ProblemDescDB::get_short(), ProblemDescDB::get_string(), ProblemDescDB::get_usa(), ProblemDescDB::get_ushort(), NonDExpansion::initialize(), NonDPolynomialChaos::initialize_u_space_model(), Iterator::iteratedModel, Iterator::maxEvalConcurrency, NonD::numContDesVars, NonD::numContEpistUncVars, Analyzer::numContinuousVars, NonD::numContStateVars, NonDExpansion::numSamplesOnModel, Iterator::outputLevel, Iterator::parallelLib, Iterator::probDescDB, NonDPolynomialChaos::randomSeed, NonDExpansion::refineControl, NonDExpansion::refineType, NonDPolynomialChaos::resolve_inputs(), NonDPolynomialChaos::sequenceIndex, NonDPolynomialChaos::tensorRegression, NonDPolynomialChaos::terms_ratio_to_samples(), NonDPolynomialChaos::terms_samples_to_ratio(), NonDPolynomialChaos::termsOrder, NonD::transform_model(), NonDExpansion::uSpaceModel, and Analyzer::vary_pattern().

NonDPolynomialChaos (Model & *model*, short *exp_coeffs_approach*, unsigned short *num_int_level*, short *u_space_type*, bool *piecewise_basis*, bool *use_derivs*)

alternate constructor

This constructor is used for helper iterator instantiation on the fly.

References Model::assign_rep(), NonDExpansion::construct_cubature(), NonDExpansion::construct_quadrature(), NonDExpansion::construct_sparse_grid(), NonDExpansion::expansionCoeffsApproach, NonDExpansion::initialize(), NonDPolynomialChaos::initialize_u_space_model(), Iterator::iteratedModel, NonD::numContDesVars, NonD::numContEpistUncVars, NonD::numContStateVars, Iterator::outputLevel, NonDPolynomialChaos::resolve_inputs(), NonD::transform_model(), and NonDExpansion::uSpaceModel.

13.104.3 Member Function Documentation

void increment_specification_sequence ( ) `[protected], [virtual]`

increment the input specification sequence (PCE only)

default implementation is overridden by PCE

Reimplemented from NonDExpansion.

References NonDPolynomialChaos::collocPtsSeqSpec, NonDPolynomialChaos::collocRatio, SharedApproxData::data_rep(), NonDIntegration::dimension_preference_to_anisotropic_order(), NonDPolynomialChaos::dimPrefSpec, SharedPecosApproxData::expansion_order(), NonDExpansion::expansionBasisType, NonDExpansion::expansionCoeffsApproach, NonDPolynomialChaos::expOrderSeqSpec, NonDPolynomialChaos::expSamplesSeqSpec, NonDExpansion::increment_specification_sequence(), Iterator::iterator_rep(), NonDQuadrature::mode(), Model::model_rep(), Analyzer::numContinuousVars, NonDExpansion::numSamplesOnModel, NonDQuadrature::quadrature_order(), NonDQuadrature::samples(), Iterator::sampling_reference(), NonDPolynomialChaos::sequenceIndex, Model::shared_approximation(), Model::subordinate_iterator(), NonDPolynomialChaos::tensorRegression, NonDPolynomialChaos::terms_ratio_to_samples(), NonDPolynomialChaos::termsOrder, DataFitSurrModel::total_points(), NonDQuadrature::update(), and NonDExpansion::uSpaceModel.

void increment_order ( ) [protected], [virtual]

uniformly increment the order of the polynomial chaos expansion

Used for uniform refinement of regression-based PCE.

Reimplemented from NonDExpansion.

References NonDPolynomialChaos::collocRatio, SharedApproxData::data_rep(), SharedPecosApproxData::expansion_order(), NonDExpansion::expansionBasisType, NonDQuadrature::increment_grid(), SharedPecosApproxData::increment_order(), Iterator::iterator_rep(), NonDQuadrature::mode(), Model::model_rep(), NonDExpansion::numSamplesOnModel, NonDQuadrature::samples(), Iterator::sampling_reference(), Model::shared_approximation(), Model::subordinate_iterator(), NonDPolynomialChaos::tensorRegression, NonDPolynomialChaos::terms_ratio_to_samples(), NonDPolynomialChaos::termsOrder, DataFitSurrModel::total_points(), NonDQuadrature::update(), and NonDExpansion::uSpaceModel.

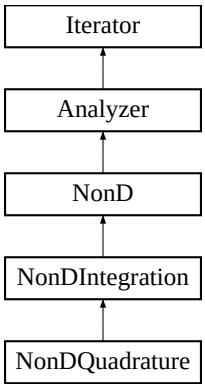
The documentation for this class was generated from the following files:

- NonDPolynomialChaos.hpp
- NonDPolynomialChaos.cpp

13.105 NonDQuadrature Class Reference

Derived nondeterministic class that generates N-dimensional numerical quadrature points for evaluation of expectation integrals over uncorrelated standard normals/uniforms/exponentials/betas/gammas.

Inheritance diagram for NonDQuadrature:



Public Member Functions

- NonDQuadrature (Model &model, const UShortArray &quad_order_seq, const RealVector &dim_pref, short driver_mode)

  alternate constructor for instantiations "on the fly" based on a quadrature order specification
- NonDQuadrature (Model &model, int num_filt_samples, const RealVector &dim_pref, short driver_mode)

  alternate constructor for instantiations "on the fly" that generate a filtered tensor product sample set
- NonDQuadrature (Model &model, int num_rand_samples, int seed, const UShortArray &quad_order_seq, const RealVector &dim_pref, short driver_mode)

  alternate constructor for instantiations "on the fly" that sample randomly from a tensor product multi-index
- void increment_grid ()

  increment SSG level/TPQ order

- void update ()

  propagate any numSamples updates and/or grid updates/increments
- const Pecos::UShortArray & quadrature_order () const

  return Pecos::TensorProductDriver::quadOrder
- void quadrature_order (const Pecos::UShortArray &dim_quad_order)

  set dimQuadOrderRef and map to Pecos::TensorProductDriver::quadOrder
- void samples (size_t samples)

  set numSamples
- short mode () const

  return quadMode

Protected Member Functions

- NonDQuadrature (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDQuadrature ()

  destructor
- void initialize_grid (const std::vector< Pecos::BasisPolynomial > &poly_basis)
- void get_parameter_sets (Model &model)

  Returns one block of samples (ndim ∗ num_samples)
- void reset ()

  restore initial state for repeated sub-iterator executions
- void sampling_reset (int min_samples, bool all_data_flag, bool stats_flag)
- void increment_grid_preference (const RealVector &dim_pref)

  increment SSG level/TPQ order and update anisotropy
- void increment_specification_sequence ()

  increment sequenceIndex and update active orders/levels
- int num_samples () const

Private Member Functions

- void increment_grid (UShortArray &dim_quad_order)

  convenience function used to make increment_grid() more modular
- void increment_grid_preference (const RealVector &dim_pref, UShortArray &dim_quad_order)

  convenience function used to make increment_grid_preference() more modular
- void compute_minimum_quadrature_order (size_t min_samples, const RealVector &dim_pref, UShortArray &dim_quad_order)

  calculate smallest dim_quad_order with at least min_samples
- void filter_parameter_sets ()

  prune allSamples back to size numSamples, retaining points with highest product weight
- void update_anisotropic_order (const RealVector &dim_pref, UShortArray &quad_order_ref)

  update quad_order_ref based on an updated dimension preference, enforcing previous values as a lower bound
- void initialize_dimension_quadrature_order (unsigned short quad_order_spec, const RealVector &dim_pref_spec, UShortArray &dim_quad_order)

initialize dim_quad_order from quad_order_spec and dim_pref_spec

- void increment_dimension_quadrature_order (UShortArray &dim_quad_order)

  increment each dim_quad_order entry by 1

- void increment_dimension_quadrature_order (const RealVector &dim_pref, UShortArray &dim_quad_order)

  increment the dim_quad_order entry with maximum preference by 1 and then rebalance

Private Attributes

- Pecos::TensorProductDriver * tpqDriver

  convenience pointer to the numIntDriver representation

- bool nestedRules

  for studies involving refinement strategies, allow for use of nested quadrature rules such as Gauss-Patterson

- UShortArray quadOrderSeqSpec

  a sequence of scalar quadrature orders, one per refinement level

- UShortArray dimQuadOrderRef

  reference point for Pecos::TensorProductDriver::quadOrder: the original user specification for the number of Gauss points per dimension, plus any refinements posted by increment_grid()

- short quadMode

  point generation mode: FULL_TENSOR, FILTERED_TENSOR, RANDOM_TENSOR

- size_t numSamples

  size of a subset of tensor quadrature points (filtered based on product weight or sampled uniformly from the tensor multi-index); used by the regression PCE approach known as "probabilistic collocation"

- int randomSeed

  seed for the random number generator used in sampling of the tensor multi-index

Additional Inherited Members

13.105.1 Detailed Description

Derived nondeterministic class that generates N-dimensional numerical quadrature points for evaluation of expectation integrals over uncorrelated standard normals/uniforms/exponentials/betas/gammas.

This class is used by NonDPolynomialChaos, but could also be used for general numerical integration of moments. It employs Gauss-Hermite, Gauss-Legendre, Gauss-Laguerre, Gauss-Jacobi and generalized Gauss--Laguerre quadrature for use with normal, uniform, exponential, beta, and gamma density functions and integration bounds. The abscissas and weights for one-dimensional integration are extracted from the appropriate Orthogonal-Polynomial class and are extended to n-dimensions using a tensor product approach.

13.105.2 Constructor & Destructor Documentation

NonDQuadrature (Model & *model*, const UShortArray & *quad_order_seq*, const RealVector & *dim_pref*, short *driver_mode*)

alternate constructor for instantiations "on the fly" based on a quadrature order specification

This alternate constructor is used for on-the-fly generation and evaluation of numerical quadrature points.

References NonDIntegration::numIntDriver, and NonDQuadrature::tpqDriver.

NonDQuadrature (Model & *model,* int *num_filt_samples,* const RealVector & *dim_pref,* short *driver_mode*)

alternate constructor for instantiations "on the fly" that generate a filtered tensor product sample set

This alternate constructor is used for on-the-fly generation and evaluation of filtered tensor quadrature points.

References NonDIntegration::numIntDriver, and NonDQuadrature::tpqDriver.

NonDQuadrature (Model & *model,* int *num_rand_samples,* int *seed,* const UShortArray & *quad_order_seq,* const RealVector & *dim_pref,* short *driver_mode*)

alternate constructor for instantiations "on the fly" that sample randomly from a tensor product multi-index

This alternate constructor is used for on-the-fly generation and evaluation of random sampling from a tensor quadrature multi-index.

References NonDIntegration::numIntDriver, and NonDQuadrature::tpqDriver.

NonDQuadrature (ProblemDescDB & *problem_db,* Model & *model*) `[protected]`

constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification. It is not currently used, as there is not yet a separate nond_quadrature method specification.

References NonDIntegration::check_variables(), Iterator::convergenceTol, ProblemDescDB::get_bool(), ProblemDescDB::get_short(), ProblemDescDB::get_ushort(), Iterator::maxEvalConcurrency, Iterator::maxIterations, NonD::natafTransform, NonDQuadrature::nestedRules, NonDIntegration::numIntDriver, Iterator::outputLevel, Iterator::probDescDB, NonDQuadrature::reset(), and NonDQuadrature::tpqDriver.

13.105.3 Member Function Documentation

void initialize_grid (const std::vector< Pecos::BasisPolynomial > & *poly_basis*) `[protected], [virtual]`

Used in combination with alternate NonDQuadrature constructor.

Implements NonDIntegration.

References Iterator::maxEvalConcurrency, NonDQuadrature::nestedRules, Analyzer::numContinuousVars, NonDQuadrature::numSamples, NonDQuadrature::quadMode, NonDQuadrature::reset(), NonDQuadrature::tpqDriver, and NonDQuadrature::update().

void sampling_reset (int *min_samples,* bool *all_data_flag,* bool *stats_flag*) `[protected], [virtual]`

used by DataFitSurrModel::build_global() to publish the minimum number of points needed from the quadrature routine in order to build a particular global approximation.

Reimplemented from Iterator.

References NonDQuadrature::compute_minimum_quadrature_order(), NonDIntegration::dimPrefSpec, NonDQuadrature::dimQuadOrderRef, NonDQuadrature::nestedRules, Analyzer::numContinuousVars, and NonDQuadrature::tpqDriver.

Referenced by NonDQuadrature::update().

int num_samples ( ) const `[inline], [protected], [virtual]`

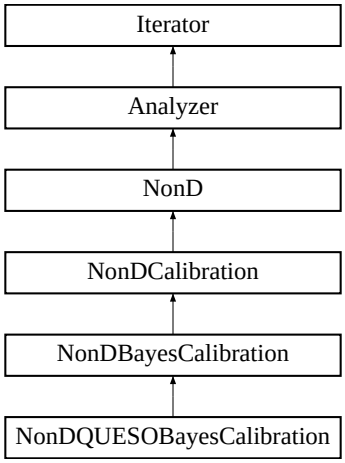
Return current number of evaluation points. Since the calculation of samples, collocation points, etc. might be costly, provide a default implementation here that backs out from the maxEvalConcurrency.

Reimplemented from Analyzer.
References NonDQuadrature::numSamples, NonDQuadrature::quadMode, and NonDQuadrature::tpqDriver.
The documentation for this class was generated from the following files:

- NonDQuadrature.hpp
- NonDQuadrature.cpp

13.106 NonDQUESOBayesCalibration Class Reference

Bayesian inference using the QUESO library from UT Austin.
Inheritance diagram for NonDQUESOBayesCalibration:



Public Member Functions

- NonDQUESOBayesCalibration (ProblemDescDB &problem_db, Model &model)

  standard constructor
- ~NonDQUESOBayesCalibration ()

  destructor

Public Attributes

- String mcmcType

  MCMC type (DRAM or Multilevel, both within QUESO)
- String rejectionType

  Rejection type (standard or delayed, in the DRAM framework)
- String metropolisType

  Metropolis type (hastings or adaptive, in the DRAM framework)
- int numSamples

  number of samples in the chain (e.g. number of MCMC samples)
- RealVector proposalCovScale

  scale factor for proposal covariance

- Real likelihoodScale

  scale factor for likelihood
- bool calibrateSigmaFlag

  flag to indicated if the sigma terms should be calibrated (default true)

Protected Member Functions

- void quantify_uncertainty ()

  redefined from DakotaNonD

Static Protected Member Functions

- static double dakotaLikelihoodRoutine (const QUESO::GslVector &paramValues, const QUESO::GslVector ∗paramDirection, const void ∗functionDataPtr, QUESO::GslVector ∗gradVector, QUESO::GslMatrix ∗hessianMatrix, QUESO::GslVector ∗hessianEffect)

  Likelihood function for call-back from QUESO to DAKOTA for evaluation.

Protected Attributes

- int randomSeed

  random seed to pass to QUESO

Private Attributes

- short emulatorType

  the emulator type: NO_EMULATOR, GP_EMULATOR, PCE_EMULATOR, or SC_EMULATOR

Static Private Attributes

- static NonDQUESOBayesCalibration ∗ NonDQUESOInstance

  Pointer to current class instance for use in static callback functions.

Additional Inherited Members

13.106.1 Detailed Description

Bayesian inference using the QUESO library from UT Austin.

This class provides a wrapper to the QUESO library developed as part of the Predictive Science Academic Alliance Program (PSAAP), specifically the PECOS (Predictive Engineering and Computational Sciences) Center at UT Austin. The name QUESO stands for Quantification of Uncertainty for Estimation, Simulation, and Optimization.

13.106.2 Constructor & Destructor Documentation

NonDQUESOBayesCalibration (ProblemDescDB & *problem_db*, Model & *model*)

standard constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

13.106.3 Member Function Documentation

void quantify_uncertainty ( ) [protected], [virtual]

redefined from DakotaNonD

Perform the uncertainty quantification

Reimplemented from NonDBayesCalibration.

References NonDQUESOBayesCalibration::calibrateSigmaFlag, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), NonDQUESOBayesCalibration::dakotaLikelihoodRoutine(), NonDBayesCalibration::emulatorModel, NonDQUESOBayesCalibration::emulatorType, NonDCalibration::expData, NonDCalibration::expDataFileAnnotated, NonDCalibration::expDataFileName, Iterator::iterator_rep(), ExperimentData::load_scalar(), NonDQUESOBayesCalibration::mcmcType, NonDQUESOBayesCalibration::metropolisType, NonDQUESOBayesCalibration::NonDQUESOInstance, Analyzer::numContinuousVars, NonDCalibration::numExpConfigVars, NonDCalibration::numExperiments, NonDCalibration::numExpStdDeviationsRead, Analyzer::numFunctions, NonDQUESOBayesCalibration::numSamples, Iterator::outputLevel, NonDQUESOBayesCalibration::proposalCovScale, NonDBayesCalibration::quantify_uncertainty(), NonDQUESOBayesCalibration::randomSeed, NonDQUESOBayesCalibration::rejectionType, ExperimentData::scalar_sigma(), NonDBayesCalibration::standardizedSpace, NonDBayesCalibration::stochExpIterator, and Dakota::strends().

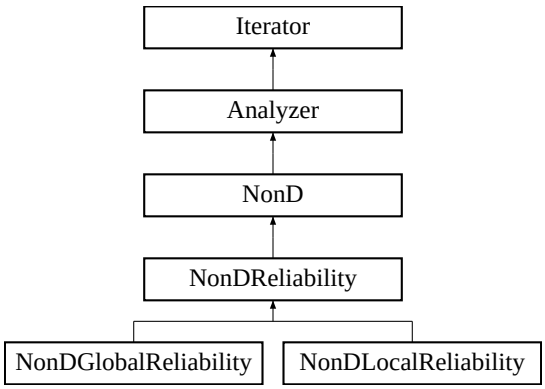
The documentation for this class was generated from the following files:

- NonDQUESOBayesCalibration.hpp
- NonDQUESOBayesCalibration.cpp

13.107 NonDReliability Class Reference

Base class for the reliability methods within DAKOTA/UQ.

Inheritance diagram for NonDReliability:



Protected Member Functions

- NonDReliability (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDReliability ()

  destructor
- void initialize_graphics (int iterator_server_id=1)

initialize graphics customized for reliability methods

- const Model & algorithm_space_model () const

Protected Attributes

- Model uSpaceModel

  Model representing the limit state in u-space, after any recastings and data fits.

- Model mppModel

  RecastModel which formulates the optimization subproblem: RIA, PMA, EGO.

- Iterator mppOptimizer

  Iterator which optimizes the mppModel.

- unsigned short mppSearchType

  the MPP search type selection: MV, x/u-space AMV, x/u-space AMV+, x/u-space TANA, x/u-space EGO, or NO_APPROX

- Iterator importanceSampler

  importance sampling instance used to compute/refine probabilities

- unsigned short integrationRefinement

  integration refinement type (NO_INT_REFINE, IS, AIS, or MMAIS) provided by `refinement` *specification*

- size_t numRelAnalyses

  number of invocations of quantify_uncertainty()

- size_t approxIters

  number of approximation cycles for the current respFnCount/levelCount

- bool approxConverged

  indicates convergence of approximation-based iterations

- int respFnCount

  counter for which response function is being analyzed

- size_t levelCount

  counter for which response/probability level is being analyzed

- size_t statCount

  counter for which final statistic is being computed

- bool pmaMaximizeG

  flag indicating maximization of G(u) within PMA formulation

- Real requestedTargetLevel

  the {response,reliability,generalized reliability} level target for the current response function

Additional Inherited Members

13.107.1 Detailed Description

Base class for the reliability methods within DAKOTA/UQ.

The NonDReliability class provides a base class for NonDLocalReliability, which implements traditional MPP-based reliability methods, and NonDGlobalReliability, which implements global limit state search using Gaussian process models in combination with multimodal importance sampling.

13.107.2 Member Function Documentation

const Model & algorithm_space_model ( ) const `[inline], [protected], [virtual]`

default definition that gets redefined in selected derived Minimizers

Reimplemented from Analyzer.

References NonDReliability::uSpaceModel.

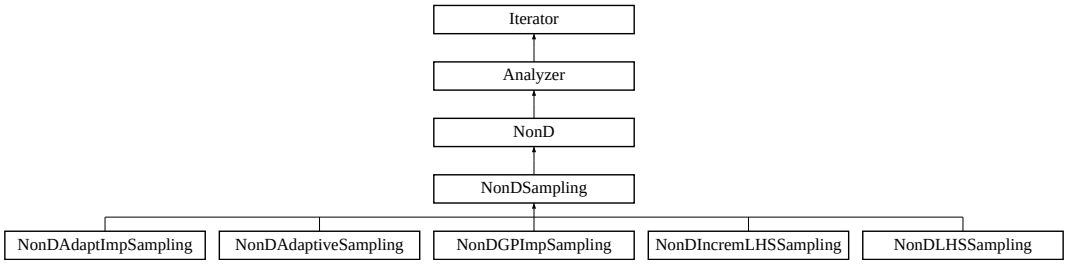
The documentation for this class was generated from the following files:

- NonDReliability.hpp
- NonDReliability.cpp

13.108 NonDSampling Class Reference

Base class for common code between NonDLHSSampling, NonDIncremLHSSampling, and NonDAdaptImpSampling.

Inheritance diagram for NonDSampling:



Public Member Functions

- void compute_distribution_mappings (const IntResponseMap &samples)

  called by compute_statistics() to calculate CDF/CCDF mappings of z to p/beta and of p/beta to z
- void update_final_statistics ()

  update finalStatistics from minValues/maxValues, momentStats, and computedProbLevels/computedRelLevels/computedRespLevels
- void print_pdf_mappings (std::ostream &s) const

  prints the PDFs computed in compute_statistics()

Protected Member Functions

- NonDSampling (ProblemDescDB &problem_db, Model &model)

  constructor
- NonDSampling (unsigned short method_name, Model &model, unsigned short sample_type, int samples, int seed, const String &rng, bool vary_pattern, short sampling_vars_mode)

  alternate constructor for sample generation and evaluation "on the fly"
- NonDSampling (unsigned short sample_type, int samples, int seed, const String &rng, const RealVector &lower_bnds, const RealVector &upper_bnds)

  alternate constructor for sample generation "on the fly"

- ~NonDSampling ()

  destructor
- int num_samples () const
- void sampling_reset (int min_samples, bool all_data_flag, bool stats_flag)

  resets number of samples and sampling flags
- void sampling_reference (int samples_ref)

  set reference number of samples, which is a lower bound during reset
- unsigned short sampling_scheme () const

  return sampleType
- void vary_pattern (bool pattern_flag)

  set varyPattern
- void get_parameter_sets (Model &model)

  Uses lhsDriver to generate a set of samples from the distributions/bounds defined in the incoming model.
- void get_parameter_sets (const RealVector &lower_bnds, const RealVector &upper_bnds)

  Uses lhsDriver to generate a set of uniform samples over lower_bnds/upper_bnds.
- void update_model_from_sample (Model &model, const Real *sample_vars)

  Override default update of continuous vars only.
- void sample_to_variables (const Real *sample_vars, Variables &vars)

  override default mapping of continuous variables only
- void variables_to_sample (const Variables &vars, Real *sample_vars)
- void initialize_lhs (bool write_message)

  increments numLHSRuns, sets random seed, and initializes lhsDriver
- void compute_statistics (const RealMatrix &vars_samples, const IntResponseMap &resp_samples)

  For the input sample set, computes mean, standard deviation, and probability/reliability/response levels (aleatory uncertainties) or intervals (epsitemic or mixed uncertainties)
- void compute_intervals (const IntResponseMap &samples)

  called by compute_statistics() to calculate min/max intervals
- void compute_moments (const IntResponseMap &samples)

  called by compute_statistics() to calculate means, std deviations, and confidence intervals
- void print_statistics (std::ostream &s) const

  prints the statistics computed in compute_statistics()
- void print_intervals (std::ostream &s) const

  prints the intervals computed in compute_intervals()
- void print_moments (std::ostream &s) const

  prints the moments computed in compute_moments()
- void view_design_counts (const Model &model, size_t &num_cdv, size_t &num_ddiv, size_t &num_ddsv, size_t &num_ddrv) const

  compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model
- void view_aleatory_uncertain_counts (const Model &model, size_t &num_cauv, size_t &num_dauiv, size_t &num_dausv, size_t &num_daurv) const

  compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model

- void view_epistemic_uncertain_counts (const Model &model, size_t &num_ceuv, size_t &num_deuiv, size_t &num_deusv, size_t &num_deurv) const

  compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model
- void view_uncertain_counts (const Model &model, size_t &num_cuv, size_t &num_duiv, size_t &num_dusv, size_t &num_durv) const

  compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model
- void view_state_counts (const Model &model, size_t &num_csv, size_t &num_dsiv, size_t &num_dssv, size_t &num_dsrv) const

  compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model
- void mode_counts (const Model &model, size_t &cv_start, size_t &num_cv, size_t &div_start, size_t &num_div, size_t &dsv_start, size_t &num_dsv, size_t &drv_start, size_t &num_drv) const

  compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model

Protected Attributes

- const int seedSpec

  the user seed specification (default is 0)
- int randomSeed

  the current seed
- const int samplesSpec

  initial specification of number of samples
- int samplesRef

  reference number of samples updated for refinement
- int numSamples

  the current number of samples to evaluate
- String rngName

  name of the random number generator
- unsigned short sampleType

  `the sample type: default, random, lhs,`
  < incremental random, or incremental lhs
- Pecos::LHSDriver lhsDriver

  the C++ wrapper for the F90 LHS library
- bool statsFlag

  flags computation/output of statistics
- bool allDataFlag

  `flags update of allResponses`
  < (allVariables or allSamples already defined)
- short samplingVarsMode

  the sampling mode: ALEATORY_UNCERTAIN{,_UNIFORM}, EPISTEMIC_UNCERTAIN{,_UNIFORM}, UNCERTAIN{,_UNIFORM}, ACTIVE{,_UNIFORM}, or ALL{,_UNIFORM}. This is a secondary control on top of the variables view that allows sampling over subsets of variables that may differ from the view.
- short sampleRanksMode

mode for input/output of LHS sample ranks: IGNORE_RANKS, GET_RANKS, SET_RANKS, or SET_GET_RANKS

- bool varyPattern

  flag for generating a sequence of seed values within multiple get_parameter_sets() calls so that these executions (e.g., for SBO/SBNLS) are not repeated, but are still repeatable

- RealMatrix sampleRanks

  data structure to hold the sample ranks

- SensAnalysisGlobal nonDSampCorr

  initialize statistical post processing

- bool backfillFlag

  flags whether to use backfill to enforce uniqueness of discrete LHS samples

Private Member Functions

- void archive_allocate_pdf ()

  allocate results array storage for pdf histograms

- void archive_pdf (size_t fn_index)

  archive a single pdf histogram for specified function

Private Attributes

- size_t numLHSRuns

  counter for number of executions of get_parameter_sets() for this object

- RealMatrix momentCIs

  Matrix of confidence internals on moments, with rows for mean_lower, mean_upper, sd_lower, sd_upper (calculated in compute_moments())

- RealMatrix extremeValues

  Minimum (row 0) and maximum (row 1) values of response functions for epistemic calculations (calculated in compute_intervals()),.

- RealVectorArray computedPDFAbscissas

  sorted response PDF intervals bounds extracted from min/max sample and requested/computedRespLevels (vector lengths = num bins + 1)

- RealVectorArray computedPDFOrdinates

  response PDF densities computed from bin counts divided by (unequal) bin widths (vector lengths = num bins)

Additional Inherited Members

13.108.1 Detailed Description

Base class for common code between NonDLHSSampling, NonDIncremLHSSampling, and NonDAdaptImpSampling.

This base class provides common code for sampling methods which employ the Latin Hypercube Sampling (LHS) package from Sandia Albuquerque's Risk and Reliability organization. NonDSampling now exclusively utilizes the 1998 Fortran 90 LHS version as documented in SAND98-0210, which was converted to a UNIX link library in

1. The 1970's vintage LHS (that had been f2c'd and converted to incomplete classes) has been removed.

13.108.2 Constructor & Destructor Documentation

NonDSampling (ProblemDescDB & *problem_db*, Model & *model*) [protected]

constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification.

References Dakota::abort_handler(), NonD::epistemicStats, NonD::initialize_final_statistics(), Iterator::maxEvalConcurrency, NonDSampling::numSamples, NonDSampling::sampleType, and NonD::totalLevelRequests.

NonDSampling (unsigned short *method_name*, Model & *model*, unsigned short *sample_type*, int *samples*, int *seed*, const String & *rng*, bool *vary_pattern*, short *sampling_vars_mode*) [protected]

alternate constructor for sample generation and evaluation "on the fly"

This alternate constructor is used for generation and evaluation of on-the-fly sample sets.

References NonD::epistemicStats, Iterator::maxEvalConcurrency, NonD::numEpistemicUncVars, NonDSampling::numSamples, NonDSampling::sampleType, NonDSampling::samplingVarsMode, and Iterator::subIteratorFlag.

NonDSampling (unsigned short *sample_type*, int *samples*, int *seed*, const String & *rng*, const RealVector & *lower_bnds*, const RealVector & *upper_bnds*) [protected]

alternate constructor for sample generation "on the fly"

This alternate constructor is used by ConcurrentStrategy for generation of uniform, uncorrelated sample sets.

References Iterator::maxEvalConcurrency, NonDSampling::numSamples, NonDSampling::sampleType, and Iterator::subIteratorFlag.

13.108.3 Member Function Documentation

int num_samples ( ) const [inline], [protected], [virtual]

Return current number of evaluation points. Since the calculation of samples, collocation points, etc. might be costly, provide a default implementation here that backs out from the maxEvalConcurrency.

Reimplemented from Analyzer.

References NonDSampling::numSamples.

Referenced by NonDAdaptImpSampling::evaluate_samples(), and NonDAdaptImpSampling::select_rep_points().

void sampling_reset (int *min_samples*, bool *all_data_flag*, bool *stats_flag*) [inline], [protected], [virtual]

resets number of samples and sampling flags

used by DataFitSurrModel::build_global() to publish the minimum number of samples needed from the sampling routine (to build a particular global approximation) and to set allDataFlag and statsFlag. In this case, allDataFlag is set to true (vectors of variable and response sets must be returned to build the global approximation) and statsFlag is set to false (statistics computations are not needed).

Reimplemented from Iterator.

References NonDSampling::allDataFlag, NonDSampling::numSamples, NonDSampling::samplesRef, and NonDSampling::statsFlag.

void get_parameter_sets (Model & *model*) [protected], [virtual]

Uses lhsDriver to generate a set of samples from the distributions/bounds defined in the incoming model.

This version of get_parameter_sets() extracts data from the user-defined model in any of the four sampling modes.

Reimplemented from Analyzer.

References Dakota::abort_handler(), Model::acv(), Model::adiv(), Model::aleatory_distribution_parameters(), Model::all_continuous_lower_bounds(), Model::all_continuous_upper_bounds(), Model::all_discrete_int_lower_bounds(), Model::all_discrete_int_upper_bounds(), Analyzer::allSamples, NonDSampling::backfillFlag, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::current_variables(), Model::discrete_design_set_int_values(), Model::discrete_design_set_real_values(), Model::discrete_design_set_string_values(), Model::discrete_state_set_int_values(), Model::discrete_state_set_real_values(), Model::discrete_state_set_string_values(), Model::epistemic_distribution_parameters(), NonDSampling::initialize_lhs(), NonDSampling::lhsDriver, NonDSampling::mode_counts(), NonD::numContDesVars, NonD::numContStateVars, NonD::numDiscIntDesVars, NonD::numDiscIntStateVars, NonD::numDiscRealDesVars, NonD::numDiscRealStateVars, NonD::numDiscStringDesVars, NonD::numDiscStringStateVars, NonDSampling::numSamples, NonDSampling::sampleRanks, NonDSampling::samplingVarsMode, and Variables::view().

Referenced by NonDLHSSampling::NonDLHSSampling(), NonDLHSSampling::pre_run(), NonDIncremLHSSampling::quantify_uncertainty(), and NonDAdaptImpSampling::quantify_uncertainty().

void get_parameter_sets (const RealVector & *lower_bnds,* const RealVector & *upper_bnds*) [protected]

Uses lhsDriver to generate a set of uniform samples over lower_bnds/upper_bnds.

This version of get_parameter_sets() does not extract data from the user-defined model, but instead relies on the incoming bounded region definition. It only support a UNIFORM sampling mode, where the distinction of ACTIVE_UNIFORM vs. ALL_UNIFORM is handled elsewhere.

References Analyzer::allSamples, NonDSampling::initialize_lhs(), NonDSampling::lhsDriver, and NonDSampling::numSamples.

void variables_to_sample (const Variables & *vars,* Real * *sample_vars*) [protected], [virtual]

Map the active variables from vars to sample_vars (column in allSamples)

Reimplemented from Analyzer.

References Variables::continuous_variables(), Variables::discrete_int_variables(), Variables::discrete_real_variables(), Model::discrete_set_string_values(), Variables::discrete_string_variables(), Iterator::iteratedModel, Analyzer::numContinuousVars, Analyzer::numDiscreteIntVars, Analyzer::numDiscreteRealVars, Analyzer::numDiscreteStringVars, Dakota::set_value_to_index(), and Variables::view().

void view_design_counts (const Model & *model,* size_t & *num_cdv,* size_t & *num_ddiv,* size_t & *num_ddsv,* size_t & *num_ddrv*) const [protected]

compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model

This function computes total design variable counts, not active counts, for use in defining offsets and counts within all variables arrays.

References Model::current_variables(), Variables::cv_start(), SharedVariablesData::design_counts(), Variables::div_start(), Variables::drv_start(), Variables::dsv_start(), NonD::numContDesVars, NonD::numDiscIntDesVars, NonD::numDiscRealDesVars, NonD::numDiscStringDesVars, Variables::shared_data(), and Variables::view().

Referenced by NonDSampling::mode_counts().

void view_aleatory_uncertain_counts (const Model & *model,* size_t & *num_cauv,* size_t & *num_dauiv,* size_t & *num_dausv,* size_t & *num_daurv*) const [protected]

compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model

This function computes total aleatory uncertain variable counts, not active counts, for use in defining offsets and counts within all variables arrays.

References SharedVariablesData::aleatory_uncertain_counts(), Model::current_variables(), NonD::numContAleatUncVars, NonD::numDiscIntAleatUncVars, NonD::numDiscRealAleatUncVars, NonD::numDiscStringAleatUncVars, Variables::shared_data(), and Variables::view().

Referenced by NonDSampling::mode_counts().

void view_epistemic_uncertain_counts (const Model & *model,* size_t & *num_ceuv,* size_t & *num_deuiv,* size_t & *num_deusv,* size_t & *num_deurv*) const [protected]

compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model

This function computes total epistemic uncertain variable counts, not active counts, for use in defining offsets and counts within all variables arrays.

References Model::current_variables(), SharedVariablesData::epistemic_uncertain_counts(), NonD::numContEpistUncVars, NonD::numDiscIntEpistUncVars, NonD::numDiscRealEpistUncVars, NonD::numDiscStringEpistUncVars, Variables::shared_data(), and Variables::view().

Referenced by NonDSampling::mode_counts().

void view_uncertain_counts (const Model & *model,* size_t & *num_cuv,* size_t & *num_duiv,* size_t & *num_dusv,* size_t & *num_durv*) const [protected]

compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model

This function computes total uncertain variable counts, not active counts, for use in defining offsets and counts within all variables arrays.

References Model::current_variables(), Variables::cv(), Variables::div(), Variables::drv(), Variables::dsv(), NonD::numContAleatUncVars, NonD::numContEpistUncVars, NonD::numDiscIntAleatUncVars, NonD::numDiscIntEpistUncVars, NonD::numDiscRealAleatUncVars, NonD::numDiscRealEpistUncVars, NonD::numDiscStringAleatUncVars, NonD::numDiscStringEpistUncVars, Variables::shared_data(), SharedVariablesData::uncertain_counts(), and Variables::view().

Referenced by NonDSampling::mode_counts().

void mode_counts (const Model & *model,* size_t & *cv_start,* size_t & *num_cv,* size_t & *div_start,* size_t & *num_div,* size_t & *dsv_start,* size_t & *num_dsv,* size_t & *drv_start,* size_t & *num_drv*) const [protected]

compute sampled subsets (all, active, uncertain) within all variables (acv/adiv/adrv) from samplingVarsMode and model

This function and its helpers to follow are needed since NonDSampling supports a richer set of sampling modes than just the active variable subset. mode_counts() manages the samplingVarsMode setting, while its helper functions (view_{design,aleatory_uncertain,epistemic_uncertain, uncertain,state}_counts) manage the active variables view. Similar to the computation of starts and counts in creating active variable views, the results of this function are starts and counts for use within model.all_*() set/get functions.

References Model::acv(), Model::adiv(), Model::adrv(), Model::adsv(), Model::current_variables(), Variables::cv(), Variables::cv_start(), Variables::div(), Variables::div_start(), Variables::drv(), Variables::drv_start(), Variables::dsv(), Variables::dsv_start(), NonDSampling::samplingVarsMode, NonDSampling::view_aleatory_uncertain_counts(), NonDSampling::view_design_counts(), NonDSampling::view_epistemic_uncertain_counts(), and NonDSampling::view_uncertain_counts().

Referenced by NonDSampling::compute_statistics(), NonDSampling::get_parameter_sets(), NonDLHSSampling::post_input(), NonDSampling::print_statistics(), NonDSampling::sample_to_variables(), and NonDSampling::update_model_from_sample().

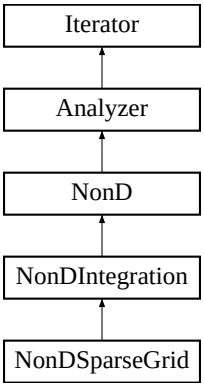
The documentation for this class was generated from the following files:

- NonDSampling.hpp
- NonDSampling.cpp

13.109 NonDSparseGrid Class Reference

Derived nondeterministic class that generates N-dimensional Smolyak sparse grids for numerical evaluation of expectation integrals over independent standard random variables.

Inheritance diagram for NonDSparseGrid:



Public Member Functions

- NonDSparseGrid (Model &model, const UShortArray &ssg_level_seq, const RealVector &dim_pref, short exp_coeffs_soln_approach, short driver_mode, short growth_rate=Pecos::MODERATE_RESTRICTED_GROWTH, short refine_control=Pecos::NO_CONTROL, bool track_uniq_prod_wts=true, bool track_colloc_indices=true)
- void increment_grid ()

  increment ssgDriver::ssgLevel
- void increment_grid_weights (const RealVector &aniso_wts)

  update ssgDriver::ssgAnisoLevelWts and increment ssgDriver::ssgLevel based on specified anisotropic weighting
- void increment_specification_sequence ()

  advance to next nevel in ssgLevelSeqSpec sequence
- const std::set< UShortArray > & active_multi_index () const

  returns SparseGridDriver::active_multi_index()
- const std::set< UShortArray > & old_multi_index () const

returns SparseGridDriver::old_multi_index()

- void print_smolyak_multi_index () const

  invokes SparseGridDriver::print_smolyak_multi_index()
- void initialize_sets ()

  invokes SparseGridDriver::initialize_sets()
- void update_reference ()

  invokes SparseGridDriver::update_reference()
- void increment_set (const UShortArray &set)

  invokes SparseGridDriver::push_trial_set()
- int increment_size () const

  invokes SparseGridDriver::unique_trial_points()
- void restore_set ()

  invokes SparseGridDriver::restore_set()
- void evaluate_set ()

  invokes SparseGridDriver::compute_trial_grid()
- void decrement_set ()

  invokes SparseGridDriver::pop_trial_set()
- void update_sets (const UShortArray &set_star)

  invokes SparseGridDriver::update_sets()
- void print_final_sets (bool converged_within_tol)

  invokes SparseGridDriver::print_final_sets(bool)
- void finalize_sets ()

  invokes SparseGridDriver::finalize_sets()
- void evaluate_grid_increment ()

  invokes SparseGridDriver::evaluate_grid_increment()
- int num_samples () const

Protected Member Functions

- NonDSparseGrid (ProblemDescDB &problem_db, Model &model)

  constructor
- ~NonDSparseGrid ()

  destructor
- void initialize_grid (const std::vector< Pecos::BasisPolynomial > &poly_basis)

  initialize integration grid by drawing from polynomial basis settings
- void get_parameter_sets (Model &model)

  *Returns one block of samples (ndim * num_samples)*
- void reset ()

  restore initial state for repeated sub-iterator executions
- void sampling_reset (int min_samples, bool all_data_flag, bool stats_flag)

Private Attributes

- Pecos::SparseGridDriver * ssgDriver

  convenience pointer to the numIntDriver representation
- UShortArray ssgLevelSeqSpec

  the user specification for the Smolyak sparse grid level, defining a sequence of refinement levels.
- unsigned short ssgLevelRef

  reference point (e.g., lower bound) for the Smolyak sparse grid level maintained within ssgDriver

Additional Inherited Members

13.109.1 Detailed Description

Derived nondeterministic class that generates N-dimensional Smolyak sparse grids for numerical evaluation of expectation integrals over independent standard random variables.

This class is used by NonDPolynomialChaos and NonDStochCollocation, but could also be used for general numerical integration of moments. It employs 1-D Clenshaw-Curtis and Gaussian quadrature rules within Smolyak sparse grids.

13.109.2 Constructor & Destructor Documentation

NonDSparseGrid (Model & *model*, const UShortArray & *ssg_level_seq*, const RealVector & *dim_pref*, short *exp_coeffs_soln_approach*, short *driver_mode*, short *growth_rate* = `Pecos::MODERATE_RESTRICTED_GROWTH`, short *refine_control* = `Pecos::NO_CONTROL`, bool *track_uniq_prod_wts* = `true`, bool *track_colloc_indices* = `true`)

This alternate constructor is used for on-the-fly generation and evaluation of sparse grids within PCE and SC.

References NonDIntegration::numIntDriver, and NonDSparseGrid::ssgDriver.

NonDSparseGrid (ProblemDescDB & *problem_db*, Model & *model*) `[protected]`

constructor

This constructor is called for a standard letter-envelope iterator instantiation. In this case, set_db_list_nodes has been called and probDescDB can be queried for settings from the method specification. It is not currently used, as there is not a separate sparse_grid method specification.

References Model::aleatory_distribution_parameters(), NonDIntegration::check_variables(), Iterator::convergenceTol, NonDIntegration::dimPrefSpec, ProblemDescDB::get_bool(), ProblemDescDB::get_short(), ProblemDescDB::get_ushort(), Iterator::iteratedModel, Iterator::maxEvalConcurrency, Iterator::maxIterations, NonD::nataf-Transform, NonDIntegration::numIntDriver, Iterator::outputLevel, Iterator::probDescDB, NonDSparseGrid::ssgDriver, and NonDSparseGrid::ssgLevelRef.

13.109.3 Member Function Documentation

int num_samples ( ) const `[inline], [virtual]`

Return current number of evaluation points. Since the calculation of samples, collocation points, etc. might be costly, provide a default implementation here that backs out from the maxEvalConcurrency.

Reimplemented from Analyzer.

References NonDSparseGrid::ssgDriver.

void sampling_reset (int *min_samples*, bool *all_data_flag*, bool *stats_flag*) [protected], [virtual]

used by DataFitSurrModel::build_global() to publish the minimum number of points needed from the sparse grid routine in order to build a particular global approximation.

Reimplemented from Iterator.

References NonDSparseGrid::ssgDriver, and NonDSparseGrid::ssgLevelRef.

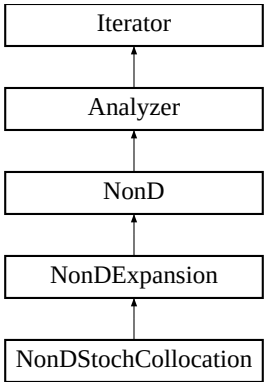
The documentation for this class was generated from the following files:

- NonDSparseGrid.hpp
- NonDSparseGrid.cpp

13.110 NonDStochCollocation Class Reference

Nonintrusive stochastic collocation approaches to uncertainty quantification.

Inheritance diagram for NonDStochCollocation:



Public Member Functions

- NonDStochCollocation (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NonDStochCollocation (Model &model, short exp_coeffs_approach, unsigned short num_int_level, short u_space_type, bool piecewise_basis, bool use_derivs)

  alternate constructor
- ~NonDStochCollocation ()

  destructor

Protected Member Functions

- void resolve_inputs (short &u_space_type, short &data_order)

  perform error checks and mode overrides
- void initialize_u_space_model ()

  initialize uSpaceModel polynomial approximations with PCE/SC data
- void update_expansion ()

  update an expansion; avoids overhead in compute_expansion()

- Real compute_covariance_metric ()

  compute 2-norm of change in response covariance
- Real compute_final_statistics_metric ()

  compute 2-norm of change in final statistics

Additional Inherited Members

13.110.1 Detailed Description

Nonintrusive stochastic collocation approaches to uncertainty quantification.

The NonDStochCollocation class uses a stochastic collocation (SC) approach to approximate the effect of parameter uncertainties on response functions of interest. It utilizes the InterpPolyApproximation class to manage multidimensional Lagrange polynomial interpolants.

13.110.2 Constructor & Destructor Documentation

NonDStochCollocation (ProblemDescDB & *problem_db*, Model & *model*)

standard constructor

This constructor is called for a standard letter-envelope iterator instantiation using the ProblemDescDB.

References Model::assign_rep(), ParallelLibrary::command_line_check(), NonDExpansion::construct_expansion_sampler(), NonDExpansion::construct_quadrature(), NonDExpansion::construct_sparse_grid(), NonDExpansion::expansionBasisType, NonDExpansion::expansionCoeffsApproach, ProblemDescDB::get_bool(), ProblemDescDB::get_rv(), ProblemDescDB::get_short(), ProblemDescDB::get_string(), ProblemDescDB::get_usa(), NonDExpansion::initialize(), NonDStochCollocation::initialize_u_space_model(), Iterator::iteratedModel, NonDExpansion::nestedRules, NonD::numContDesVars, NonD::numContEpistUncVars, NonD::numContStateVars, NonDExpansion::numSamplesOnModel, Iterator::outputLevel, Iterator::parallelLib, NonDExpansion::piecewiseBasis, Iterator::probDescDB, NonDExpansion::refineControl, NonDStochCollocation::resolve_inputs(), NonD::transform_model(), and NonDExpansion::uSpaceModel.

NonDStochCollocation (Model & *model*, short *exp_coeffs_approach*, unsigned short *num_int_level*, short *u_space_type*, bool *piecewise_basis*, bool *use_derivs*)

alternate constructor

This constructor is used for helper iterator instantiation on the fly.

References Model::assign_rep(), NonDExpansion::construct_quadrature(), NonDExpansion::construct_sparse_grid(), NonDExpansion::expansionBasisType, NonDExpansion::expansionCoeffsApproach, NonDExpansion::initialize(), NonDStochCollocation::initialize_u_space_model(), Iterator::iteratedModel, NonD::numContDesVars, NonD::numContEpistUncVars, NonD::numContStateVars, Iterator::outputLevel, NonDExpansion::piecewiseBasis, NonDStochCollocation::resolve_inputs(), NonD::transform_model(), and NonDExpansion::uSpaceModel.

13.110.3 Member Function Documentation

Real compute_covariance_metric ( ) `[protected], [virtual]`

compute 2-norm of change in response covariance

computes the default refinement metric based on change in respCovariance

Reimplemented from NonDExpansion.

References Model::approximations(), NonDExpansion::compute_covariance_metric(), PecosApproximation::delta_covariance(), PecosApproximation::expansion_coefficient_flag(), NonDExpansion::expansionBasisType, NonDExpansion::initialPtU, NonD::numContDesVars, NonD::numContEpistUncVars, NonD::numContStateVars, Analyzer::numFunctions, NonDExpansion::respCovariance, and NonDExpansion::uSpaceModel.

Real compute_final_statistics_metric ( ) [protected], [virtual]

compute 2-norm of change in final statistics

computes a "goal-oriented" refinement metric employing finalStatistics

Reimplemented from NonDExpansion.

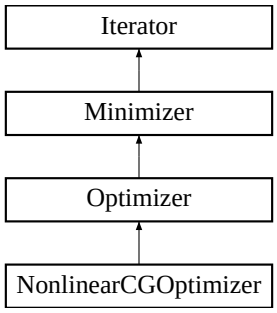
References Model::approximations(), NonD::cdfFlag, NonDExpansion::compute_final_statistics_metric(), NonDExpansion::compute_statistics(), PecosApproximation::delta_beta(), PecosApproximation::delta_z(), PecosApproximation::expansion_coefficient_flag(), NonDExpansion::expansionBasisType, NonD::finalStatistics, Response::function_values(), NonDExpansion::initialPtU, Response::num_functions(), NonD::numContDesVars, NonD::numContEpistUncVars, NonD::numContStateVars, Analyzer::numFunctions, NonD::requestedGenRelLevels, NonD::requestedProbLevels, NonD::requestedRelLevels, NonD::requestedRespLevels, NonD::respLevelTarget, and NonDExpansion::uSpaceModel.

The documentation for this class was generated from the following files:

- NonDStochCollocation.hpp
- NonDStochCollocation.cpp

13.111 NonlinearCGOptimizer Class Reference

Inheritance diagram for NonlinearCGOptimizer:



Public Member Functions

- NonlinearCGOptimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor

- ~NonlinearCGOptimizer ()

  destructor

- Real linesearch_eval (const Real &trial_step, short req_val=1)

  evaluate the objective function given a particular step size (public for use in boost_ls_eval functor; could use friend)

Protected Member Functions

- void find_optimum ()

  Used within the optimizer branch for computing the optimal solution. Redefines the run virtual function for the optimizer branch.

Private Member Functions

- void parse_options ()

  constructor helper function to parse misc_options from ProblemDescDB
- void compute_direction ()

  compute next direction via choice of method
- bool compute_step ()

  compute step: fixed, simple decrease, sufficient decrease
- void bracket_min (Real &xa, Real &xb, Real &xc, Real &fa, Real &fb, Real &fc)

  bracket the 1-D minimum in the linesearch
- Real brent_minimize (Real a, Real b, Real tol)

  Perform 1-D minimization for the stepLength using Brent's method.

Private Attributes

- Real initialStep

  initial step length
- Real linesearchTolerance

  approximate accuracy of absissca in LS
- unsigned linesearchType

  type of line search (if any)
- unsigned maxLinesearchIters

  maximum evaluations in line search
- Real relFunctionTol

  stopping criterion for rel change in fn
- Real relGradientTol

  stopping criterion for rel reduction in g
- bool resetStep

  whether to reset step with each linesearch
- unsigned restartIter

  iter at which to reset to steepest descent
- unsigned updateType

  type of CG direction update
- unsigned iterCurr

  current iteration number
- RealVector designVars

  current decision variables in the major iteration
- RealVector trialVars

  decision variables in the linesearch

- Real functionCurr

  current function value
- Real functionPrev

  previous function value
- RealVector gradCurr

  current gradient
- RealVector gradPrev

  previous gradient
- RealVector gradDiff

  temporary for gradient difference (gradCurr - gradPrev)
- RealVector searchDirection

  current aggregate search direction
- Real stepLength

  current step length parameter alpha
- Real gradDotGrad_init

  initial gradient norm squared
- Real gradDotGrad_curr

  gradCurr dot gradCurr
- Real gradDotGrad_prev

  gradPrev dot gradPrev

Additional Inherited Members

13.111.1 Detailed Description

Experimental implementation of nonlinear CG optimization

13.111.2 Member Function Documentation

Real brent_minimize (Real *a*, Real *b*, Real *tol*) `[private]`

Perform 1-D minimization for the stepLength using Brent's method.

Perform 1-D minimization for the stepLength using Brent's method. This is a C translation of fmin.f from Netlib.

References NonlinearCGOptimizer::linesearch_eval(), NonlinearCGOptimizer::maxLinesearchIters, and Iterator::outputLevel.

Referenced by NonlinearCGOptimizer::compute_step().

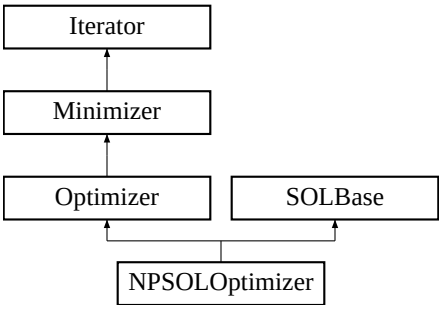
The documentation for this class was generated from the following files:

- NonlinearCGOptimizer.hpp
- NonlinearCGOptimizer.cpp

13.112 NPSOLOptimizer Class Reference

Wrapper class for the NPSOL optimization library.

Inheritance diagram for NPSOLOptimizer:



Public Member Functions

- NPSOLOptimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor
- NPSOLOptimizer (Model &model)

  alternate constructor for Iterator instantiations by name
- NPSOLOptimizer (Model &model, const int &derivative_level, const Real &conv_tol)

  alternate constructor for instantiations "on the fly"
- NPSOLOptimizer (const RealVector &initial_point, const RealVector &var_lower_bnds, const RealVector &var_upper_bnds, const RealMatrix &lin_ineq_coeffs, const RealVector &lin_ineq_lower_bnds, const RealVector &lin_ineq_upper_bnds, const RealMatrix &lin_eq_coeffs, const RealVector &lin_eq_targets, const RealVector &nonlin_ineq_lower_bnds, const RealVector &nonlin_ineq_upper_bnds, const RealVector &nonlin_eq_targets, void(∗user_obj_eval)(int &, int &, double ∗, double &, double ∗, int &), void(∗user_con_eval)(int &, int &, int &, int &, int ∗, double ∗, double ∗, double ∗, int &), const int &derivative_level, const Real &conv_tol)

  alternate constructor for instantiations "on the fly"
- ∼NPSOLOptimizer ()

  destructor
- void find_optimum ()

  Used within the optimizer branch for computing the optimal solution. Redefines the run virtual function for the optimizer branch.

Private Member Functions

- void find_optimum_on_model ()

  called by find_optimum for setUpType == "model"
- void find_optimum_on_user_functions ()

  called by find_optimum for setUpType == "user_functions"

Static Private Member Functions

- static void objective_eval (int &mode, int &n, double *x, double &f, double *gradf, int &nstate)

  OBJFUN in NPSOL manual: computes the value and first derivatives of the objective function (passed by function pointer to NPSOL).

Private Attributes

- String setUpType

  controls iteration mode: "model" (normal usage) or "user_functions" (user-supplied functions mode for "on the fly" instantiations). NonDReliability currently uses the user_functions mode.
- RealVector initialPoint

  holds initial point passed in for "user_functions" mode.
- RealVector lowerBounds

  holds variable lower bounds passed in for "user_functions" mode.
- RealVector upperBounds

  holds variable upper bounds passed in for "user_functions" mode.
- void(* userObjectiveEval)(int &, int &, double *, double &, double *, int &)

  holds function pointer for objective function evaluator passed in for "user_functions" mode.
- void(* userConstraintEval)(int &, int &, int &, int &, int *, double *, double *, double *, int &)

  holds function pointer for constraint function evaluator passed in for "user_functions" mode.

Static Private Attributes

- static NPSOLOptimizer * npsolInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.112.1 Detailed Description

Wrapper class for the NPSOL optimization library.

The NPSOLOptimizer class provides a wrapper for NPSOL, a Fortran 77 sequential quadratic programming library from Stanford University marketed by Stanford Business Associates. It uses a function pointer approach for which passed functions must be either global functions or static member functions. Any attribute used within static member functions must be either local to that function or accessed through a static pointer.

The user input mappings are as follows: `max_function_evaluations` is implemented directly in NPSOLOptimizer's evaluator functions since there is no NPSOL parameter equivalent, and `max_iterations`, `convergence_tolerance`, `output` verbosity, `verify_level`, `function_precision`, and `linesearch_tolerance` are mapped into NPSOL's "Major Iteration Limit", "Optimality Tolerance", "Major Print Level" (`verbose`: Major Print Level = 20; `quiet`: Major Print Level = 10), "Verify Level", "Function Precision", and "Linesearch Tolerance" parameters, respectively, using NPSOL's npoptn() subroutine (as wrapped by npoptn2() from the npoptn_wrapper.f file). Refer to [Gill, P.E., Murray, W., Saunders, M.A., and Wright, M.H., 1986] for information on NPSOL's optional input parameters and the npoptn() subroutine.

13.112.2 Constructor & Destructor Documentation

NPSOLOptimizer (ProblemDescDB & *problem_db,* Model & *model*)

standard constructor

This is the primary constructor. It accepts a Model reference.

References Minimizer::constraintTol, Iterator::convergenceTol, Model::fd_gradient_step_size(), ProblemDescDB::get_int(), ProblemDescDB::get_real(), Model::gradient_type(), Iterator::iteratedModel, Iterator::maxIterations, Iterator::outputLevel, Iterator::probDescDB, SOLBase::set_options(), Minimizer::speculativeFlag, and Minimizer::vendorNumericalGradFlag.

NPSOLOptimizer (Model & *model*)

alternate constructor for Iterator instantiations by name

This is an alternate constructor which accepts a Model but does not have a supporting method specification from the ProblemDescDB.

References Minimizer::constraintTol, Iterator::convergenceTol, Model::fd_gradient_step_size(), Model::gradient_type(), Iterator::iteratedModel, Iterator::maxIterations, Iterator::outputLevel, SOLBase::set_options(), Minimizer::speculativeFlag, and Minimizer::vendorNumericalGradFlag.

NPSOLOptimizer (Model & *model,* const int & *derivative_level,* const Real & *conv_tol*)

alternate constructor for instantiations "on the fly"

This is an alternate constructor for instantiations on the fly using a Model but no ProblemDescDB.

NPSOLOptimizer (const RealVector & *initial_point,* const RealVector & *var_lower_bnds,* const RealVector & *var_upper_bnds,* const RealMatrix & *lin_ineq_coeffs,* const RealVector & *lin_ineq_lower_bnds,* const RealVector & *lin_ineq_upper_bnds,* const RealMatrix & *lin_eq_coeffs,* const RealVector & *lin_eq_targets,* const RealVector & *nonlin_ineq_lower_bnds,* const RealVector & *nonlin_ineq_upper_bnds,* const RealVector & *nonlin_eq_targets,* void(∗)(int &, int &, double ∗, double &, double ∗, int &) *user_obj_eval,* void(∗)(int &, int &, int &, int &, int ∗, double ∗, double ∗, double ∗, int &) *user_con_eval,* const int & *derivative_level,* const Real & *conv_tol*)

alternate constructor for instantiations "on the fly"

This is an alternate constructor for performing an optimization using the passed in objective function and constraint function pointers.

References SOLBase::allocate_arrays(), SOLBase::allocate_workspace(), SOLBase::augment_bounds(), NPSOLOptimizer::lowerBounds, Minimizer::numContinuousVars, Minimizer::numLinearConstraints, Minimizer::numNonlinearConstraints, and NPSOLOptimizer::upperBounds.

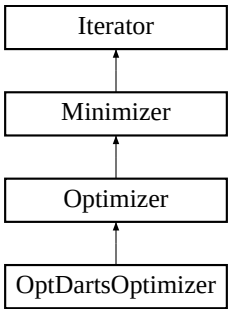
The documentation for this class was generated from the following files:

- NPSOLOptimizer.hpp
- NPSOLOptimizer.cpp

13.113 OptDartsOptimizer Class Reference

Wrapper class for OptDarts Optimizer.

Inheritance diagram for OptDartsOptimizer:



Public Member Functions

- OptDartsOptimizer (ProblemDescDB &problem_db, Model &model)

  Constructor.
- OptDartsOptimizer (Model &model)

  alternate constructor for Iterator instantiations by name
- ~OptDartsOptimizer ()

  Destructor.
- void find_optimum ()

  Calls the OptDarts algorithm.

Private Member Functions

- void load_parameters (Model &model)

  Convenience function for Parameter loading.
- double opt_darts_f ()

  Function evaluation.
- void opt_darts_execute (size_t num_dim, size_t budget, double *xmin, double *xmax, double TOL, size_t problem_index, double fw_MC, double fb_MC)

  Run the OPT-DARTS method.
- void opt_darts_initiate (double *xmin, double *xmax)

  Initialize OPT-DARTS.
- void **opt_darts_reset_convex_hull** ()
- size_t opt_darts_pick_candidate (size_t ifunc)

  Choose the next trial iterate.
- void **retrieve_extended_neighbors** (size_t icandidate)
- void **opt_darts_sample_from_candidate_neighborhood** (size_t icandidate, size_t ifunc)
- void **DIRECT_sample_from_candidate_neighborhood** (size_t icandidate)
- void **opt_darts_add_dart** ()
- void **opt_darts_update_K_h_approximate_Voronoi** (size_t isample)
- void opt_darts_terminate ()

  Release memory and exit cleanly.
- void opt_darts_plot_discs_2d (size_t icandidate)

  Convenience function for plotting iterates.

- void opt_darts_plot_hull_2d (size_t icandidate, size_t ifunc)

  Convenience function for plotting convex hull.
- void **initiate_random_generator** (unsigned long x)
- double **generate_a_random_number** ()
- void **sample_uniformly_from_unit_sphere_surface** (double *dart, size_t num_dim)
- bool **trim_line_using_Hyperplane** (size_t num_dim, double *st, double *end, double *qH, double *nH)

Private Attributes

- double * **_xmin**
- double * **_xmax**
- double * **_dart**
- double * **_st**
- double * **_end**
- double * **_tmp_point**
- double * **_qH**
- double * **_nH**
- double ** **_x**
- double ** **_xc**
- double ** **_f**
- double ** **_K**
- double * **_h**
- double * **_r**
- size_t ** **_neighbors**
- size_t * **_tmp_neighbors**
- size_t * **_ext_neighbors**
- size_t **_num_ext_neighbors**
- bool **_use_opt_darts**
- bool **_estimate_K**
- size_t **_ib**
- size_t **_num_samples**
- size_t **_budget**
- size_t **_num_dim**
- double **_diag**
- size_t **_problem_index**
- double **_fb**
- double **_fw**
- double **_fval**
- size_t **_corner_index**
- size_t **_num_corners**
- size_t * **_corners**
- double **_epsilon**
- double **_fb_MC**
- double **_fw_MC**
- double ** **_xm**
- double ** **_xp**

- double * **_alpha_Deceptive**
- double **Q** [1220]
- int **indx**
- double **cc**
- double **c**
- double **zc**
- double **zx**
- double **zy**
- size_t **qlen**
- bool **use_DIRECT**
- int **numTotalVars**
- int **randomSeed**
- int **maxBlackBoxEvals**
- int **maxIterations**

Additional Inherited Members

13.113.1 Detailed Description

Wrapper class for OptDarts Optimizer.

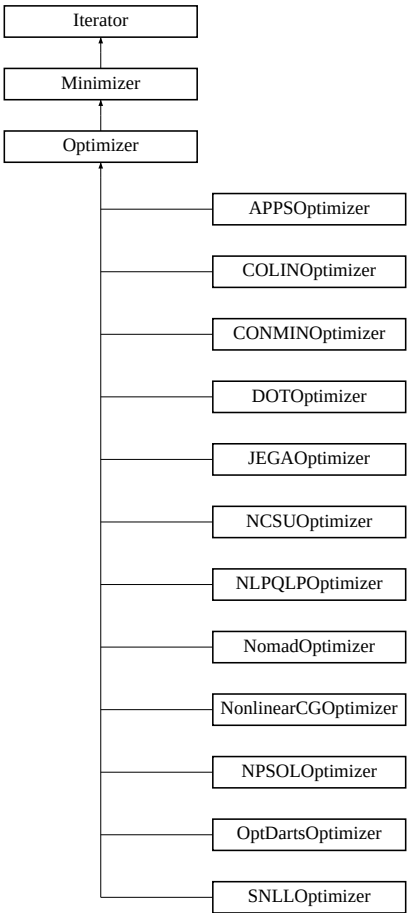
The documentation for this class was generated from the following files:

- OptDartsOptimizer.hpp
- OptDartsOptimizer.cpp

13.114 Optimizer Class Reference

Base class for the optimizer branch of the iterator hierarchy.

Inheritance diagram for Optimizer:



Static Public Member Functions

- static void not_available (const std::string &package_name)

  Static helper function: third-party opt packages which are not available.

Protected Member Functions

- Optimizer ()

  default constructor
- Optimizer (ProblemDescDB &problem_db, Model &model)

  alternate constructor; accepts a model
- Optimizer (unsigned short method_name, Model &model)

  alternate constructor for "on the fly" instantiations
- Optimizer (unsigned short method_name, size_t num_cv, size_t num_div, size_t num_dsv, size_t num_drv, size_t num_lin_ineq, size_t num_lin_eq, size_t num_nln_ineq, size_t num_nln_eq)

  alternate constructor for "on the fly" instantiations
- ~Optimizer ()

  destructor

- void initialize_run ()
- void core_run ()

  core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post
- void post_run (std::ostream &s)
- void finalize_run ()

  utility function to perform common operations following post_run(); deallocation and resetting of instance pointers
- void print_results (std::ostream &s)
- virtual void find_optimum ()=0

  Used within the optimizer branch for computing the optimal solution. Redefines the run virtual function for the optimizer branch.

Protected Attributes

- size_t numObjectiveFns

  number of objective functions (iterator view)
- bool localObjectiveRecast

  flag indicating whether local recasting to a single objective is used
- Optimizer * prevOptInstance

  pointer containing previous value of optimizerInstance

Static Protected Attributes

- static Optimizer * optimizerInstance

  pointer to Optimizer instance used in static member functions

Private Member Functions

- void reduce_model (bool local_nls_recast, bool require_hessians)

  Wrap iteratedModel in a RecastModel that performs (weighted) multi-objective or sum-of-squared residuals transformation.
- void objective_reduction (const Response &full_response, const BoolDeque &sense, const RealVector &full_wts, Response &reduced_response) const

  forward mapping: maps multiple primary response functions to a single weighted objective for single-objective optimizers
- void local_objective_recast_retrieve (const Variables &vars, Response &response) const

  infers MOO/NLS solution from the solution of a single-objective optimizer

Static Private Member Functions

- static void primary_resp_reducer (const Variables &full_vars, const Variables &reduced_vars, const Response &full_response, Response &reduced_response)

  Recast callback to reduce multiple objectives or residuals to a single objective, with gradients and Hessians as needed.

Additional Inherited Members

13.114.1 Detailed Description

Base class for the optimizer branch of the iterator hierarchy.

The Optimizer class provides common data and functionality for DOTOptimizer, CONMINOptimizer, NPSOLOptimizer, SNLLOptimizer, NLPQLPOptimizer, COLINOptimizer, and JEGAOptimizer.

13.114.2 Member Function Documentation

void initialize_run ( ) [protected], [virtual]

Implements portions of initialize_run specific to Optimizers. This function should be invoked (or reimplemented) by any derived implementations of initialize_run() (which would otherwise hide it).

Reimplemented from Minimizer.

Reimplemented in SNLLOptimizer, NLPQLPOptimizer, and DOTOptimizer.

References Minimizer::initialize_run(), Iterator::iteratedModel, Minimizer::minimizerRecasts, Optimizer::optimizerInstance, Optimizer::prevOptInstance, and Model::update_from_subordinate_model().

Referenced by DOTOptimizer::initialize_run(), CONMINOptimizer::initialize_run(), NLPQLPOptimizer::initialize_run(), and SNLLOptimizer::initialize_run().

void core_run ( ) [inline], [protected], [virtual]

core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post

Virtual run function for the iterator class hierarchy. All derived classes need to redefine it.

Reimplemented from Iterator.

References Optimizer::find_optimum().

void post_run (std::ostream & *s*) [protected], [virtual]

Implements portions of post_run specific to Optimizers. This function should be invoked (or reimplemented) by any derived implementations of post_run() (which would otherwise hide it).

Reimplemented from Minimizer.

Reimplemented in SNLLOptimizer.

References Dakota::abort_handler(), Response::active_set_request_vector(), Iterator::bestResponseArray, Iterator::bestVariablesArray, Variables::continuous_variables(), Response::copy(), Minimizer::cvScaleMultipliers, Minimizer::cvScaleOffsets, Minimizer::cvScaleTypes, Minimizer::expData, Response::function_value(), Response::function_values(), Optimizer::local_objective_recast_retrieve(), Optimizer::localObjectiveRecast, Minimizer::modify_s2n(), Minimizer::need_resp_trans_byvars(), Minimizer::numExperiments, Minimizer::numNonlinearConstraints, Minimizer::numUserPrimaryFns, Minimizer::obsDataFlag, Minimizer::post_run(), Minimizer::primaryRespScaleFlag, Minimizer::response_modify_s2n(), ExperimentData::scalar_data(), Minimizer::secondaryRespScaleFlag, Response::update_partial(), and Minimizer::varsScaleFlag.

Referenced by COLINOptimizer::post_run(), and SNLLOptimizer::post_run().

void finalize_run ( ) [inline], [protected], [virtual]

utility function to perform common operations following post_run(); deallocation and resetting of instance pointers

Optional: perform finalization phases of run sequence, like deallocating memory and resetting instance pointers. Commonly used in sub-iterator executions. This is a virtual function; when re-implementing, a derived class must call its nearest parent's finalize_run(), typically *after* performing its own implementation steps.

Reimplemented from Minimizer.

Reimplemented in SNLLOptimizer.

References Minimizer::finalize_run(), Optimizer::optimizerInstance, and Optimizer::prevOptInstance.

Referenced by SNLLOptimizer::finalize_run().

void print_results (std::ostream & *s*) [protected], [virtual]

Redefines default iterator results printing to include optimization results (objective functions and constraints).

Reimplemented from Iterator.

References Dakota::abort_handler(), Minimizer::archive_allocate_best(), Minimizer::archive_best(), Iterator::bestResponseArray, Iterator::bestVariablesArray, Dakota::data_pairs, Model::interface_id(), Iterator::iteratedModel, Dakota::lookup_by_val(), Minimizer::numContinuousVars, Minimizer::numFunctions, Minimizer::numNonlinearConstraints, Minimizer::optimizationFlag, Model::primary_response_fn_weights(), Model::subordinate_model(), and Dakota::write_precision.

void reduce_model (bool *local_nls_recast,* bool *require_hessians*) [private]

Wrap iteratedModel in a RecastModel that performs (weighted) multi-objective or sum-of-squared residuals transformation.

Reduce model for least-squares or multi-objective transformation. Doesn't map variables, or secondary responses. Maps active set for Gauss-Newton. Maps primary responses to single objective so user vs. iterated matters.

References Iterator::activeSet, Model::assign_rep(), Model::current_response(), Minimizer::gnewton_set_recast(), Model::hessian_type(), Iterator::iteratedModel, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, Minimizer::numNonlinearIneqConstraints, Optimizer::numObjectiveFns, Minimizer::numRowsExpData, Minimizer::numUserPrimaryFns, Minimizer::obsDataFlag, Iterator::outputLevel, Optimizer::primary_resp_reducer(), Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), ActiveSet::request_vector(), Response::reshape(), and Minimizer::secondary_resp_copier().

Referenced by Optimizer::Optimizer().

void primary_resp_reducer (const Variables & *full_vars,* const Variables & *reduced_vars,* const Response & *full_response,* Response & *reduced_response*) [static], [private]

Recast callback to reduce multiple objectives or residuals to a single objective, with gradients and Hessians as needed.

Objective function map from multiple primary responses (objective or residuals) to a single objective. Currently supports weighted sum; may later want more general transformations, e.g., goal-oriented

References Iterator::iteratedModel, Optimizer::objective_reduction(), Optimizer::optimizerInstance, Iterator::outputLevel, Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), and Model::subordinate_model().

Referenced by Optimizer::reduce_model().

void objective_reduction (const Response & *full_response,* const BoolDeque & *sense,* const RealVector & *full_wts,* Response & *reduced_response*) const [private]

forward mapping: maps multiple primary response functions to a single weighted objective for single-objective optimizers

This function is responsible for the mapping of multiple objective functions into a single objective for publishing to single-objective optimizers. Used in DOTOptimizer, NPSOLOptimizer, SNLLOptimizer, and SGOPTApplication on every function evaluation. The simple weighting approach (using primaryRespFnWts) is the

only technique supported currently. The weightings are used to scale function values, gradients, and Hessians as needed.

References Response::active_set_request_vector(), Response::function_gradient_view(), Response::function_gradients(), Response::function_hessian_view(), Response::function_hessians(), Response::function_value(), Response::function_values(), Response::num_functions(), Minimizer::numConstraints, Minimizer::objective(), Minimizer::objective_gradient(), Minimizer::objective_hessian(), Iterator::outputLevel, and Dakota::write_precision.

Referenced by Optimizer::primary_resp_reducer().

void local_objective_recast_retrieve (const Variables & *vars*, Response & *response*) const `[private]`

infers MOO/NLS solution from the solution of a single-objective optimizer

Retrieve a MOO/NLS response based on the data returned by a single objective optimizer by performing a data_pairs search. This may get called even for a single user-specified function, since we may be recasting a single NLS residual into a squared objective.

References Response::active_set(), Dakota::data_pairs, Response::function_value(), Model::interface_id(), Iterator::iteratedModel, Dakota::lookup_by_val(), Minimizer::numRowsExpData, Minimizer::numUserPrimaryFns, Minimizer::obsDataFlag, ActiveSet::reshape(), and Response::update().

Referenced by Optimizer::post_run().

The documentation for this class was generated from the following files:

- DakotaOptimizer.hpp
- DakotaOptimizer.cpp

13.115 OutputManager Class Reference

Class to manage redirection of stdout/stderr, keep track of current redir state, and manage rank 0 output. Also manage tabular data output for post-processing with Matlab, Tecplot, etc. and delegate to Graphics for X Windows Graphics.

Public Member Functions

- OutputManager ()

  Default constructor (needed for default environment ctors)
- OutputManager (const ProgramOptions &prog_opts, int dakota_world_rank=0, bool dakota_mpirun_flag=false)

  Standard constructor, taking user-specified program options and optionally taking the rank of this process in Dakota's MPI_Comm.
- ~OutputManager ()

  Destructor that closes streams and other outputs.
- void close_streams ()

  helper to close streams during destructor or abnormal abort
- Graphics & graphics ()

  retrieve the graphics handler object
- void parse (const ProblemDescDB &problem_db)

  Extract environment options from ProblemDescDB.
- void startup_message (const String &start_msg)

  Set the Dakota startup message ("Running on...")

- void push_output_tag (const String &iterator_tag, const ProgramOptions &prog_opts, bool force_cout_redirect, bool force_rst_redirect)

  Update the tag to use on files and rebind any streams as needed.
- void pop_output_tag ()

  (Potentially) remove an output context and rebind streams
- void output_version (std::ostream &os=Cout) const

  Output the current Dakota version.
- void output_startup_message (std::ostream &os=Cout) const

  Output the startup header and time.
- void output_helper (const String &message, std::ostream &os) const

  Output only on Dakota world rank 0 (for version, help, etc.)
- void append_restart (const ParamResponsePair &prp)

  append a parameter/response set to the restart file
- void add_datapoint (const Variables &vars, const String &iface, const Response &response)

  adds data to each window in the 2d graphics and adds a row to the tabular data file based on the results of a model evaluation
- void create_tabular_datastream (const Variables &vars, const Response &resp)

  initialize the tabular datastream on iterator leaders
- void close_tabular ()

  close tabular datastream
- void graphics_counter (int cntr)

  set graphicsCntr equal to cntr
- int graphics_counter () const

  return graphicsCntr
- void tabular_counter_label (const std::string &label)

  set tabularCntrLabel equal to label

Public Attributes

- bool graph2DFlag

  whether user requested 2D graphics plots
- bool tabularDataFlag

  whether user requested tabular data file
- bool resultsOutputFlag

  whether user requested results data output
- String tabularDataFile

  filename for tabulation of graphics data
- String resultsOutputFile

  filename for results data

Private Member Functions

- void read_write_restart (bool restart_requested, bool read_restart_flag, const String &read_restart_filename, size_t stop_restart_eval, const String &write_restart_filename)

  conditionally import evaluations from restart file, then always create or overwrite restart file

Private Attributes

- int worldRank

  output manager handles rank 0 only output when needed
- bool mpirunFlag

  some output is only for MPI runs
- StringArray fileTags

  set of tags for various input/output files (default none)
- bool redirCalled

  temporary variable to prevent recursive tagging initially
- ConsoleRedirector coutRedirector

  set of redirections for Dakota::Cout; stores any tagged filename when there are concurrent Iterators
- ConsoleRedirector cerrRedirector

  set of redirections for Dakota::Cerr; stores any tagged filename when there are concurrent Iterators and error redirection is requested
- std::vector< boost::shared_ptr < RestartWriter > > restartDestinations

  Stack of active restart destinations; end is the last (active) redirection. All remain open until popped or destroyed.
- String startupMessage

  message to print at startup when proceeding to instantiate objects
- Graphics dakotaGraphics

  graphics and tabular data output handler used by meta-iterators, models, and approximations; encapsulated here so destroyed with the OutputManager
- int graphicsCntr

  used for x axis values in 2D graphics and for 1st column in tabular data
- std::ofstream tabularDataFStream

  file stream for tabulation of graphics data within compute_response
- std::string tabularCntrLabel

  label for counter used in first line comment w/i the tabular data file
- short outputLevel

  output level (for debugging only; not passed in)

13.115.1 Detailed Description

Class to manage redirection of stdout/stderr, keep track of current redir state, and manage rank 0 output. Also manage tabular data output for post-processing with Matlab, Tecplot, etc. and delegate to Graphics for X Windows Graphics.

13.115.2 Constructor & Destructor Documentation

OutputManager (const ProgramOptions & *prog_opts*, int *dakota_world_rank* = *0*, bool *dakota_mpirun_flag* = *false*)

Standard constructor, taking user-specified program options and optionally taking the rank of this process in Dakota's MPI_Comm.

Only get minimal information off ProgramOptions as may be updated later by broadcast.

References OutputManager::cerrRedirector, OutputManager::coutRedirector, ProgramOptions::error_file(), OutputManager::mpirunFlag, ProgramOptions::output_file(), ConsoleRedirector::push_back(), Dakota::start_dakota_heartbeat(), ProgramOptions::user_stderr_redirect(), ProgramOptions::user_stdout_redirect(), and OutputManager::worldRank.

13.115.3 Member Function Documentation

void pop_output_tag ( )

(Potentially) remove an output context and rebind streams

For now this assumes the tag is .<int>

References OutputManager::cerrRedirector, OutputManager::coutRedirector, OutputManager::fileTags, OutputManager::outputLevel, ConsoleRedirector::pop_back(), OutputManager::restartDestinations, and OutputManager::worldRank.

Referenced by ParallelLibrary::pop_output_tag().

void add_datapoint (const Variables & *vars*, const String & *iface*, const Response & *response*)

adds data to each window in the 2d graphics and adds a row to the tabular data file based on the results of a model evaluation

Adds data to each 2d plot and each tabular data column (one for each active variable and for each response function). graphicsCntr is used for the x axis in the graphics and the first column in the tabular data.

References Response::active_set_request_vector(), Graphics::add_datapoint(), OutputManager::dakotaGraphics, OutputManager::graphicsCntr, and OutputManager::tabularDataFStream.

Referenced by Model::compute_response(), SurrBasedLocalMinimizer::minimize_surrogates(), Model::synchronize(), and Model::synchronize_nowait().

void create_tabular_datastream (const Variables & *vars*, const Response & *response*)

initialize the tabular datastream on iterator leaders

Opens the tabular data file stream and prints headings, one for each active continuous and discrete variable and one for each response function, using the variable and response function labels. This tabular data is used for post-processing of DAKOTA results in Matlab, Tecplot, etc.

References OutputManager::fileTags, OutputManager::tabularDataFile, and OutputManager::tabularDataFStream.

Referenced by SurrBasedMinimizer::initialize_graphics(), and Iterator::initialize_graphics().

The documentation for this class was generated from the following files:

- OutputManager.hpp
- OutputManager.cpp

13.116 OutputWriter Class Reference

Public Member Functions

- OutputWriter (std::ostream *output_stream)

  ostream constructor; used to construct a writer to existing stream, e.g., std::cout
- OutputWriter (const String &output_filename)

  file redirect constructor; opens an overwriting file stream to given name
- const String & filename () const

the (possibly empty) file name for this stream

- std::ostream * output_stream ()

  a pointer to the stream, either cout/cerr or a file

Protected Attributes

- String outputFilename

  the name of the output file (empty when constructed from pointer)
- std::ofstream outputFS

  file output stream for console text; only open if string non-empty
- std::ostream * outputStream

  pointer to the stream for this writer

13.116.1 Detailed Description

Component to manage a redirected output or error stream

The documentation for this class was generated from the following files:

- OutputManager.hpp
- OutputManager.cpp

13.117 ParallelConfiguration Class Reference

Container class for a set of ParallelLevel list iterators that collectively identify a particular multilevel parallel configuration.

Public Member Functions

- ParallelConfiguration ()

  default constructor
- ParallelConfiguration (const ParallelConfiguration &pl)

  copy constructor
- ~ParallelConfiguration ()

  destructor
- ParallelConfiguration & operator= (const ParallelConfiguration &pl)

  assignment operator
- const ParallelLevel & w_parallel_level () const

  return the ParallelLevel corresponding to miPLIters.front()
- const ParallelLevel & mi_parallel_level (size_t index=_NPOS) const

  return the ParallelLevel corresponding to miPLIters[index]
- const ParallelLevel & ie_parallel_level () const

  return the ParallelLevel corresponding to iePLIter
- const ParallelLevel & ea_parallel_level () const

  return the ParallelLevel corresponding to eaPLIter
- bool w_parallel_level_defined () const

test for definition of world parallel level

- bool mi_parallel_level_defined (size_t index=_NPOS) const

  test for definition of meta-iterator-iterator parallel level
- bool ie_parallel_level_defined () const

  test for definition of iterator-evaluation parallel level
- bool ea_parallel_level_defined () const

  test for definition of evaluation-analysis parallel level
- ParLevLIter w_parallel_level_iterator () const

  return miPLIters.front()
- ParLevLIter mi_parallel_level_iterator (size_t index=_NPOS) const

  return miPLIters[index]
- ParLevLIter ie_parallel_level_iterator () const

  return iePLIter
- ParLevLIter ea_parallel_level_iterator () const

  return eaPLIter
- size_t mi_parallel_level_index (ParLevLIter pl_iter) const

  return the index within miPLIters corresponding to pl_iter
- size_t mi_parallel_level_last_index () const

  return the index of the last entry in miPLIters

Private Member Functions

- void assign (const ParallelConfiguration &pl)

  assign the attributes of the incoming pl to this object

Private Attributes

- short numParallelLevels

  number of parallel levels
- std::vector< ParLevLIter > miPLIters

  list iterator for world level followed by any concurrent iterator partitions (there may be multiple per parallel configuration instance)
- ParLevLIter iePLIter

  list iterator identifying the iterator-evaluation parallelLevel (there can only be one)
- ParLevLIter eaPLIter

  list iterator identifying the evaluation-analysis parallelLevel (there can only be one)
- ParLevLIter endPLIter

  snapshot of the end of ParallelLibrary::parallelLevels; used for detecting when a component of the parallel configuration has been initialized

Friends

- class ParallelLibrary

  the ParallelLibrary class has special access priveleges in order to streamline implementation

13.117.1 Detailed Description

Container class for a set of ParallelLevel list iterators that collectively identify a particular multilevel parallel configuration.

Rather than containing the multilevel parallel configuration directly, ParallelConfiguration instead provides a set of list iterators which point into a combined list of ParallelLevels. This approach allows different configurations to reuse ParallelLevels without copying them. A list of ParallelConfigurations is contained in ParallelLibrary (ParallelLibrary::parallelConfigurations).

13.117.2 Member Function Documentation

const ParallelLevel & mi_parallel_level (size_t *index* = _NPOS) const `[inline]`

return the ParallelLevel corresponding to miPLIters[index]

If a meaningful index is not provided, return the last mi parallel level. This is useful within the Model context, for which we need the lowest level partition after any meta-iterator recursions.

References Dakota::_NPOS, and ParallelConfiguration::miPLIters.

Referenced by ApplicationInterface::set_evaluation_communicators().

ParLevLIter mi_parallel_level_iterator (size_t *index* = _NPOS) const `[inline]`

return miPLIters[index]

If a meaningful index is not provided, return the last mi parallel level. This is useful within the Model context, for which we need the lowest level partition after any meta-iterator recursions.

References Dakota::_NPOS, and ParallelConfiguration::miPLIters.

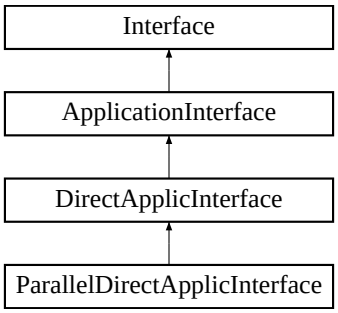
The documentation for this class was generated from the following file:

- ParallelLibrary.hpp

13.118 ParallelDirectApplicInterface Class Reference

Sample derived interface class for testing parallel simulator plug-ins using assign_rep().

Inheritance diagram for ParallelDirectApplicInterface:



Public Member Functions

- ParallelDirectApplicInterface (const Dakota::ProblemDescDB &problem_db, const MPI_Comm &analysis_comm)

  constructor

- ~ParallelDirectApplicInterface ()

  destructor

Protected Member Functions

- int derived_map_ac (const Dakota::String &ac_name)

  execute an analysis code portion of a direct evaluation invocation
- void derived_map_asynch (const Dakota::ParamResponsePair &pair)

  no-op hides base error; job batching occurs within wait_local_evaluations()
- void wait_local_evaluations (Dakota::PRPQueue &prp_queue)

  evaluate the batch of jobs contained in prp_queue
- void test_local_evaluations (Dakota::PRPQueue &prp_queue)

  invokes wait_local_evaluations() (no special nowait support)
- void set_communicators_checks (int max_eval_concurrency)

  no-op hides default run-time error checks at DirectApplicInterface level

Private Member Functions

- int text_book (const Dakota::RealVector &c_vars, const Dakota::ShortArray &asv, Dakota::RealVector &fn_vals, Dakota::RealMatrix &fn_grads, Dakota::RealSymMatrixArray &fn_hessians)

  demo evaluator function for parallel plug-ins

Additional Inherited Members

13.118.1 Detailed Description

Sample derived interface class for testing parallel simulator plug-ins using assign_rep().

The plug-in ParallelDirectApplicInterface resides in namespace SIM and uses a copy of textbook() to perform parallel parameter to response mappings. It is used to demonstrate plugging in a parallel direct analysis driver into Dakota in library mode. Test input files can then use an analysis_driver of "plugin_textbook".

13.118.2 Member Function Documentation

void test_local_evaluations (Dakota::PRPQueue & *prp_queue*) `[inline], [protected]`

invokes wait_local_evaluations() (no special nowait support)

For use by ApplicationInterface::serve_evaluations_asynch(), which can provide a batch processing capability within message passing schedulers (called using chain ApplicationInterface::serve_evaluations() from Model::serve() from IteratorScheduler::run_iterator()).

References ParallelDirectApplicInterface::wait_local_evaluations().

The documentation for this class was generated from the following files:

- PluginParallelDirectApplicInterface.hpp
- PluginParallelDirectApplicInterface.cpp

13.119 ParallelLevel Class Reference

Container class for the data associated with a single level of communicator partitioning.

Public Member Functions

- ParallelLevel ()

  default constructor
- ParallelLevel (const ParallelLevel &pl)

  copy constructor
- ~ParallelLevel ()

  destructor
- ParallelLevel & operator= (const ParallelLevel &pl)

  assignment operator
- bool dedicated_master () const

  return dedicatedMasterFlag
- bool communicator_split () const

  return commSplitFlag
- bool server_master () const

  return serverMasterFlag
- bool message_pass () const

  return messagePass
- bool idle_partition () const

  return idlePartition
- int num_servers () const

  return numServers
- int processors_per_server () const

  return procsPerServer
- int processor_remainder () const

  return procRemainder
- const MPI_Comm & server_intra_communicator () const

  return serverIntraComm
- int server_communicator_rank () const

  return serverCommRank
- int server_communicator_size () const

  return serverCommSize
- const MPI_Comm & hub_server_intra_communicator () const

  return hubServerIntraComm
- int hub_server_communicator_rank () const

  return hubServerCommRank
- int hub_server_communicator_size () const

  return hubServerCommSize
- const MPI_Comm & hub_server_inter_communicator () const

  return hubServerInterComm
- MPI_Comm * hub_server_inter_communicators () const

  return hubServerInterComms
- int server_id () const

return serverId

- void read (MPIUnpackBuffer &s)

  read a ParallelLevel object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a ParallelLevel object to a packed MPI buffer

Private Member Functions

- void assign (const ParallelLevel &pl)

  assign the attributes of the incoming pl to this object

Private Attributes

- bool dedicatedMasterFlag

  signals dedicated master partitioning
- bool commSplitFlag

  signals a communicator split was used
- bool serverMasterFlag

  identifies master server processors
- bool messagePass

  flag for message passing at this level
- bool idlePartition

  `identifies presence of an idle processor`
  < partition at this level
- int numServers

  number of servers
- int procsPerServer

  processors per server
- int procRemainder

  proc remainder after equal distribution
- MPI_Comm serverIntraComm

  intracomm. for each server partition
- int serverCommRank

  rank in serverIntraComm
- int serverCommSize

  size of serverIntraComm
- MPI_Comm hubServerIntraComm

  `intracomm for all serverCommRank==0`
  < w/i next higher level serverIntraComm
- int hubServerCommRank

  rank in hubServerIntraComm
- int hubServerCommSize

  size of hubServerIntraComm
- MPI_Comm hubServerInterComm

`intercomm. between a server & the hub`
< (on server partitions only)

- MPI_Comm * hubServerInterComms

  intercomm. array on hub processor

- int serverId

  server identifier

Friends

- class ParallelLibrary

  the ParallelLibrary class has special access priveleges in order to streamline implementation

13.119.1 Detailed Description

Container class for the data associated with a single level of communicator partitioning.

A list of these levels is contained in ParallelLibrary (ParallelLibrary::parallelLevels), which defines all of the parallelism levels across one or more multilevel parallelism configurations.

The documentation for this class was generated from the following file:

- ParallelLibrary.hpp

13.120 ParallelLibrary Class Reference

Class for partitioning multiple levels of parallelism and managing message passing within these levels.

Public Member Functions

- ParallelLibrary ()

  default constructor (used for dummy_lib)

- ParallelLibrary (const MPIManager &mpi_mgr, ProgramOptions &prog_opts, OutputManager &output_mgr)

  stand-alone and default library mode constructor; don't require options

- ~ParallelLibrary ()

  destructor

- const ParallelLevel & init_iterator_communicators (int iterator_servers, int procs_per_iterator, int min_procs_per_iterator, int max_procs_per_iterator, int max_iterator_concurrency, short default_config, short iterator_scheduling, bool peer_dynamic_avail)

  split MPI_COMM_WORLD into iterator communicators

- const ParallelLevel & init_evaluation_communicators (int evaluation_servers, int procs_per_evaluation, int min_procs_per_eval, int max_procs_per_eval, int max_evaluation_concurrency, int asynch_local_evaluation_concurrency, short default_config, short evaluation_scheduling, bool peer_dynamic_avail)

  split an iterator communicator into evaluation communicators

- const ParallelLevel & init_analysis_communicators (int analysis_servers, int procs_per_analysis, int min_procs_per_analysis, int max_procs_per_analysis, int max_analysis_concurrency, int asynch_local_analysis_concurrency, short default_config, short analysis_scheduling, bool peer_dynamic_avail)

  split an evaluation communicator into analysis communicators

- void free_iterator_communicators (ParLevLIter mi_pl_iter)

deallocate iterator communicators

- void free_evaluation_communicators ()

  deallocate evaluation communicators
- void free_analysis_communicators ()

  deallocate analysis communicators
- void print_configuration ()

  print the parallel level settings for a particular parallel configuration
- void push_output_tag (const ParallelLevel &pl)

  conditionally append an iterator server id tag to the hierarchical output tag, manage restart, and rebind cout/cerr
- void pop_output_tag (const ParallelLevel &pl)

  pop the last output tag and rebind streams as needed; pl isn't yet used, but may be in the future when we generalize to arbitrary output context switching
- void write_restart (const ParamResponsePair &prp)

  write a parameter/response set to the restart file
- ProgramOptions & program_options ()

  return programOptions reference
- OutputManager & output_manager ()

  return outputManager reference
- void terminate_modelcenter ()

  terminate ModelCenter if running
- void abort_helper (int code)

  finalize MPI with correct communicator for abort
- bool command_line_check () const

  return checkFlag
- bool command_line_pre_run () const

  return preRunFlag
- bool command_line_run () const

  return runFlag
- bool command_line_post_run () const

  return postRunFlag
- bool command_line_user_modes () const

  return userModesFlag
- const String & command_line_pre_run_input () const

  preRunInput filename
- const String & command_line_pre_run_output () const

  preRunOutput filename
- const String & command_line_run_input () const

  runInput filename
- const String & command_line_run_output () const

  runOutput filename
- const String & command_line_post_run_input () const

  postRunInput filename
- const String & command_line_post_run_output () const

postRunOutput fname

- void send (MPIPackBuffer &send_buff, int dest, int tag, const ParallelLevel &parent_pl, const ParallelLevel &child_pl)

  blocking buffer send at the current communication level
- void send (int &send_int, int dest, int tag, const ParallelLevel &parent_pl, const ParallelLevel &child_pl)

  blocking integer send at the current communication level
- void isend (MPIPackBuffer &send_buff, int dest, int tag, MPI_Request &send_req, const ParallelLevel &parent_pl, const ParallelLevel &child_pl)

  nonblocking buffer send at the current communication level
- void isend (int &send_int, int dest, int tag, MPI_Request &send_req, const ParallelLevel &parent_pl, const ParallelLevel &child_pl)

  nonblocking integer send at the current communication level
- void recv (MPIUnpackBuffer &recv_buff, int source, int tag, MPI_Status &status, const ParallelLevel &parent_pl, const ParallelLevel &child_pl)

  blocking buffer receive at the current communication level
- void recv (int &recv_int, int source, int tag, MPI_Status &status, const ParallelLevel &parent_pl, const ParallelLevel &child_pl)

  blocking integer receive at the current communication level
- void irecv (MPIUnpackBuffer &recv_buff, int source, int tag, MPI_Request &recv_req, const ParallelLevel &parent_pl, const ParallelLevel &child_pl)

  nonblocking buffer receive at the current communication level
- void irecv (int &recv_int, int source, int tag, MPI_Request &recv_req, const ParallelLevel &parent_pl, const ParallelLevel &child_pl)

  nonblocking integer receive at the current communication level
- void check_mi_index (size_t &index) const

  process _NPOS default and perform error checks
- void send_mi (int &send_int, int dest, int tag, size_t index=_NPOS)

  blocking send at the metaiterator-iterator communication level
- void isend_mi (int &send_int, int dest, int tag, MPI_Request &send_req, size_t index=_NPOS)

  nonblocking send at the metaiterator-iterator communication level
- void recv_mi (int &recv_int, int source, int tag, MPI_Status &status, size_t index=_NPOS)

  blocking receive at the metaiterator-iterator communication level
- void irecv_mi (int &recv_int, int source, int tag, MPI_Request &recv_req, size_t index=_NPOS)

  nonblocking receive at the metaiterator-iterator communication level
- void send_mi (MPIPackBuffer &send_buff, int dest, int tag, size_t index=_NPOS)

  blocking send at the metaiterator-iterator communication level
- void isend_mi (MPIPackBuffer &send_buff, int dest, int tag, MPI_Request &send_req, size_t index=_NPOS)

  nonblocking send at the metaiterator-iterator communication level
- void recv_mi (MPIUnpackBuffer &recv_buff, int source, int tag, MPI_Status &status, size_t index=_NPOS)

  blocking receive at the metaiterator-iterator communication level
- void irecv_mi (MPIUnpackBuffer &recv_buff, int source, int tag, MPI_Request &recv_req, size_t index=_NPOS)

  nonblocking receive at the metaiterator-iterator communication level
- void send_ie (int &send_int, int dest, int tag)

blocking send at the iterator-evaluation communication level

- void isend_ie (int &send_int, int dest, int tag, MPI_Request &send_req)

  nonblocking send at the iterator-evaluation communication level
- void recv_ie (int &recv_int, int source, int tag, MPI_Status &status)

  blocking receive at the iterator-evaluation communication level
- void irecv_ie (int &recv_int, int source, int tag, MPI_Request &recv_req)

  nonblocking receive at the iterator-evaluation communication level
- void send_ie (MPIPackBuffer &send_buff, int dest, int tag)

  blocking send at the iterator-evaluation communication level
- void isend_ie (MPIPackBuffer &send_buff, int dest, int tag, MPI_Request &send_req)

  nonblocking send at the iterator-evaluation communication level
- void recv_ie (MPIUnpackBuffer &recv_buff, int source, int tag, MPI_Status &status)

  blocking receive at the iterator-evaluation communication level
- void irecv_ie (MPIUnpackBuffer &recv_buff, int source, int tag, MPI_Request &recv_req)

  nonblocking receive at the iterator-evaluation communication level
- void send_ea (int &send_int, int dest, int tag)

  blocking send at the evaluation-analysis communication level
- void isend_ea (int &send_int, int dest, int tag, MPI_Request &send_req)

  nonblocking send at the evaluation-analysis communication level
- void recv_ea (int &recv_int, int source, int tag, MPI_Status &status)

  blocking receive at the evaluation-analysis communication level
- void irecv_ea (int &recv_int, int source, int tag, MPI_Request &recv_req)

  nonblocking receive at the evaluation-analysis communication level
- void bcast (int &data, const ParallelLevel &pl)

  broadcast an integer across the serverIntraComm of a ParallelLevel
- void bcast (short &data, const ParallelLevel &pl)

  broadcast an integer across the serverIntraComm of a ParallelLevel
- void bcast_hs (int &data, const ParallelLevel &pl)

  broadcast an integer across the hubServerIntraComm of a ParallelLevel
- void bcast_hs (MPIPackBuffer &send_buff, const ParallelLevel &pl)

  broadcast a MPIPackBuffer across the hubServerIntraComm of a ParallelLevel
- void bcast_hs (MPIUnpackBuffer &recv_buff, const ParallelLevel &pl)

  broadcast a MPIUnpackBuffer across the hubServerIntraComm of a ParallelLevel
- void bcast_w (int &data)

  broadcast an integer across MPI_COMM_WORLD
- void bcast_i (int &data, size_t index=_NPOS)

  broadcast an integer across an iterator communicator
- void bcast_i (short &data, size_t index=_NPOS)

  broadcast a short integer across an iterator communicator
- void bcast_e (int &data)

  broadcast an integer across an evaluation communicator
- void bcast_a (int &data)

  broadcast an integer across an analysis communicator

- void bcast_mi (int &data, size_t index=_NPOS)

  broadcast an integer across a metaiterator-iterator intra communicator
- void bcast_w (MPIPackBuffer &send_buff)

  broadcast a packed buffer across MPI_COMM_WORLD
- void bcast_i (MPIPackBuffer &send_buff, size_t index=_NPOS)

  broadcast a packed buffer across an iterator communicator
- void bcast_e (MPIPackBuffer &send_buff)

  broadcast a packed buffer across an evaluation communicator
- void bcast_a (MPIPackBuffer &send_buff)

  broadcast a packed buffer across an analysis communicator
- void bcast_mi (MPIPackBuffer &send_buff, size_t index=_NPOS)

  broadcast a packed buffer across a metaiterator-iterator intra communicator
- void bcast_w (MPIUnpackBuffer &recv_buff)

  matching receive for packed buffer broadcast across MPI_COMM_WORLD
- void bcast_i (MPIUnpackBuffer &recv_buff, size_t index=_NPOS)

  matching receive for packed buffer bcast across an iterator communicator
- void bcast_e (MPIUnpackBuffer &recv_buff)

  matching receive for packed buffer bcast across an evaluation communicator
- void bcast_a (MPIUnpackBuffer &recv_buff)

  matching receive for packed buffer bcast across an analysis communicator
- void bcast_mi (MPIUnpackBuffer &recv_buff, size_t index=_NPOS)

  matching recv for packed buffer bcast across a metaiterator-iterator intra comm
- void barrier_w ()

  enforce MPI_Barrier on MPI_COMM_WORLD
- void barrier_i (size_t index=_NPOS)

  enforce MPI_Barrier on an iterator communicator
- void barrier_e ()

  enforce MPI_Barrier on an evaluation communicator
- void barrier_a ()

  enforce MPI_Barrier on an analysis communicator
- void reduce_sum_ea (double *local_vals, double *sum_vals, int num_vals)

  compute a sum over an eval-analysis intra-communicator using MPI_Reduce
- void reduce_sum_a (double *local_vals, double *sum_vals, int num_vals)

  compute a sum over an analysis communicator using MPI_Reduce
- void test (MPI_Request &request, int &test_flag, MPI_Status &status)

  test a nonblocking send/receive request for completion
- void wait (MPI_Request &request, MPI_Status &status)

  wait for a nonblocking send/receive request to complete
- void waitall (int num_recvs, MPI_Request *&recv_reqs)

  wait for all messages from a series of nonblocking receives
- void waitsome (int num_sends, MPI_Request *&recv_requests, int &num_recvs, int *&index_array, MPI_-Status *&status_array)

  wait for at least one message from a series of nonblocking receives but complete all that are available

- void free (MPI_Request &request)

  free an MPI_Request
- int world_size () const

  return MPIManager::worldSize
- int world_rank () const

  return MPIManager::worldRank
- bool mpirun_flag () const

  return MPIManager::mpirunFlag
- bool is_null () const

  return dummyFlag
- Real parallel_time () const

  returns current MPI wall clock time
- void parallel_configuration_iterator (ParConfigLIter pc_iter)

  set the current ParallelConfiguration node
- ParConfigLIter parallel_configuration_iterator () const

  return the current ParallelConfiguration node
- const ParallelConfiguration & parallel_configuration () const

  return the current ParallelConfiguration instance
- size_t num_parallel_configurations () const

  returns the number of entries in parallelConfigurations
- bool parallel_configuration_is_complete ()

  identifies if the current ParallelConfiguration has been fully populated
- void increment_parallel_configuration (ParLevLIter mi_pl_iter)

  add a new node to parallelConfigurations and increment currPCIter; limit miPLIters within new configuration to mi_pl_iter level
- void increment_parallel_configuration ()

  add a new node to parallelConfigurations and increment currPCIter; copy all of miPLIters into new configuration
- bool w_parallel_level_defined () const

  test current parallel configuration for definition of world parallel level
- bool mi_parallel_level_defined (size_t index=_NPOS) const

  test current parallel configuration for definition of meta-iterator-iterator parallel level
- bool ie_parallel_level_defined () const

  test current parallel configuration for definition of iterator-evaluation parallel level
- bool ea_parallel_level_defined () const

  test current parallel configuration for definition of evaluation-analysis parallel level
- ParLevLIter w_parallel_level_iterator ()

  for this level, access through ParallelConfiguration is not necessary
- size_t parallel_level_index (ParLevLIter pl_iter)

  return the index within parallelLevels corresponding to pl_iter
- std::vector< MPI_Comm > analysis_intra_communicators ()

  return the set of analysis intra communicators for all parallel configurations (used for setting up direct simulation interfaces prior to execution time).

Private Member Functions

- void init_mpi_comm ()

  convenience function for initializing DAKOTA's top-level MPI communicators, based on dakotaMPIComm
- void initialize_timers ()

  initialize DAKOTA and UTILIB timers
- void output_timers ()

  conditionally output timers in destructor
- void init_communicators (const ParallelLevel &parent_pl, int num_servers, int procs_per_server, int min_procs_per_server, int max_procs_per_server, int max_concurrency, int asynch_local_concurrency, short default_config, short scheduling_override, bool peer_dynamic_avail)

  split a parent communicator into child server communicators
- void free_communicators (ParallelLevel &pl)

  deallocate intra/inter communicators for a particular ParallelLevel
- void split_communicator_dedicated_master (const ParallelLevel &parent_pl, ParallelLevel &child_pl)

  split a parent communicator into a dedicated master processor and num_servers child communicators
- void split_communicator_peer_partition (const ParallelLevel &parent_pl, ParallelLevel &child_pl)

  split a parent communicator into num_servers peer child communicators (no dedicated master processor)
- void resolve_inputs (ParallelLevel &child_pl, int avail_procs, int min_procs_per_server, int max_procs_per_server, int max_concurrency, int capacity_multiplier, short default_config, short scheduling_override, bool peer_dynamic_avail, bool print_rank)

  resolve user inputs into a sensible partitioning scheme
- void bcast (int &data, const MPI_Comm &comm)

  broadcast an integer across a communicator
- void bcast (short &data, const MPI_Comm &comm)

  broadcast a short integer across a communicator
- void bcast (MPIPackBuffer &send_buff, const MPI_Comm &comm)

  send a packed buffer across a communicator using a broadcast
- void bcast (MPIUnpackBuffer &recv_buff, const MPI_Comm &comm)

  matching receive for a packed buffer broadcast
- void barrier (const MPI_Comm &comm)

  enforce MPI_Barrier on comm
- void reduce_sum (double *local_vals, double *sum_vals, int num_vals, const MPI_Comm &comm)

  compute a sum over comm using MPI_Reduce
- void check_error (const String &err_source, int err_code)

  check the MPI return code and abort if error
- void inherit_as_server_comm (const ParallelLevel &parent_pl, ParallelLevel &child_pl)

  convenience function for updating child serverIntraComm from parent serverIntraComm
- void inherit_as_hub_server_comm (const ParallelLevel &parent_pl, ParallelLevel &child_pl)

  convenience function for updating child hubServerIntraComm from parent serverIntraComm

Private Attributes

- const MPIManager & mpiManager

  reference to the MPI manager with Dakota's MPI options
- ProgramOptions & programOptions

  programOptions is non-const due to updates from broadcast
- OutputManager & outputManager

  Non-const output handler to help with file redirection.
- bool dummyFlag

  prevents multiple MPI_Finalize calls due to dummy_lib
- bool outputTimings

  timing info only beyond help/version/check
- Real startCPUTime

  start reference for UTILIB CPU timer
- Real startWCTime

  start reference for UTILIB wall clock timer
- Real startMPITime

  start reference for MPI wall clock timer
- long startClock

  `start reference for local clock() timer measuring`
  < parent+child CPU
- std::list< ParallelLevel > parallelLevels

  the complete set of parallelism levels for managing multilevel parallelism among one or more configurations
- std::list< ParallelConfiguration > parallelConfigurations

  the set of parallel configurations which manage list iterators for indexing into parallelLevels
- ParConfigLIter currPCIter

  list iterator identifying the current node in parallelConfigurations

13.120.1 Detailed Description

Class for partitioning multiple levels of parallelism and managing message passing within these levels.

The ParallelLibrary class encapsulates all of the details of performing message passing within multiple levels of parallelism. It provides functions for partitioning of levels according to user configuration input and functions for passing messages within and across MPI communicators for each of the parallelism levels. If support for other message-passing libraries beyond MPI becomes needed (PVM, ...), then ParallelLibrary would be promoted to a base class with virtual functions to encapsulate the library-specific syntax.

13.120.2 Constructor & Destructor Documentation

ParallelLibrary ( )

default constructor (used for dummy_lib)

This constructor is used for creation of the global dummy_lib object, which is used to satisfy initialization requirements when the real ParallelLibrary object is not available.

ParallelLibrary (const MPIManager & *mpi_mgr,* ProgramOptions & *prog_opts,* OutputManager & *output_mgr*)

stand-alone and default library mode constructor; don't require options

library mode constructor accepting communicator

TODO: Update comment.

Same constructor is used for executable and library environments and sequencing of object construction is ordered, so no need to separately get updates off command line (programOptions)

References ParallelLibrary::init_mpi_comm(), and ParallelLibrary::initialize_timers().

13.120.3 Member Function Documentation

void push_output_tag (const ParallelLevel & *pl*)

conditionally append an iterator server id tag to the hierarchical output tag, manage restart, and rebind cout/cerr

If the user has specified the use of files for DAKOTA standard output and/or standard error, then bind these filenames to the Cout/Cerr macros. In addition, if concurrent iterators are to be used, create and tag multiple output streams in order to prevent jumbled output. Manage restart file(s) by processing any incoming evaluations from an old restart file and by setting up the binary output stream for new evaluations. Only master iterator processor(s) read & write restart information. This function must follow init_iterator_communicators so that restart can be managed properly for concurrent iterator strategies. In the case of concurrent iterators, each iterator has its own restart file tagged with iterator number.

References ParallelLibrary::bcast(), ParallelLevel::dedicatedMasterFlag, OutputManager::graph2DFlag, ParallelLevel::hubServerCommRank, ParallelLevel::hubServerCommSize, ParallelLevel::hubServerIntraComm, ParallelLevel::numServers, ParallelLibrary::outputManager, ParallelLibrary::programOptions, OutputManager::push_output_tag(), OutputManager::resultsOutputFile, OutputManager::resultsOutputFlag, ParallelLevel::serverCommRank, ParallelLevel::serverId, MPIPackBuffer::size(), OutputManager::tabularDataFile, and OutputManager::tabularDataFlag.

Referenced by Environment::construct(), and IteratorScheduler::init_iterator_parallelism().

void terminate_modelcenter ( )

terminate ModelCenter if running

Close streams associated with manage_outputs and manage_restart and terminate any additional services that may be active.

References Dakota::abort_handler(), Dakota::dc_ptr_int, and Dakota::mc_ptr_int.

Referenced by ParallelLibrary::~ParallelLibrary().

void increment_parallel_configuration (ParLevLIter *mi_pl_iter*) `[inline]`

add a new node to parallelConfigurations and increment currPCIter; limit miPLIters within new configuration to mi_pl_iter level

Called from the ParallelLibrary ctor and from Model::init_communicators(). An increment is performed for each Model initialization except the first (which inherits the world level from the first partial configuration).

References ParallelLibrary::currPCIter, ParallelConfiguration::eaPLIter, ParallelConfiguration::endPLIter, ParallelConfiguration::iePLIter, ParallelConfiguration::miPLIters, ParallelConfiguration::numParallelLevels, ParallelLibrary::parallelConfigurations, and ParallelLibrary::parallelLevels.

Referenced by Iterator::init_communicators(), and Model::init_communicators().

void init_mpi_comm ( ) [private]

convenience function for initializing DAKOTA's top-level MPI communicators, based on dakotaMPIComm

shared function for initializing based on passed MPI_Comm

References Dakota::abort_handler(), MPIManager::dakota_mpi_comm(), ParallelLibrary::increment_parallel_configuration(), ParallelLibrary::mpiManager, MPIManager::mpirun_flag(), ParallelLevel::numServers, ParallelLibrary::outputManager, ParallelLibrary::parallelLevels, ParallelLevel::procsPerServer, ParallelLevel::serverCommRank, ParallelLevel::serverCommSize, ParallelLevel::serverId, ParallelLevel::serverIntraComm, ParallelLevel::serverMasterFlag, ParallelLibrary::startMPITime, OutputManager::startup_message(), MPIManager::world_rank(), and MPIManager::world_size().

Referenced by ParallelLibrary::ParallelLibrary().

void init_communicators (const ParallelLevel & *parent_pl*, int *num_servers*, int *procs_per_server*, int *min_procs_per_server*, int *max_procs_per_server*, int *max_concurrency*, int *asynch_local_concurrency*, short *default_config*, short *scheduling_override*, bool *peer_dynamic_avail*) [private]

split a parent communicator into child server communicators

Split parent communicator into concurrent child server partitions as specified by the passed parameters. This constructs new child intra-communicators and parent-child inter-communicators. This fn is called from MetaIterators and NestedModel for the concurrent iterator level and from ApplicationInterface::init_communicators() for the concurrent evaluation and concurrent analysis levels.

References ParallelLibrary::currPCIter, ParallelLevel::dedicatedMasterFlag, ParallelLevel::messagePass, ParallelLevel::numServers, ParallelLibrary::parallelLevels, ParallelLevel::procsPerServer, ParallelLibrary::resolve_inputs(), ParallelLevel::serverCommRank, ParallelLevel::serverCommSize, ParallelLibrary::split_communicator_dedicated_master(), and ParallelLibrary::split_communicator_peer_partition().

Referenced by ParallelLibrary::init_analysis_communicators(), ParallelLibrary::init_evaluation_communicators(), and ParallelLibrary::init_iterator_communicators().

void resolve_inputs (ParallelLevel & *child_pl*, int *avail_procs*, int *min_procs_per_server*, int *max_procs_per_server*, int *max_concurrency*, int *capacity_multiplier*, short *default_config*, short *scheduling_override*, bool *peer_dynamic_avail*, bool *print_rank*) [private]

resolve user inputs into a sensible partitioning scheme

This function is responsible for the "auto-configure" intelligence of DAKOTA. It resolves a variety of inputs and overrides into a sensible partitioning configuration for a particular parallelism level. It also handles the general case in which a user's specification request does not divide out evenly with the number of available processors for the level. If num_servers & procs_per_server are both nondefault, then the former takes precedence.

References Dakota::abort_handler(), ParallelLevel::dedicatedMasterFlag, ParallelLevel::numServers, ParallelLevel::procRemainder, and ParallelLevel::procsPerServer.

Referenced by ParallelLibrary::init_communicators().

The documentation for this class was generated from the following files:

- ParallelLibrary.hpp
- ParallelLibrary.cpp

13.121 ParamResponsePair Class Reference

Container class for a variables object, a response object, and an evaluation id.

Public Member Functions

- ParamResponsePair ()

  default constructor
- ParamResponsePair (const Variables &vars, const String &interface_id, const Response &response, bool deep_copy=false)

  alternate constructor for temporaries
- ParamResponsePair (const Variables &vars, const String &interface_id, const Response &response, const int eval_id, bool deep_copy=true)

  standard constructor for history uses
- ParamResponsePair (const ParamResponsePair &pair)

  copy constructor
- ~ParamResponsePair ()

  destructor
- ParamResponsePair & operator= (const ParamResponsePair &pair)

  assignment operator
- void read (std::istream &s)

  read a ParamResponsePair object from an std::istream
- void write (std::ostream &s) const

  write a ParamResponsePair object to an std::ostream
- void read_annotated (std::istream &s)

  read a ParamResponsePair object in annotated format from an std::istream
- void write_annotated (std::ostream &s) const

  write a ParamResponsePair object in annotated format to an std::ostream
- void write_tabular (std::ostream &s) const

  write a ParamResponsePair object in tabular format (all variables active/inactive) to an std::ostream
- void write_tabular_labels (std::ostream &s) const

  write PRP labels in tabular format to an std::ostream
- void read (MPIUnpackBuffer &s)

  read a ParamResponsePair object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a ParamResponsePair object to a packed MPI buffer
- int eval_id () const

  return the evaluation identifier
- void eval_id (int id)

  set the evaluation identifier
- const String & interface_id () const

  return the interface identifier from the response object
- const IntStringPair & eval_interface_ids () const

  return the aggregate eval/interface identifier from the response object
- const Variables & prp_parameters () const

  return the parameters object
- void prp_parameters (const Variables &vars)

  set the parameters object

- const Response & prp_response () const

  return the response object
- void prp_response (const Response &response)

  set the response object
- const ActiveSet & active_set () const

  return the active set object from the response object
- void active_set (const ActiveSet &set)

  set the active set object within the response object

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  serialize the PRP: write and read are symmetric for this class

Private Attributes

- Variables prPairParameters

  the set of parameters for the function evaluation
- Response prPairResponse

  the response set for the function evaluation
- IntStringPair evalInterfaceIds

  the evalInterfaceIds aggregate

Friends

- class boost::serialization::access

  allow boost access to serialize this class
- bool operator== (const ParamResponsePair &pair1, const ParamResponsePair &pair2)

  equality operator
- bool operator!= (const ParamResponsePair &pair1, const ParamResponsePair &pair2)

  inequality operator

13.121.1 Detailed Description

Container class for a variables object, a response object, and an evaluation id.

ParamResponsePair provides a container class for association of the input for a particular function evaluation (a variables object) with the output from this function evaluation (a response object), along with an evaluation identifier. This container defines the basic unit used in the data_pairs cache, in restart file operations, and in a variety of scheduling algorithm queues. With the advent of STL, replacement of arrays of this class with map<> and pair<> template constructs may be possible (using map<pair<int,String>, pair<Variables,Response> >, for example), assuming that deep copies, I/O, alternate constructors, etc., can be adequately addressed. Boost tuple<> may also be a candidate.

13.121.2 Constructor & Destructor Documentation

ParamResponsePair (const Variables & *vars*, const String & *interface_id*, const Response & *response*, bool *deep_copy* = `false`) `[inline]`

alternate constructor for temporaries

Uses of this constructor often employ the standard Variables and Response copy constructors to share representations since this constructor is commonly used for search_pairs (which are local instantiations that go out of scope prior to any changes to values; i.e., they are not used for history).

ParamResponsePair (const Variables & *vars*, const String & *interface_id*, const Response & *response*, const int *eval_id*, bool *deep_copy* = `true`) `[inline]`

standard constructor for history uses

Uses of this constructor often do not share representations since deep copies are used when history mechanisms (e.g., data_pairs and beforeSynchCorePRPQueue) are involved.

13.121.3 Member Function Documentation

void read (MPIUnpackBuffer & *s*) `[inline]`

read a ParamResponsePair object from a packed MPI buffer

interfaceId is omitted since master processor retains interface ids and communicates asv and response data only with slaves.

References ParamResponsePair::evalInterfaceIds, ParamResponsePair::prPairParameters, and ParamResponsePair::prPairResponse.

void write (MPIPackBuffer & *s*) const `[inline]`

write a ParamResponsePair object to a packed MPI buffer

interfaceId is omitted since master processor retains interface ids and communicates asv and response data only with slaves.

References ParamResponsePair::evalInterfaceIds, ParamResponsePair::prPairParameters, and ParamResponsePair::prPairResponse.

13.121.4 Member Data Documentation

IntStringPair evalInterfaceIds `[private]`

the evalInterfaceIds aggregate

the function evaluation identifier (assigned from Interface::evalIdCntr) is paired with the interface used to generate the response object. Used in PRPCache id_vars_set_compare to prevent duplicate detection on results from different interfaces. evalInterfaceIds belongs here rather than in Response since some Response objects involve consolidation of several fn evals (e.g., Model::synchronize_derivatives()) that are not, in total, generated by a single interface. The prPair, on the other hand, is used for storage of all low level fn evals that get evaluated in ApplicationInterface::map().

Referenced by ParamResponsePair::eval_id(), ParamResponsePair::eval_interface_ids(), ParamResponsePair::interface_id(), ParamResponsePair::operator=(), Dakota::operator==(), ParamResponsePair::read(), and ParamResponsePair::write().

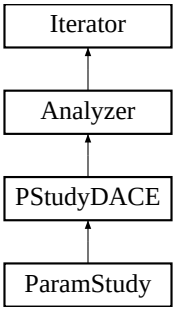
The documentation for this class was generated from the following file:

- ParamResponsePair.hpp

13.122 ParamStudy Class Reference

Class for vector, list, centered, and multidimensional parameter studies.

Inheritance diagram for ParamStudy:



Public Member Functions

- ParamStudy (ProblemDescDB &problem_db, Model &model)

  constructor
- ~ParamStudy ()

  destructor
- void pre_run ()

  pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori
- void extract_trends ()

  Mapping of the core_run() virtual function for the PStudy/DACE branch.
- void post_input ()

  read tabular data for post-run mode
- void post_run (std::ostream &s)

  post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way

Private Member Functions

- void sample ()

  performs the parameter study by sampling from a list of points
- void vector_loop ()

  performs the parameter study by sampling along a vector, starting from an initial point followed by numSteps increments along continous/discrete step vectors
- void centered_loop ()

  performs a number of plus and minus offsets for each parameter centered about an initial point
- void multidim_loop ()

  performs a full factorial combination for all intersections defined by a set of multidimensional partitions
- bool load_distribute_points (const String &points_filename, bool annotated, bool active_only)

  load list of points from data file and distribute among listCVPoints, listDIVPoints, listDSVPoints, and listDRVPoints

- template<typename OrdinalType , typename ScalarTypeA , typename ScalarTypeC , typename ScalarTypeDI , typename ScalarTypeDS , typename ScalarTypeDR >
  bool distribute (const Teuchos::SerialDenseVector< OrdinalType, ScalarTypeA > &all_data, Teuchos::SerialDenseVector< OrdinalType, ScalarTypeC > &c_data, Teuchos::SerialDenseVector< OrdinalType, ScalarTypeDI > &di_data, Teuchos::SerialDenseVector< OrdinalType, ScalarTypeDS > &ds_data, Teuchos::SerialDenseVector< OrdinalType, ScalarTypeDR > &dr_data)

  distributes incoming all vector in standard variable ordering among continuous, discrete int, discrete string, and discrete real vectors
- template<typename ScalarType >
  bool distribute (const std::vector< ScalarType > &all_data, std::vector< ScalarType > &c_data, std::vector< ScalarType > &di_data, std::vector< ScalarType > &ds_data, std::vector< ScalarType > &dr_data)

  distributes incoming all array in standard variable ordering among continuous, discrete int, discrete string, and discrete real arrays
- bool distribute_list_of_points (const RealVector &list_of_pts)

  distributes list_of_pts coming from user spec among listCVPoints, listDIVPoints, listDSVPoints, and listDRVPoints
- void final_point_to_step_vector ()

  compute step vectors from finalPoint, initial points, and numSteps
- void distribute_partitions ()

  compute step vectors from {cont,discInt,discString,discReal}VarPartitions and global bounds
- bool check_num_steps (int num_steps)

  perform error checks on numSteps
- bool check_step_vector (const RealVector &step_vector)

  perform error checks on numSteps
- bool check_final_point (const RealVector &final_pt)

  perform error checks on finalPoint
- bool check_steps_per_variable (const IntVector &steps_per_var)

  perform error checks on stepsPerVariable
- bool check_variable_partitions (const UShortArray &partitions)

  perform error checks on variable partitions
- bool check_finite_bounds ()

  check for finite variable bounds within iteratedModel, as required for computing partitions of finite ranges
- bool check_ranges_sets (int num_steps)

  sanity check for vector parameter study
- bool check_ranges_sets (const IntVector &c_steps, const IntVector &di_steps, const IntVector &ds_steps, const IntVector &dr_steps)

  sanity check for centered parameter study
- bool check_sets (const IntVector &c_steps, const IntVector &di_steps, const IntVector &ds_steps, const IntVector &dr_steps)

  sanity check for increments along int/real set dimensions
- int integer_step (int range, int num_steps) const

  check for integer remainder and return step
- int index_step (size_t start, size_t end, int num_steps) const

  check for out of bounds and index remainder and return step
- void c_step (size_t c_index, int increment, Variables &vars)

helper function for performing a continuous step in one variable

- void dri_step (size_t di_index, int increment, Variables &vars)

  helper function for performing a discrete step in an integer range variable
- void dsi_step (size_t di_index, int increment, const IntSet &values, Variables &vars)

  helper function for performing a discrete step in an integer set variable
- void dss_step (size_t ds_index, int increment, const StringSet &values, Variables &vars)

  helper function for performing a discrete step in an string set variable
- void dsr_step (size_t dr_index, int increment, const RealSet &values, Variables &vars)

  helper function for performing a discrete step in a real set variable
- void reset (Variables &vars)

  reset vars to initial point (center)
- void centered_header (const String &type, size_t var_index, int step, size_t hdr_index)

  store a centered parameter study header within allHeaders

Private Attributes

- size_t numEvals

  total number of parameter study evaluations computed from specification
- RealVectorArray listCVPoints

  array of continuous evaluation points for the list_parameter_study
- IntVectorArray listDIVPoints

  array of discrete int evaluation points for the list_parameter_study
- StringMulti2DArray listDSVPoints

  array of discrete string evaluation points for the list_parameter_study
- RealVectorArray listDRVPoints

  array of discrete real evaluation points for the list_parameter_study
- RealVector initialCVPoint

  the continuous start point for vector and centered parameter studies
- IntVector initialDIVPoint

  the discrete int start point for vector and centered parameter studies
- StringMultiArray initialDSVPoint

  the discrete string start point for vector and centered parameter studies
- RealVector initialDRVPoint

  the discrete real start point for vector and centered parameter studies
- RealVector finalCVPoint

  the continuous ending point for vector_parameter_study
- IntVector finalDIVPoint

  the discrete int range value or set index ending point for vector_parameter_study
- IntVector finalDSVPoint

  the discrete string set index ending point for vector_parameter_study
- IntVector finalDRVPoint

  the discrete real set index ending point for vector_parameter_study
- RealVector contStepVector

the n-dimensional continuous increment

- IntVector discIntStepVector

  the n-dimensional discrete integer range value or set index increment
- IntVector discStringStepVector

  the n-dimensional discrete string set index increment
- IntVector discRealStepVector

  the n-dimensional discrete real set index increment
- int numSteps

  the number of times continuous/discrete step vectors are applied for vector_parameter_study (a specification option)
- IntVector contStepsPerVariable

  number of offsets in the plus and the minus direction for each continuous variable in a centered_parameter_study
- IntVector discIntStepsPerVariable

  number of offsets in the plus and the minus direction for each discrete integer variable in a centered_parameter_-study
- IntVector discStringStepsPerVariable

  number of offsets in the plus and the minus direction for each discrete string variable in a centered_parameter_study
- IntVector discRealStepsPerVariable

  number of offsets in the plus and the minus direction for each discrete real variable in a centered_parameter_study
- UShortArray contVarPartitions

  number of partitions for each continuous variable in a multidim_parameter_study
- UShortArray discIntVarPartitions

  number of partitions for each discrete integer variable in a multidim_parameter_study
- UShortArray discStringVarPartitions

  number of partitions for each discrete string variable in a multidim_parameter_study
- UShortArray discRealVarPartitions

  number of partitions for each discrete real variable in a multidim_parameter_study

Additional Inherited Members

13.122.1 Detailed Description

Class for vector, list, centered, and multidimensional parameter studies.

The ParamStudy class contains several algorithms for performing parameter studies of different types. The vector parameter study steps along an n-dimensional vector from an arbitrary initial point to an arbitrary final point in a specified number of steps. The centered parameter study performs a number of plus and minus offsets in each coordinate direction around a center point. A multidimensional parameter study fills an n-dimensional hypercube based on bounds and a specified number of partitions for each dimension. And the list parameter study provides for a user specification of a list of points to evaluate, which allows general parameter investigations not fitting the structure of vector, centered, or multidim parameter studies.

13.122.2 Member Function Documentation

void pre_run ( ) `[virtual]`

pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

pre-run phase, which a derived iterator may optionally reimplement; when not present, pre-run is likely integrated into the derived run function. This is a virtual function; when re-implementing, a derived class must call its nearest parent's pre_run(), if implemented, typically *before* performing its own implementation steps.

Reimplemented from Iterator.

References Dakota::abort_handler(), SharedVariablesData::active_components_totals(), Analyzer::allHeaders, Analyzer::allVariables, ParamStudy::centered_loop(), Variables::continuous_variables(), ParamStudy::contStepsPerVariable, ParamStudy::contStepVector, ParamStudy::contVarPartitions, Dakota::copy_data(), Model::current_variables(), ParamStudy::discIntStepsPerVariable, ParamStudy::discIntStepVector, ParamStudy::discIntVarPartitions, ParamStudy::discRealStepsPerVariable, ParamStudy::discRealStepVector, ParamStudy::discRealVarPartitions, Variables::discrete_int_variables(), Variables::discrete_real_variables(), Variables::discrete_string_variables(), ParamStudy::discStringStepsPerVariable, ParamStudy::discStringStepVector, ParamStudy::discStringVarPartitions, ParamStudy::distribute_partitions(), ParamStudy::final_point_to_step_vector(), ParamStudy::finalCVPoint, ParamStudy::finalDIVPoint, ParamStudy::finalDRVPoint, ParamStudy::finalDSVPoint, ParamStudy::initialCVPoint, ParamStudy::initialDIVPoint, ParamStudy::initialDRVPoint, ParamStudy::initialDSVPoint, Iterator::iteratedModel, Iterator::method_enum_to_string(), Iterator::methodName, ParamStudy::multidim_loop(), Analyzer::numDiscreteStringVars, ParamStudy::numEvals, ParamStudy::numSteps, Iterator::outputLevel, Iterator::pre_run(), ParamStudy::sample(), Variables::shared_data(), ParamStudy::vector_loop(), and Dakota::write_ordered().

void post_run (std::ostream & *s*) `[virtual]`

post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/Responses and perform final analysis phase in a standalone way

Post-run phase, which a derived iterator may optionally reimplement; when not present, post-run is likely integrated into run. This is a virtual function; when re-implementing, a derived class must call its nearest parent's post_run(), typically *after* performing its own implementation steps.

Reimplemented from Analyzer.

References Analyzer::allResponses, Analyzer::allVariables, SensAnalysisGlobal::compute_correlations(), Model::discrete_set_string_values(), Iterator::iteratedModel, Iterator::methodName, Analyzer::post_run(), PStudyDACE::pStudyDACESensGlobal, and Iterator::subIteratorFlag.

bool load_distribute_points (const String & *points_filename,* bool *annotated,* bool *active_only*) `[private]`

load list of points from data file and distribute among listCVPoints, listDIVPoints, listDSVPoints, and listDRVPoints

Load from file and distribute points; using this function to manage construction of the temporary arrays. Historically all data was read as a real (mixture of values and indices), but now points_file is valued-based (reals, integers, strings) so file input matches tabular data output. Return false on success.

References Dakota::_NPOS, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Variables::copy(), Model::current_variables(), Model::discrete_int_lower_bounds(), Model::discrete_int_sets(), Model::discrete_int_upper_bounds(), Model::discrete_set_int_values(), Model::discrete_set_real_values(), Model::discrete_set_string_values(), Iterator::iteratedModel, ParamStudy::listCVPoints, ParamStudy::listDIVPoints, ParamStudy::listDRVPoints, ParamStudy::listDSVPoints, Analyzer::numContinuousVars, Analyzer::numDiscreteIntVars, Analyzer::numDiscreteRealVars, Analyzer::numDiscreteStringVars, ParamStudy::numEvals, and Dakota::set_value_to_index().

Referenced by ParamStudy::ParamStudy().

bool distribute_list_of_points (const RealVector & *list_of_pts*) [private]

distributes list_of_pts coming from user spec among listCVPoints, listDIVPoints, listDSVPoints, and listDRVPoints

Parse list of points into typed data containers; list_of_pts will contain values for continuous and discrete integer range, but indices for all discrete set types (int, string, real)

References Model::discrete_int_sets(), Model::discrete_set_int_values(), Model::discrete_set_real_values(), Model::discrete_set_string_values(), ParamStudy::distribute(), Iterator::iteratedModel, ParamStudy::listCVPoints, ParamStudy::listDIVPoints, ParamStudy::listDRVPoints, ParamStudy::listDSVPoints, Analyzer::numContinuousVars, Analyzer::numDiscreteIntVars, Analyzer::numDiscreteRealVars, Analyzer::numDiscreteStringVars, ParamStudy::numEvals, and Dakota::set_index_to_value().

Referenced by ParamStudy::ParamStudy().

The documentation for this class was generated from the following files:

- ParamStudy.hpp
- ParamStudy.cpp

13.123 partial_prp_equality Struct Reference

predicate for comparing ONLY the interfaceId and Vars attributes of PRPair

Public Member Functions

- bool operator() (const ParamResponsePair &database_pr, const ParamResponsePair &search_pr) const

  access operator

13.123.1 Detailed Description

predicate for comparing ONLY the interfaceId and Vars attributes of PRPair

The documentation for this struct was generated from the following file:

- PRPMultiIndex.hpp

13.124 partial_prp_hash Struct Reference

wrapper to delegate to the ParamResponsePair hash_value function

Public Member Functions

- std::size_t operator() (const ParamResponsePair &prp) const

  access operator

13.124.1 Detailed Description

wrapper to delegate to the ParamResponsePair hash_value function

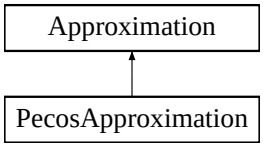
The documentation for this struct was generated from the following file:

- PRPMultiIndex.hpp

13.125 PecosApproximation Class Reference

Derived approximation class for global basis polynomials.

Inheritance diagram for PecosApproximation:



Public Member Functions

- PecosApproximation ()

  default constructor
- PecosApproximation (ProblemDescDB &problem_db, const SharedApproxData &shared_data)

  standard ProblemDescDB-driven constructor
- PecosApproximation (const SharedApproxData &shared_data)

  alternate constructor
- ~PecosApproximation ()

  destructor
- void expansion_coefficient_flag (bool coeff_flag)

  set pecosBasisApprox.configOptions.expansionCoeffFlag
- bool expansion_coefficient_flag () const

  get pecosBasisApprox.configOptions.expansionCoeffFlag
- void expansion_gradient_flag (bool grad_flag)

  set pecosBasisApprox.configOptions.expansionGradFlag
- bool expansion_gradient_flag () const

  get pecosBasisApprox.configOptions.expansionGradFlag
- void compute_component_effects ()

  Performs global sensitivity analysis using Sobol' Indices by computing component (main and interaction) effects.
- void compute_total_effects ()

  Performs global sensitivity analysis using Sobol' Indices by computing total effects.
- const Pecos::RealVector & sobol_indices () const

  return polyApproxRep->sobolIndices
- const Pecos::RealVector & total_sobol_indices () const

  return polyApproxRep->totalSobolIndices
- Pecos::ULongULongMap sparse_sobol_index_map () const

  return RegressOrthogPolyApproximation::sparseSobolIndexMap
- const Pecos::RealVector & dimension_decay_rates () const

  return OrthogPolyApproximation::decayRates
- void allocate_arrays ()

  invoke Pecos::PolynomialApproximation::allocate_arrays()
- Real mean ()

return the mean of the expansion, treating all variables as random

- Real mean (const Pecos::RealVector &x)

  return the mean of the expansion for a given parameter vector, treating a subset of the variables as random

- const Pecos::RealVector & mean_gradient ()

  return the gradient of the expansion mean for a given parameter vector, treating all variables as random

- const Pecos::RealVector & mean_gradient (const Pecos::RealVector &x, const Pecos::SizetArray &dvv)

  return the gradient of the expansion mean for a given parameter vector and given DVV, treating a subset of the variables as random

- Real variance ()

  return the variance of the expansion, treating all variables as random

- Real variance (const Pecos::RealVector &x)

  return the variance of the expansion for a given parameter vector, treating a subset of the variables as random

- const Pecos::RealVector & variance_gradient ()

  return the gradient of the expansion variance for a given parameter vector, treating all variables as random

- const Pecos::RealVector & variance_gradient (const Pecos::RealVector &x, const Pecos::SizetArray &dvv)

  return the gradient of the expansion variance for a given parameter vector and given DVV, treating a subset of the variables as random

- Real covariance (PecosApproximation *pecos_approx_2)

  return the covariance between two response expansions, treating all variables as random

- Real covariance (const Pecos::RealVector &x, PecosApproximation *pecos_approx_2)

  return the covariance between two response expansions, treating a subset of the variables as random

- Real delta_covariance (PecosApproximation *pecos_approx_2)

  return the change in covariance between two response expansions, treating all variables as random

- Real delta_covariance (const Pecos::RealVector &x, PecosApproximation *pecos_approx_2)

  return the change in covariance between two response expansions, treating a subset of the variables as random

- Real delta_mean ()

  return the change in mean between two response expansions, treating all variables as random

- Real delta_mean (const RealVector &x)

  return the change in mean between two response expansions, treating a subset of variables as random

- Real delta_std_deviation ()

  return the change in standard deviation between two response expansions, treating all variables as random

- Real delta_std_deviation (const RealVector &x)

  return the change in standard deviation between two response expansions, treating a subset of variables as random

- Real delta_beta (bool cdf_flag, Real z_bar)

  return the change in reliability index (mapped from z_bar) between two response expansions, treating all variables as random

- Real delta_beta (const RealVector &x, bool cdf_flag, Real z_bar)

  return the change in reliability index (mapped from z_bar) between two response expansions, treating a subset of variables as random

- Real delta_z (bool cdf_flag, Real beta_bar)

  return the change in response level (mapped from beta_bar) between two response expansions, treating all variables as random

- Real delta_z (const RealVector &x, bool cdf_flag, Real beta_bar)

return the change in response level (mapped from beta_bar) between two response expansions, treating a subset of the variables as random

- void compute_moments ()

  compute moments up to the order supported by the Pecos polynomial approximation
- void compute_moments (const Pecos::RealVector &x)

  compute moments in all-variables mode up to the order supported by the Pecos polynomial approximation
- const RealVector & moments () const

  return virtual Pecos::PolynomialApproximation::moments()
- const RealVector & expansion_moments () const

  return Pecos::PolynomialApproximation::expansionMoments
- const RealVector & numerical_integration_moments () const

  return Pecos::PolynomialApproximation::numericalMoments
- void standardize_moments (const Pecos::RealVector &central_moments, Pecos::RealVector &std_moments)

  standardize the central moments returned from Pecos
- Pecos::BasisApproximation & pecos_basis_approximation ()

  return pecosBasisApprox

Protected Member Functions

- Real value (const Variables &vars)

  retrieve the approximate function value for a given parameter vector
- const Pecos::RealVector & gradient (const Variables &vars)

  retrieve the approximate function gradient for a given parameter vector
- const Pecos::RealSymMatrix & hessian (const Variables &vars)

  retrieve the approximate function Hessian for a given parameter vector
- int min_coefficients () const

  return the minimum number of samples (unknowns) required to build the derived class approximation type in numVars dimensions
- void build ()

  builds the approximation from scratch
- void rebuild ()

  rebuilds the approximation incrementally
- void pop (bool save_data)

  removes entries from end of SurrogateData::{vars,resp}Data (last points appended, or as specified in args)
- void restore ()

  restores state prior to previous append()
- void finalize ()

  finalize approximation by applying all remaining trial sets
- void store ()

  store current approximation for later combination
- void combine (short corr_type)

  combine current approximation with previously stored approximation
- void print_coefficients (std::ostream &s, bool normalized)

  print the coefficient array computed in build()/rebuild()

- RealVector approximation_coefficients (bool normalized) const

  return expansion coefficients in a form consistent with the shared multi-index
- void approximation_coefficients (const RealVector &approx_coeffs, bool normalized)

  set expansion coefficients in a form consistent with the shared multi-index
- void coefficient_labels (std::vector< std::string > &coeff_labels) const

  print the coefficient array computed in build()/rebuild()

Private Member Functions

- void approx_type_to_basis_type (const String &approx_type, short &basis_type)

  utility to convert Dakota type string to Pecos type enumeration

Private Attributes

- Pecos::BasisApproximation pecosBasisApprox

  the Pecos basis approximation, encompassing OrthogPolyApproximation and InterpPolyApproximation
- Pecos::PolynomialApproximation * polyApproxRep

  convenience pointer to representation of Pecos polynomial approximation

Additional Inherited Members

13.125.1 Detailed Description

Derived approximation class for global basis polynomials.

The PecosApproximation class provides a global approximation based on basis polynomials. This includes orthogonal polynomials used for polynomial chaos expansions and interpolation polynomials used for stochastic collocation.

13.125.2 Member Function Documentation

void build ( ) `[inline], [protected], [virtual]`

builds the approximation from scratch

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented from Approximation.

References Approximation::build(), and PecosApproximation::pecosBasisApprox.

void rebuild ( ) `[inline], [protected], [virtual]`

rebuilds the approximation incrementally

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented from Approximation.

References PecosApproximation::pecosBasisApprox.

void pop (bool *save_data*) [inline], [protected], [virtual]

removes entries from end of SurrogateData::{vars,resp}Data (last points appended, or as specified in args)

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented from Approximation.

References PecosApproximation::pecosBasisApprox, and Approximation::pop().

void restore ( ) [inline], [protected], [virtual]

restores state prior to previous append()

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented from Approximation.

References PecosApproximation::pecosBasisApprox, and Approximation::restore().

void finalize ( ) [inline], [protected], [virtual]

finalize approximation by applying all remaining trial sets

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented from Approximation.

References Approximation::finalize(), and PecosApproximation::pecosBasisApprox.

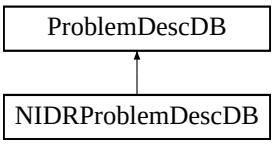
The documentation for this class was generated from the following files:

- PecosApproximation.hpp
- PecosApproximation.cpp

13.126 ProblemDescDB Class Reference

The database containing information parsed from the DAKOTA input file.

Inheritance diagram for ProblemDescDB:



Public Member Functions

- ProblemDescDB ()

  default constructor
- ProblemDescDB (ParallelLibrary &parallel_lib)

  standard constructor
- ProblemDescDB (const ProblemDescDB &db)

  copy constructor
- ~ProblemDescDB ()

destructor

- ProblemDescDB operator= (const ProblemDescDB &db)

  assignment operator

- void parse_inputs (const ProgramOptions &prog_opts, DbCallbackFunctionPtr callback=NULL, void *callback_data=NULL)

  Parses the input file or input string if present and executes callbacks. Does not perform any validation.

- void check_and_broadcast (const ProgramOptions &prog_opts)

  performs check_input, broadcast, and post_process, but for now, allowing separate invocation through the public API as well

- void check_input ()

  verifies that there is at least one of each of the required keywords in the dakota input file

- void broadcast ()

  invokes send_db_buffer() and receive_db_buffer() to broadcast DB data across the processor allocation. Used by manage_inputs().

- void post_process ()

  post-processes the (minimal) input specification to assign default variables/responses specification arrays. Used by manage_inputs().

- void lock ()

  Locks the database in order to prevent data access when the list nodes may not be set properly. Unlocked by a set nodes operation.

- void unlock ()

  Explicitly unlocks the database. Use with care.

- void set_db_list_nodes (const String &method_tag)

  set dataMethodIter based on a method identifier string to activate a particular method specification in dataMethodList and use pointers from this method specification to set all other list iterators.

- void set_db_list_nodes (size_t method_index)

  set dataMethodIter based on an index within dataMethodList to activate a particular method specification and use pointers from this method specification to set all other list iterators.

- void resolve_top_method (bool set_model_nodes=true)

  For a (default) environment lacking a top method pointer, this function is used to determine which of several potential method specifications corresponds to the top method and then sets the list nodes accordingly.

- void set_db_method_node (const String &method_tag)

  set dataMethodIter based on a method identifier string to activate a particular method specification (only).

- void set_db_method_node (size_t method_index)

  set dataMethodIter based on an index within dataMethodList to activate a particular method specification (only).

- size_t get_db_method_node ()

  return the index of the active node in dataMethodList

- void set_db_model_nodes (const String &model_tag)

  set the model list iterators (dataModelIter, dataVariablesIter, dataInterfaceIter, and dataResponsesIter) based on the model identifier string

- void set_db_model_nodes (size_t model_index)

  set the model list iterators (dataModelIter, dataVariablesIter, dataInterfaceIter, and dataResponsesIter) based on an index within dataModelList

- size_t get_db_model_node ()

  return the index of the active node in dataModelList

- void set_db_variables_node (const String &variables_tag)

  set dataVariablesIter based on the variables identifier string
- void set_db_interface_node (const String &interface_tag)

  set dataInterfaceIter based on the interface identifier string
- void set_db_responses_node (const String &responses_tag)

  set dataResponsesIter based on the responses identifier string
- ParallelLibrary & parallel_library () const

  return the parallelLib reference
- IteratorList & iterator_list ()

  return a list of all Iterator objects that have been instantiated
- ModelList & model_list ()

  return a list of all Model objects that have been instantiated
- VariablesList & variables_list ()

  return a list of all Variables objects that have been instantiated
- InterfaceList & interface_list ()

  return a list of all Interface objects that have been instantiated
- ResponseList & response_list ()

  return a list of all Response objects that have been instantiated
- const RealVector & get_rv (const String &entry_name) const

  get a RealVector out of the database based on an identifier string
- const IntVector & get_iv (const String &entry_name) const

  get an IntVector out of the database based on an identifier string
- const BitArray & get_ba (const String &entry_name) const

  get a BitArray out of the database based on an identifier string
- const SizetArray & get_sza (const String &entry_name) const

  get an SizetArray out of the database based on an identifier string
- const UShortArray & get_usa (const String &entry_name) const

  get an UShortArray out of the database based on an identifier string
- const RealSymMatrix & get_rsm (const String &entry_name) const

  get a RealSymMatrix out of the database based on an identifier string
- const RealVectorArray & get_rva (const String &entry_name) const

  get a RealVectorArray out of the database based on an identifier string
- const IntVectorArray & get_iva (const String &entry_name) const

  get an IntVectorArray out of the database based on an identifier string
- const IntSet & get_is (const String &entry_name) const

  get an IntSet out of the database based on an identifier string
- const IntSetArray & get_isa (const String &entry_name) const

  get an IntSetArray out of the database based on an identifier string
- const StringSetArray & get_ssa (const String &entry_name) const

  get an StringSetArray out of the database based on an identifier string
- const RealSetArray & get_rsa (const String &entry_name) const

  get a RealSetArray out of the database based on an identifier string
- const IntRealMapArray & get_irma (const String &entry_name) const

get an IntRealMapArray out of the database based on an identifier string

- const StringRealMapArray & get_srma (const String &entry_name) const

  get an StringRealMapArray out of the database based on an identifier string
- const RealRealMapArray & get_rrma (const String &entry_name) const

  get a RealRealMapArray out of the database based on an identifier string
- const RealRealPairRealMapArray & get_rrrma (const String &entry_name) const

  get a RealRealPairRealMapArray out of the database based on an identifier string
- const IntIntPairRealMapArray & get_iirma (const String &entry_name) const

  get an IntIntPairRealMapArray out of the database based on an identifier string
- const StringArray & get_sa (const String &entry_name) const

  get a StringArray out of the database based on an identifier string
- const String2DArray & get_s2a (const String &entry_name) const

  get a String2DArray out of the database based on an identifier string
- const String & get_string (const String &entry_name) const

  get a String out of the database based on an identifier string
- const Real & get_real (const String &entry_name) const

  get a Real out of the database based on an identifier string
- int get_int (const String &entry_name) const

  get an int out of the database based on an identifier string
- short get_short (const String &entry_name) const

  get a short out of the database based on an identifier string
- unsigned short get_ushort (const String &entry_name) const

  get an unsigned short out of the database based on an identifier string
- size_t get_sizet (const String &entry_name) const

  get a size_t out of the database based on an identifier string
- bool get_bool (const String &entry_name) const

  get a bool out of the database based on an identifier string
- void ** get_voidss (const String &entry_name) const

  *for getting a void**, e.g., &dlLib*
- void insert_node (const DataEnvironment &data_env)

  set the DataEnvironment object
- void insert_node (const DataMethod &data_method)

  add a DataMethod object to the dataMethodList
- void insert_node (const DataModel &data_model)

  add a DataModel object to the dataModelList
- void insert_node (DataVariables &data_variables)

  add a DataVariables object to the dataVariablesList
- void insert_node (const DataInterface &data_interface)

  add a DataInterface object to the dataInterfaceList
- void insert_node (const DataResponses &data_responses)

  add a DataResponses object to the dataResponsesList
- void set (const String &entry_name, const RealVector &rv)

  set a RealVector within the database based on an identifier string

- void set (const String &entry_name, const IntVector &iv)

  set an IntVector within the database based on an identifier string
- void set (const String &entry_name, const BitArray &ba)

  set a BitArray within the database based on an identifier string
- void set (const String &entry_name, const RealSymMatrix &rsm)

  set a RealMatrix within the database based on an identifier string
- void set (const String &entry_name, const RealVectorArray &rva)

  set a RealVectorArray within the database based on an identifier string
- void set (const String &entry_name, const IntVectorArray &iva)

  set an IntVectorArray within the database based on an identifier string
- void set (const String &entry_name, const IntSetArray &isa)

  set an IntSetArray within the database based on an identifier string
- void set (const String &entry_name, const RealSetArray &rsa)

  set a RealSetArray within the database based on an identifier string
- void set (const String &entry_name, const IntRealMapArray &irma)

  set an IntRealMapArray within the database based on an identifier string
- void set (const String &entry_name, const StringRealMapArray &srma)

  set a StringRealMapArray within the database based on an identifier string
- void set (const String &entry_name, const RealRealMapArray &rrma)

  set a RealRealMapArray within the database based on an identifier string
- void set (const String &entry_name, const RealRealPairRealMapArray &iirma)

  set a RealRealPairRealMapArray in the db based on an identifier string
- void set (const String &entry_name, const IntIntPairRealMapArray &iirma)

  set an IntIntPairRealMapArray in the db based on an identifier string
- void set (const String &entry_name, const StringArray &sa)

  set a StringArray within the database based on an identifier string
- int min_procs_per_ea ()

  compute minimum evaluation partition size based on lower level overrides
- int max_procs_per_ea ()

  compute maximum evaluation partition size based on lower level concurrency
- int min_procs_per_ie ()

  compute minimum iterator partition size based on lower level overrides
- int max_procs_per_ie (int max_eval_concurrency)

  compute maximum iterator partition size based on lower level concurrency
- int min_procs_per_mi (int min_procs_per_iter, int ppi_spec, int num_i_serv_spec)

  compute minimum meta-iterator partition size based on lower level overrides
- int min_procs_per_mi ()

  compute minimum meta-iterator partition size based on lower level overrides
- int max_procs_per_mi (int max_procs_per_iter, int ppi_spec, int num_i_serv_spec, short i_sched_spec)

  compute maximum meta-iterator partition size based on lower level overrides
- int max_procs_per_mi (int max_eval_concurrency)

  compute maximum meta-iterator partition size based on lower level concurrency
- bool is_null () const

  function to check dbRep (does this envelope contain a letter)

Static Public Member Functions

- static int min_procs_per_ea (int ppa_spec, int num_a_serv_spec)

  compute minimum evaluation partition size based on passed overrides
- static int max_procs_per_ea (int num_drivers, int num_a_serv_spec, short a_sched_spec, int alac_spec)

  compute maximum evaluation partition size based on passed overrides

Protected Member Functions

- ProblemDescDB (BaseConstructor, ParallelLibrary &parallel_lib)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
- virtual void derived_parse_inputs (const ProgramOptions &prog_opts)

  derived class specifics within parse_inputs()
- virtual void derived_broadcast ()

  derived class specifics within broadcast()
- virtual void derived_post_process ()

  derived class specifics within post_process()

Protected Attributes

- DataEnvironment environmentSpec

  the environment specification (only one allowed) resulting from a call to environment_kwhandler() or insert_node()
- std::list< DataMethod > dataMethodList

  list of method specifications, one for each call to method_kwhandler() or insert_node()
- std::list< DataModel > dataModelList

  list of model specifications, one for each call to model_kwhandler() or insert_node()
- std::list< DataVariables > dataVariablesList

  list of variables specifications, one for each call to variables_kwhandler() or insert_node()
- std::list< DataInterface > dataInterfaceList

  list of interface specifications, one for each call to interface_kwhandler() or insert_node()
- std::list< DataResponses > dataResponsesList

  list of responses specifications, one for each call to responses_kwhandler() or insert_node()
- size_t environmentCntr

  counter for environment specifications used in check_input

Private Member Functions

- const Iterator & get_iterator ()

  retrieve an existing Iterator, if it exists, or instantiate a new one
- const Iterator & get_iterator (Model &model)

  retrieve an existing Iterator, if it exists, or instantiate a new one
- const Model & get_model ()

  retrieve an existing Model, if it exists, or instantiate a new one
- const Variables & get_variables ()

retrieve an existing Variables, if it exists, or instantiate a new one

- const Interface & get_interface ()

  retrieve an existing Interface, if it exists, or instantiate a new one

- const Response & get_response (const Variables &vars)

  retrieve an existing Response, if it exists, or instantiate a new one

- ProblemDescDB * get_db (ParallelLibrary &parallel_lib)

  Used by the envelope constructor to instantiate the correct letter class.

- void send_db_buffer ()

  MPI send of a large buffer containing environmentSpec and all objects in dataMethodList, dataModelList, dataVariablesList, dataInterfaceList, and dataResponsesList. Used by manage_inputs().

- void receive_db_buffer ()

  MPI receive of a large buffer containing environmentSpec and all objects in dataMethodList, dataModelList, dataVariablesList, dataInterfaceList, and dataResponsesList. Used by manage_inputs().

- bool model_has_interface (DataModelRep *model_rep) const

  helper function for determining whether an interface specification should be active, based on model type

- void echo_input_file (const ProgramOptions &prog_opts)

  echo the (potentially) specified input file or string to stdout

Private Attributes

- ParallelLibrary & parallelLib

  reference to the parallel_lib object passed from main

- std::list< DataMethod >::iterator dataMethodIter

  iterator identifying the active list node in dataMethodList

- std::list< DataModel >::iterator dataModelIter

  iterator identifying the active list node in dataModelList

- std::list< DataVariables >
  ::iterator dataVariablesIter

  iterator identifying the active list node in dataVariablesList

- std::list< DataInterface >
  ::iterator dataInterfaceIter

  iterator identifying the active list node in dataInterfaceList

- std::list< DataResponses >
  ::iterator dataResponsesIter

  iterator identifying the active list node in dataResponsesList

- IteratorList iteratorList

  list of iterator objects, one for each method specification

- ModelList modelList

  list of model objects, one for each model specification

- VariablesList variablesList

  list of variables objects, one for each variables specification

- InterfaceList interfaceList

  list of interface objects, one for each interface specification

- ResponseList responseList

list of response objects, one for each responses specification

- bool methodDBLocked

  prevents use of get_<type> retrieval and set_<type> update functions prior to setting the list node for the active method specification

- bool modelDBLocked

  prevents use of get_<type> retrieval and set_<type> update functions prior to setting the list node for the active model specification

- bool variablesDBLocked

  prevents use of get_<type> retrieval and set_<type> update functions prior to setting the list node for the active variables specification

- bool interfaceDBLocked

  prevents use of get_<type> retrieval and set_<type> update functions prior to setting the list node for the active interface specification

- bool responsesDBLocked

  prevents use of get_<type> retrieval and set_<type> update functions prior to setting the list node for the active responses specification

- ProblemDescDB * dbRep

  pointer to the letter (initialized only for the envelope)

- int referenceCount

  number of objects sharing dbRep

Friends

- class Model

  Model requires access to get_variables() and get_response()

- class SingleModel

  SingleModel requires access to get_interface()

- class HierarchSurrModel

  HierarchSurrModel requires access to get_model()

- class DataFitSurrModel

  DataFitSurrModel requires access to get_iterator() and get_model()

- class NestedModel

  NestedModel requires access to get_interface(), get_response(), get_iterator(), and get_model()

- class Environment

  Environment requires access to get_iterator()

- class IteratorScheduler

  Environment requires access to get_iterator()

- class Iterator

  Iterator requires access to get_model()

- class MetaIterator

  Iterator requires access to get_model()

- class SeqHybridMetaIterator

  SeqHybridMetaIterator requires access to get_model()

- class CollabHybridMetaIterator

CollabHybridMetaIterator requires access to get_model()

- class ConcurrentMetaIterator

  ConcurrentMetaIterator requires access to get_model()

- class SurrBasedLocalMinimizer

  SurrBasedLocalMinimizer requires access to get_iterator()

- class SurrBasedGlobalMinimizer

  SurrBasedGlobalMinimizer requires access to get_iterator()

13.126.1 Detailed Description

The database containing information parsed from the DAKOTA input file.

The ProblemDescDB class is a database for DAKOTA input file data that is populated by a parser defined in a derived class. When the parser reads a complete keyword, it populates a data class object (DataEnvironment, DataMethod, DataVariables, DataInterface, or DataResponses) and, for all cases except environment, appends the object to a linked list (dataMethodList, dataVariablesList, dataInterfaceList, or dataResponsesList). No environment linked list is used since only one environment specification is allowed.

13.126.2 Constructor & Destructor Documentation

ProblemDescDB ( )

default constructor

The default constructor: dbRep is NULL in this case. This makes it necessary to check for NULL in the copy constructor, assignment operator, and destructor.

ProblemDescDB (ParallelLibrary & *parallel_lib*)

standard constructor

This is the envelope constructor which uses problem_db to build a fully populated db object. It only needs to extract enough data to properly execute get_db(problem_db), since the constructor overloaded with BaseConstructor builds the actual base class data inherited by the derived classes.

References Dakota::abort_handler(), ProblemDescDB::dbRep, and ProblemDescDB::get_db().

ProblemDescDB (const ProblemDescDB & *db*)

copy constructor

Copy constructor manages sharing of dbRep and incrementing of referenceCount.

References ProblemDescDB::dbRep, and ProblemDescDB::referenceCount.

~ProblemDescDB ( )

destructor

Destructor decrements referenceCount and only deletes dbRep when referenceCount reaches zero.

References Dakota::Dak_pddb, ProblemDescDB::dbRep, and ProblemDescDB::referenceCount.

ProblemDescDB (BaseConstructor , ParallelLibrary & *parallel_lib*) [protected]

constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor is the one which must build the base class data for all derived classes. get_db() instantiates a derived class letter and the derived constructor selects this base class constructor in its initialization list (to avoid the recursion of the base class constructor calling get_db() again). Since the letter IS the representation, its representation pointer is set to NULL (an uninitialized pointer causes problems in ~ProblemDescDB).

13.126.3 Member Function Documentation

ProblemDescDB operator= (const ProblemDescDB & *db*)

assignment operator

Assignment operator decrements referenceCount for old dbRep, assigns new dbRep, and increments referenceCount for new dbRep.

References ProblemDescDB::dbRep, and ProblemDescDB::referenceCount.

void parse_inputs (const ProgramOptions & *prog_opts*, DbCallbackFunctionPtr *callback* = *NULL*, void * *callback_data* = *NULL*)

Parses the input file or input string if present and executes callbacks. Does not perform any validation.

DB setup phase 1: parse the input file and execute callback functions if present. Rank 0 only.

DB setup phase 2: optionally insert additional data via late sets. Rank 0 only.

References Dakota::abort_handler(), ProblemDescDB::dbRep, ProblemDescDB::derived_parse_inputs(), ProgramOptions::echo_input(), ProblemDescDB::echo_input_file(), ProgramOptions::input_file(), ProgramOptions::input_string(), ProblemDescDB::parallelLib, ProblemDescDB::parse_inputs(), and ParallelLibrary::world_rank().

Referenced by Environment::parse(), and ProblemDescDB::parse_inputs().

void check_and_broadcast (const ProgramOptions & *prog_opts*)

performs check_input, broadcast, and post_process, but for now, allowing separate invocation through the public API as well

DB setup phase 3: perform basic checks on keywords counts in current DB state, then sync to all processors.

References ProblemDescDB::broadcast(), ProblemDescDB::check_and_broadcast(), ProblemDescDB::check_input(), ProblemDescDB::dbRep, ProblemDescDB::parallelLib, ProblemDescDB::post_process(), and ParallelLibrary::world_rank().

Referenced by ProblemDescDB::check_and_broadcast(), LibraryEnvironment::done_modifying_db(), and Environment::parse().

void check_input ( )

verifies that there is at least one of each of the required keywords in the dakota input file

NOTE: when using library mode in a parallel application, check_input() should either be called only on worldRank 0, or it should follow a matched send_db_buffer()/receive_db_buffer() pair.

References Dakota::abort_handler(), ProblemDescDB::check_input(), ParallelLibrary::command_line_post_run_input(), ParallelLibrary::command_line_post_run_output(), ParallelLibrary::command_line_pre_run_input(), ParallelLibrary::command_line_pre_run_output(), ParallelLibrary::command_line_run_input(), ParallelLibrary::command_line_run_output(), ParallelLibrary::command_line_user_modes(), ProblemDescDB::dataInterfaceList, ProblemDescDB::dataMethodList, ProblemDescDB::dataModelList, ProblemDescDB::dataResponsesList, ProblemDescDB-

::dataVariablesList, ProblemDescDB::dbRep, ProblemDescDB::environmentCntr, ProblemDescDB::parallelLib, and Dakota::strbegins().

Referenced by ProblemDescDB::check_and_broadcast(), and ProblemDescDB::check_input().

void post_process ( )

post-processes the (minimal) input specification to assign default variables/responses specification arrays. Used by manage_inputs().

When using library mode in a parallel application, post_process() should be called on all processors following broadcast() of a minimal problem specification.

References ProblemDescDB::dbRep, and ProblemDescDB::derived_post_process().

Referenced by ProblemDescDB::check_and_broadcast().

ProblemDescDB * get_db (ParallelLibrary & *parallel_lib*) `[private]`

Used by the envelope constructor to instantiate the correct letter class.

Initializes dbRep to the appropriate derived type. The standard derived class constructors are invoked.

References Dakota::Dak_pddb.

Referenced by ProblemDescDB::ProblemDescDB().

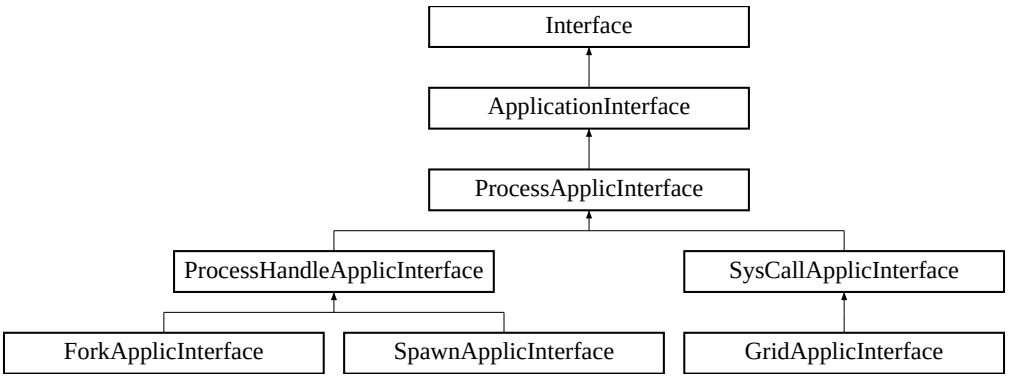
The documentation for this class was generated from the following files:

- ProblemDescDB.hpp
- ProblemDescDB.cpp

13.127 ProcessApplicInterface Class Reference

Derived application interface class that spawns a simulation code using a separate process and communicates with it through files.

Inheritance diagram for ProcessApplicInterface:



Public Member Functions

- ProcessApplicInterface (const ProblemDescDB &problem_db)

  constructor
- ~ProcessApplicInterface ()

  destructor

Protected Member Functions

- void derived_map (const Variables &vars, const ActiveSet &set, Response &response, int fn_eval_id)

  Called by map() and other functions to execute the simulation in synchronous mode. The portion of performing an evaluation that is specific to a derived class.
- void derived_map_asynch (const ParamResponsePair &pair)

  Called by map() and other functions to execute the simulation in asynchronous mode. The portion of performing an asynchronous evaluation that is specific to a derived class.
- const StringArray & analysis_drivers () const

  retrieve the analysis drivers specification for application interfaces
- void file_cleanup () const
- void remove_params_results_files (const bfs::path &params_path, const bfs::path &results_path) const

  Remove (potentially autotagged for multiple programs) parameters and results files with passed root names.
- void autotag_files (const bfs::path &params_path, const bfs::path &results_path, const String &eval_id_tag) const

  Utility to automatically tag parameters and results files with passed root names (the files may already need per-program tagging)
- virtual void map_bookkeeping (pid_t pid, int fn_eval_id)=0

  bookkeeping of process and evaluation ids for asynchronous maps
- virtual pid_t create_evaluation_process (bool block_flag)=0

  Spawn the evaluation by managing the input filter, analysis drivers, and output filter. Called from derived_map() & derived_map_asynch().
- void synchronous_local_analyses (int start, int end, int step)

  execute analyses synchronously on the local processor
- void define_filenames (const String &eval_id_tag)

  define modified filenames from user input by handling Unix temp file and optionally tagging with given eval_id_tag
- void write_parameters_files (const Variables &vars, const ActiveSet &set, const Response &response, const int id)

  write the parameters data and response request data to one or more parameters files (using one or more invocations of write_parameters_file()) in either standard or aprepro format
- void read_results_files (Response &response, const int id, const String &eval_id_tag)

  read the response object from one or more results files using full eval_id_tag passed
- bfs::path get_workdir_name ()

  construct a work directory name (tmp or named), with optional tag
- void prepare_process_environment ()

  set PATH, environment variables, and change directory prior to fork/system/spawn
- void reset_process_environment ()

  reset PATH and current directory after system/spawn (workdir case)

Protected Attributes

- bool fileTagFlag

  flags tagging of parameter/results files
- bool fileSaveFlag

  flags retention of parameter/results files

- bool commandLineArgs

  flag indicating use of passing of filenames as command line arguments to the analysis drivers and input/output filters
- bool apreproFlag

  flag indicating use of the APREPRO (the Sandia "A PRE PROcessor" utility) format for parameter files
- bool multipleParamsFiles

  flag indicating the need for separate parameters files for multiple analysis drivers
- std::string iFilterName

  the name of the input filter (input_filter user specification)
- std::string oFilterName

  the name of the output filter (output_filter user specification)
- std::vector< String > programNames

  the names of the analysis code programs (analysis_drivers user specification)
- std::string specifiedParamsFileName

  the name of the parameters file from user specification
- std::string paramsFileName

  the parameters file name actually used (modified with tagging or temp files); only valid from define_filenames to write_parameters_files
- std::string paramsFileWritten

  actual, qualified name of the params file written, possibly with workdir
- std::string specifiedResultsFileName

  the name of the results file from user specification
- std::string resultsFileName

  the results file name actually used (modified with tagging or temp files); only valid from define_filenames to write_parameters_files
- std::string resultsFileWritten

  actual, qualified name of the results file written, possibly with workdir
- std::string fullEvalId

  complete evalIdTag, possibly including hierarchical tagging and final eval id, but not program numbers, for passing to write_parameters_files
- bool allowExistingResults

  by default analysis code interfaces delete results files if they exist; user may override with this flag and we'll try to gather and only fork if needed
- std::map< int, PathTriple > fileNameMap

  Maps function evaluation ID to triples (parameters, results, and workdir) paths used in spawning function evaluations. Workdir will be empty if not created specifically for this eval.
- bool useWorkdir

  whether to use a work_directory
- std::string workDirName

  work_directory name, if specified...
- bool dirTag

  whether to tag the working directory
- bool dirSave

  whether dir_save was specified

- bfs::path curWorkdir

  active working directory for this evaluation; valid only from define_filenames to create_evaluation_process
- bfs::path createdDir

  non-empty if created for this eval; valid only from define_filenames to write_parameters_files
- StringArray linkFiles

  template directory (if specified)
- StringArray copyFiles

  template files (if specified)
- bool templateReplace

  whether to replace existing files

Private Member Functions

- void write_parameters_file (const Variables &vars, const ActiveSet &set, const Response &response, const std::string &prog, const std::vector< String > &an_comps, const std::string &params_fname)

  write the variables, active set vector, derivative variables vector, and analysis components to the specified parameters file in either standard or aprepro format

Private Attributes

- String2DArray analysisComponents

  the set of optional analysis components used by the analysis drivers (from the analysis_components interface specification)

13.127.1 Detailed Description

Derived application interface class that spawns a simulation code using a separate process and communicates with it through files.

ProcessApplicInterface is subclassed for process handles or file completion testing.

13.127.2 Member Function Documentation

void file_cleanup ( ) const `[protected], [virtual]`

Remove any files and directories still referenced in the fileNameMap

Reimplemented from Interface.

References WorkdirHelper::concat_path(), ProcessApplicInterface::dirSave, ProcessApplicInterface::fileNameMap, ProcessApplicInterface::fileSaveFlag, ProcessApplicInterface::iFilterName, ProcessApplicInterface::multipleParamsFiles, ProcessApplicInterface::programNames, and WorkdirHelper::recursive_remove().

void autotag_files (const bfs::path & *params_path,* const bfs::path & *results_path,* const String & *eval_id_tag*) const `[protected]`

Utility to automatically tag parameters and results files with passed root names (the files may already need per-program tagging)

Move specified params and results files to unique tagged versions when needed

References WorkdirHelper::concat_path(), ProcessApplicInterface::iFilterName, ProcessApplicInterface::multipleParamsFiles, ProcessApplicInterface::oFilterName, Interface::outputLevel, ProcessApplicInterface::programNames,

WorkdirHelper::rename(), ProcessApplicInterface::specifiedParamsFileName, ProcessApplicInterface::specified-ResultsFileName, and ApplicationInterface::suppressOutput.

Referenced by ProcessApplicInterface::read_results_files().

void synchronous_local_analyses (int *start,* int *end,* int *step*) [inline], [protected]

execute analyses synchronously on the local processor

Execute analyses synchronously in succession on the local processor (start to end in step increments). Modeled after ApplicationInterface::synchronous_local_evaluations().

References ApplicationInterface::synchronous_local_analysis().

Referenced by ProcessHandleApplicInterface::create_evaluation_process().

void prepare_process_environment ( ) [protected]

set PATH, environment variables, and change directory prior to fork/system/spawn

Guidance: environment (PATH, current directory) should be set immediately before Dakota spawns a process and reset immediately afterwards (except fork which never returns)

References WorkdirHelper::change_directory(), ProcessApplicInterface::curWorkdir, Interface::outputLevel, ProcessApplicInterface::paramsFileName, ProcessApplicInterface::resultsFileName, WorkdirHelper::set_environment(), WorkdirHelper::set_preferred_path(), and ProcessApplicInterface::useWorkdir.

Referenced by SpawnApplicInterface::create_analysis_process(), ForkApplicInterface::create_analysis_process(), SysCallApplicInterface::spawn_analysis_to_shell(), SysCallApplicInterface::spawn_evaluation_to_shell(), SysCall-ApplicInterface::spawn_input_filter_to_shell(), and SysCallApplicInterface::spawn_output_filter_to_shell().

void reset_process_environment ( ) [protected]

reset PATH and current directory after system/spawn (workdir case)

Undo anything done prior to spawn

References Interface::outputLevel, WorkdirHelper::reset(), WorkdirHelper::startup_pwd(), and ProcessApplic-Interface::useWorkdir.

Referenced by SpawnApplicInterface::create_analysis_process(), ForkApplicInterface::create_analysis_process(), SysCallApplicInterface::spawn_analysis_to_shell(), SysCallApplicInterface::spawn_evaluation_to_shell(), SysCall-ApplicInterface::spawn_input_filter_to_shell(), and SysCallApplicInterface::spawn_output_filter_to_shell().

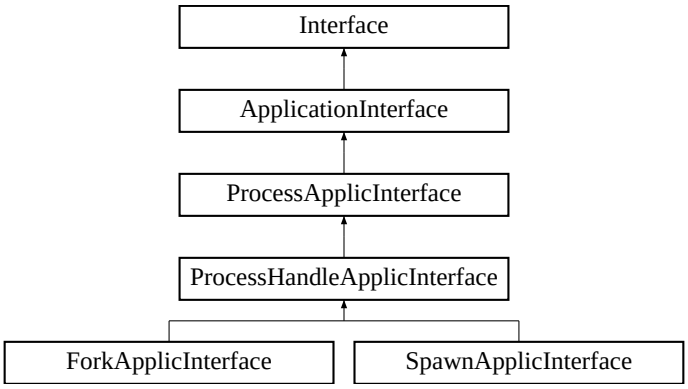
The documentation for this class was generated from the following files:

- ProcessApplicInterface.hpp
- ProcessApplicInterface.cpp

13.128 ProcessHandleApplicInterface Class Reference

Derived application interface class that spawns a simulation code using a separate process, receives a process identifier, and communicates with the spawned process through files.

Inheritance diagram for ProcessHandleApplicInterface:



Public Member Functions

- ProcessHandleApplicInterface (const ProblemDescDB &problem_db)

  constructor
- ~ProcessHandleApplicInterface ()

  destructor

Protected Member Functions

- int synchronous_local_analysis (int analysis_id)
- void init_communicators_checks (int max_eval_concurrency)
- void set_communicators_checks (int max_eval_concurrency)
- void map_bookkeeping (pid_t pid, int fn_eval_id)

  bookkeeping of process and evaluation ids for asynchronous maps
- pid_t create_evaluation_process (bool block_flag)
- virtual pid_t create_analysis_process (bool block_flag, bool new_group)=0

  spawn a child process for an analysis component within an evaluation
- virtual size_t wait_local_analyses ()=0

  wait for asynchronous analyses on the local processor, completing at least one job
- virtual size_t test_local_analyses_send (int analysis_id)=0

  test for asynchronous analysis completions on the local processor and return results for any completions by sending messages
- virtual void join_evaluation_process_group (bool new_group)

  create (if new_group) and join the process group for asynch evaluations
- virtual void join_analysis_process_group (bool new_group)

  create (if new_group) and join the process group for asynch analyses
- virtual void evaluation_process_group_id (pid_t pgid)

  set evalProcGroupId
- virtual pid_t evaluation_process_group_id () const

  return evalProcGroupId
- virtual void analysis_process_group_id (pid_t pgid)

  set analysisProcGroupId

- virtual pid_t analysis_process_group_id () const

  return analysisProcGroupId
- void process_local_evaluation (PRPQueue &prp_queue, const pid_t pid)

  Common processing code used by {wait,test}_local_evaluations.
- void check_wait (pid_t pid, int status)

  check the exit status of a forked process and abort if an error code was returned
- void asynchronous_local_analyses (int start, int end, int step)

  execute analyses asynchronously on the local processor
- void serve_analyses_asynch ()

  serve the analysis scheduler and execute analysis jobs asynchronously
- void ifilter_argument_list ()

  set argList for execution of the input filter
- void ofilter_argument_list ()

  set argList for execution of the output filter
- void driver_argument_list (int analysis_id)

  set argList for execution of the specified analysis driver
- void create_command_arguments (boost::shared_array< const char * > &av, StringArray &driver_and_args)

  parse argList into argument array av suitable for passing to execvp, appending parameters and results filenames if requested by commandLineArgs

Protected Attributes

- std::map< pid_t, int > evalProcessIdMap

  map of fork process id's to function evaluation id's for asynchronous evaluations
- std::map< pid_t, int > analysisProcessIdMap

  map of fork process id's to analysis job id's for asynchronous analyses
- std::vector< std::string > argList

  *an array of strings for use with execvp(const char *, char * const *). These are converted to an array of const char*'s in fork_program().*

13.128.1 Detailed Description

Derived application interface class that spawns a simulation code using a separate process, receives a process identifier, and communicates with the spawned process through files.

ProcessHandleApplicInterface is subclassed for fork/execvp/waitpid (Unix) and spawnvp (Windows).

13.128.2 Constructor & Destructor Documentation

ProcessHandleApplicInterface (const ProblemDescDB & *problem_db*) `[inline]`

constructor

argList sized 3 for [driver name, input file, output file]

13.128.3 Member Function Documentation

int synchronous_local_analysis (int *analysis_id*) [inline], [protected], [virtual]

This code provides the derived function used by ApplicationInterface:: serve_analyses_synch() as well as a convenience function for ProcessHandleApplicInterface::synchronous_local_analyses() below.

Reimplemented from ApplicationInterface.

References ProcessHandleApplicInterface::create_analysis_process(), and ProcessHandleApplicInterface::driver_argument_list().

void init_communicators_checks (int *max_eval_concurrency*) [inline], [protected], [virtual]

No derived interface plug-ins, so perform construct-time checks. However, process init issues as warnings since some contexts (e.g., HierarchSurrModel) initialize more configurations than will be used.

Reimplemented from ApplicationInterface.

References ApplicationInterface::check_multiprocessor_analysis(), and ApplicationInterface::check_multiprocessor_asynchronous().

void set_communicators_checks (int *max_eval_concurrency*) [inline], [protected], [virtual]

Process run-time issues as hard errors.

Reimplemented from ApplicationInterface.

References Dakota::abort_handler(), ApplicationInterface::check_multiprocessor_analysis(), and ApplicationInterface::check_multiprocessor_asynchronous().

pid_t create_evaluation_process (bool *block_flag*) [protected], [virtual]

Manage the input filter, 1 or more analysis programs, and the output filter in blocking or nonblocking mode as governed by block_flag. In the case of a single analysis and no filters, a single fork is performed, while in other cases, an initial fork is reforked multiple times. Called from derived_map() with block_flag == BLOCK and from derived_map_asynch() with block_flag == FALL_THROUGH. Uses create_analysis_process() to spawn individual program components within the function evaluation.

Implements ProcessApplicInterface.

References Dakota::abort_handler(), ProcessHandleApplicInterface::analysis_process_group_id(), ApplicationInterface::analysisServerId, ApplicationInterface::asynchLocalAnalysisConcurrency, ApplicationInterface::asynchLocalAnalysisFlag, ProcessHandleApplicInterface::asynchronous_local_analyses(), ParallelLibrary::barrier_e(), ProcessApplicInterface::commandLineArgs, ProcessHandleApplicInterface::create_analysis_process(), ProcessHandleApplicInterface::driver_argument_list(), ApplicationInterface::eaDedMasterFlag, ApplicationInterface::evalCommRank, ApplicationInterface::evalCommSize, ProcessHandleApplicInterface::evalProcessIdMap, ProcessHandleApplicInterface::evaluation_process_group_id(), ProcessHandleApplicInterface::ifilter_argument_list(), ProcessApplicInterface::iFilterName, ProcessHandleApplicInterface::join_evaluation_process_group(), ApplicationInterface::master_dynamic_schedule_analyses(), ProcessApplicInterface::multipleParamsFiles, ApplicationInterface::numAnalysisDrivers, ApplicationInterface::numAnalysisServers, ProcessHandleApplicInterface::ofilter_argument_list(), ProcessApplicInterface::oFilterName, ApplicationInterface::parallelLib, ProcessApplicInterface::paramsFileName, ProcessApplicInterface::programNames, ProcessApplicInterface::resultsFileName, ProcessHandleApplicInterface::serve_analyses_asynch(), ApplicationInterface::serve_analyses_synch(), ApplicationInterface::suppressOutput, and ProcessApplicInterface::synchronous_local_analyses().

void check_wait (pid_t *pid*, int *status*) [protected]

check the exit status of a forked process and abort if an error code was returned

Check to see if the process terminated abnormally (WIFEXITED(status)==0) or if either execvp or the application returned a status code of -1 (WIFEXITED(status)!=0 && (signed char)WEXITSTATUS(status)==-1). If one of these conditions is detected, output a failure message and abort. Note: the application code should not return a status code of -1 unless an immediate abort of dakota is wanted. If for instance, failure capturing is to be used, the application code should write the word "FAIL" to the appropriate results file and return a status code of 0 through exit().

References Dakota::abort_handler().

Referenced by ForkApplicInterface::create_analysis_process(), SpawnApplicInterface::test_local_analyses_send(), SpawnApplicInterface::test_local_evaluations(), ForkApplicInterface::wait(), SpawnApplicInterface::wait_local_analyses(), and SpawnApplicInterface::wait_local_evaluations().

void asynchronous_local_analyses (int *start,* int *end,* int *step*) [protected]

execute analyses asynchronously on the local processor

Schedule analyses asynchronously on the local processor using a dynamic scheduling approach (start to end in step increments). Concurrency is limited by asynchLocalAnalysisConcurrency. Modeled after ApplicationInterface::asynchronous_local_evaluations(). NOTE: This function should be elevated to ApplicationInterface if and when another derived interface class supports asynchronous local analyses.

References Dakota::abort_handler(), ProcessHandleApplicInterface::analysisProcessIdMap, ApplicationInterface::asynchLocalAnalysisConcurrency, ProcessHandleApplicInterface::create_analysis_process(), ProcessHandleApplicInterface::driver_argument_list(), ApplicationInterface::numAnalysisDrivers, and ProcessHandleApplicInterface::wait_local_analyses().

Referenced by ProcessHandleApplicInterface::create_evaluation_process().

void serve_analyses_asynch ( ) [protected]

serve the analysis scheduler and execute analysis jobs asynchronously

This code runs multiple asynch analyses on each server. It is modeled after ApplicationInterface::serve_evaluations_asynch(). NOTE: This fn should be elevated to ApplicationInterface if and when another derived interface class supports hybrid analysis parallelism.

References Dakota::abort_handler(), ProcessHandleApplicInterface::analysisProcessIdMap, ApplicationInterface::asynchLocalAnalysisConcurrency, ProcessHandleApplicInterface::create_analysis_process(), ProcessHandleApplicInterface::driver_argument_list(), ParallelLibrary::irecv_ea(), ApplicationInterface::numAnalysisDrivers, ApplicationInterface::parallelLib, ParallelLibrary::recv_ea(), ParallelLibrary::test(), and ProcessHandleApplicInterface::test_local_analyses_send().

Referenced by ProcessHandleApplicInterface::create_evaluation_process().

void create_command_arguments (boost::shared_array< const char * > & *av,* StringArray & *driver_and_args*) [protected]

parse argList into argument array av suitable for passing to execvp, appending parameters and results filenames if requested by commandLineArgs

This function will split the analysis command in argList[0] based on whitespace, but preserve spaces within quoted strings, such that quoted strings can be passed as single command arguments. NOTE: This function allocates memory in av that might be implicitly freed when the child exits (control never returns to caller). driver_and_args needs to be a return argument because av will contain pointers into its c_str()'s when done.

References ProcessHandleApplicInterface::argList, ProcessApplicInterface::commandLineArgs, and WorkdirHelper::tokenize_driver().

Referenced by SpawnApplicInterface::create_analysis_process(), and ForkApplicInterface::create_analysis_process().

The documentation for this class was generated from the following files:

- ProcessHandleApplicInterface.hpp
- ProcessHandleApplicInterface.cpp

13.129 ProgramOptions Class Reference

ProgramOptions stores options whether from the CLH or from library user; initially valid only on worldRank = 0, but then broadcast in ParallelLibrary::push_output_tag()

Public Member Functions

- ProgramOptions ()

  default constructor (needed for default environment ctors and could be used by library clients to late update data)
- ProgramOptions (int world_rank)

  constructor that accepts world rank to help with I/O control; allows default constructed ProgramOptions to get rank in library mode
- ProgramOptions (int argc, char *argv[ ], int world_rank)

  standard constructor that uses a CommandLineHandler to parse user options
- const String & input_file () const

  Dakota input file base name (no tag)
- const String & input_string () const

  alternate Dakota input string literal
- bool echo_input () const

  is input echo specified?
- const String & parser_options () const

  (deprecated) NIDR parser options
- String output_file () const

  output (user-provided or default) file base name (no tag)
- const String & error_file () const

  error file base name (no tag)
- const String & exit_mode () const

  behavior of abort_handler (throw or exit)
- const String & read_restart_file () const

  restart file base name (no tag)
- size_t stop_restart_evals () const

  eval ID at which to stop reading restart
- String write_restart_file () const

  write retart (user-provided or default) file base name (no tag)
- bool help () const

  is help mode active?
- bool version () const

  is version mode active?
- bool check () const

is check mode active?

- bool pre_run () const

  is pre-run mode active?
- bool run () const

  is run mode active?
- bool post_run () const

  is post-run mode active?
- bool user_modes () const

  are any non-default, user-specified run modes active?
- const String & pre_run_input () const

  filename for pre-run input
- const String & pre_run_output () const

  filename for pre-run output
- const String & run_input () const

  filename for run input
- const String & run_output () const

  filename for run output
- const String & post_run_input () const

  filename for post-run input
- const String & post_run_output () const

  filename for post-run output
- bool proceed_to_instantiate () const

  whether steps beyond help/version are requested (instantiation required)
- bool proceed_to_run () const

  Whether steps beyond check are requested.
- bool user_stdout_redirect () const

  whether the user/client code requested a redirect of stdout
- bool user_stderr_redirect () const

  whether the user/client code requested a redirect of stderr
- void world_rank (int world_rank)

  set the world rank to govern early conditional output
- void input_file (const String &in_file)

  set Dakota input file base name (no tag)
- void input_string (const String &in_string)

  set alternate Dakota input string literal
- void echo_input (bool echo_flag)

  set whether to echo input to output
- void exit_mode (const String &mode)

  set behavior for abort_handler
- void output_file (const String &out_file)

  set base file name for Dakota output
- void error_file (const String &err_file)

  set base file name for Dakota errors

- void read_restart_file (const String &read_rst)

  set base file name for restart file from which to read
- void stop_restart_evals (size_t stop_rst)

  set eval ID at which to stop reading restart
- void write_restart_file (const String &write_rst)

  set base file name for restart file to write
- void help (bool help_flag)

  set true to print help information and exit
- void version (bool version_flag)

  set true to print version information and exit
- void check (bool check_flag)

  set true to check input and instantiate objects, then exit
- void pre_run (bool pre_run_flag)

  set to enable/disable pre-run phase
- void run (bool run_flag)

  set to enable/disable run phase
- void post_run (bool post_run_flag)

  set to enable/disable post-run phase
- void pre_run_input (const String &pre_run_in)

  Specify the pre-run phase input filename.
- void pre_run_output (const String &pre_run_out)

  Specify the pre-run phase output filename.
- void run_input (const String &run_in)

  Specify the run phase input filename.
- void run_output (const String &run_out)

  Specify the run phase output filename.
- void post_run_input (const String &post_run_in)

  Specify the post-run phase input filename.
- void post_run_output (const String &post_run_out)

  Specify the post-run phase output filename.
- void parse (const ProblemDescDB &problem_db)

  Extract environment options from ProblemDescDB.
- void read (MPIUnpackBuffer &s)

  helper function for reading some class data from MPI buffer
- void write (MPIPackBuffer &s) const

  helper function for writing some class data to MPI buffer

Private Member Functions

- void parse_environment_options ()

  any environment variables affecting global behavior get read here
- void manage_run_modes (const CommandLineHandler &clh)

  retrieve run mode options from the CLH
- void split_filenames (const char *filenames, std::string &input_filename, std::string &output_filename)

  manage pre/run/post filenames
- void validate ()

  verify consistency of user settings (helpful for library mode especially)
- void validate_run_modes ()

  validate user run modes and set userModesFlag
- void set_option (const ProblemDescDB &problem_db, const String &db_name, String &data_member)

  retrieve environment.<db_name> from the problem db and update data_member, warning if needed

Private Attributes

- int worldRank

  cache the world rank to help with conditional output
- String inputFile

  Dakota input file name, e.g., "dakota.in".
- String inputString

  alternate input means for library clients: input string (mutually exclusive with input file)
- bool echoInput

  whether to echo client's input file at parse
- String parserOptions

  Deprecated option for NIDR parser options.
- String exitMode

  Abort or throw on error.
- String outputFile

  Dakota output base file name, e.g., "dakota.out".
- String errorFile

  Dakota error base file name, e.g., "dakota.err".
- String readRestartFile

  e.g., "dakota.old.rst"
- size_t stopRestartEvals

  eval number at which to stop restart read
- String writeRestartFile

  e.g., "dakota.new.rst"
- bool helpFlag

  whether to print help message and exit
- bool versionFlag

  whether to print version message and exit
- bool checkFlag

flags invocation with command line option -check

- bool preRunFlag

  flags invocation with command line option -pre_run

- bool runFlag

  flags invocation with command line option -run

- bool postRunFlag

  flags invocation with command line option -post_run

- bool userModesFlag

  whether any user run modes are active

- String preRunInput

  filename for pre_run input

- String preRunOutput

  filename for pre_run output

- String runInput

  filename for run input

- String runOutput

  filename for run output

- String postRunInput

  filename for post_run input

- String postRunOutput

  filename for post_run output

13.129.1 Detailed Description

ProgramOptions stores options whether from the CLH or from library user; initially valid only on worldRank = 0, but then broadcast in ParallelLibrary::push_output_tag()

13.129.2 Member Function Documentation

void split_filenames (const char * *filenames*, std::string & *input_filename*, std::string & *output_filename*) `[private]`

manage pre/run/post filenames

Tokenize colon-delimited input and output filenames, returns unchanged strings if tokens not found.

Referenced by ProgramOptions::manage_run_modes().

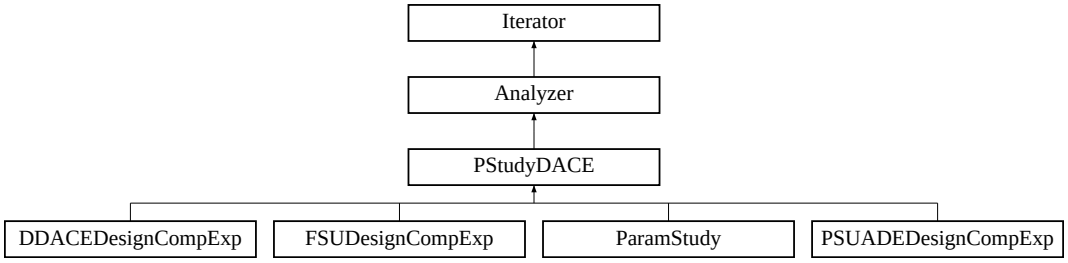
The documentation for this class was generated from the following files:

- ProgramOptions.hpp
- ProgramOptions.cpp

13.130 PStudyDACE Class Reference

Base class for managing common aspects of parameter studies and design of experiments methods.

Inheritance diagram for PStudyDACE:



Protected Member Functions

- PStudyDACE (ProblemDescDB &problem_db, Model &model)

  constructor
- PStudyDACE (unsigned short method_name, Model &model)

  alternate constructor for instantiations "on the fly"
- ~PStudyDACE ()

  destructor
- void core_run ()

  core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post
- void print_results (std::ostream &s)

  print the final iterator results
- virtual void extract_trends ()=0

  Mapping of the core_run() virtual function for the PStudy/DACE branch.
- void volumetric_quality (int ndim, int num_samples, double *sample_points)

  Calculation of volumetric quality measures.

Protected Attributes

- SensAnalysisGlobal pStudyDACESensGlobal

  initialize statistical post processing
- bool volQualityFlag

  flag which specifies evaluation of volumetric quality measures
- bool varBasedDecompFlag

  flag which specifies calculating variance based decomposition sensitivity analysis metrics

Private Attributes

- double chiMeas

  quality measure
- double dMeas

  quality measure
- double hMeas

  quality measure
- double tauMeas

  quality measure

Additional Inherited Members

13.130.1 Detailed Description

Base class for managing common aspects of parameter studies and design of experiments methods.

The PStudyDACE base class manages common data and functions, such as those involving the best solutions located during the parameter set evaluations or the printing of final results.

13.130.2 Member Function Documentation

void core_run ( ) [inline], [protected], [virtual]

core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post

Virtual run function for the iterator class hierarchy. All derived classes need to redefine it.

Reimplemented from Iterator.

References Analyzer::bestVarsRespMap, and PStudyDACE::extract_trends().

void print_results (std::ostream & *s*) [protected], [virtual]

print the final iterator results

This virtual function provides additional iterator-specific final results outputs beyond the function evaluation summary printed in finalize_run().

Reimplemented from Analyzer.

References PStudyDACE::chiMeas, Analyzer::compactMode, Model::continuous_variable_labels(), SensAnalysisGlobal::correlations_computed(), Model::discrete_int_variable_labels(), Model::discrete_real_variable_labels(), Model::discrete_string_variable_labels(), PStudyDACE::dMeas, PStudyDACE::hMeas, Iterator::iteratedModel, Analyzer::numLSqTerms, Analyzer::numObjFns, SensAnalysisGlobal::print_correlations(), Analyzer::print_results(), Analyzer::print_sobol_indices(), PStudyDACE::pStudyDACESensGlobal, Model::response_labels(), PStudyDACE::tauMeas, PStudyDACE::varBasedDecompFlag, and PStudyDACE::volQualityFlag.

void volumetric_quality (int *ndim*, int *num_samples*, double * *sample_points*) [protected]

Calculation of volumetric quality measures.

Calculation of volumetric quality measures developed by FSU.

References PStudyDACE::chiMeas, PStudyDACE::dMeas, PStudyDACE::hMeas, and PStudyDACE::tauMeas.

Referenced by FSUDesignCompExp::get_parameter_sets(), and DDACEDesignCompExp::get_parameter_sets().

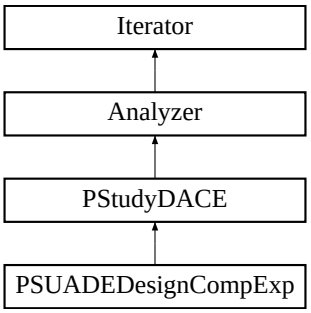
The documentation for this class was generated from the following files:

- DakotaPStudyDACE.hpp
- DakotaPStudyDACE.cpp

13.131 PSUADEDesignCompExp Class Reference

Wrapper class for the PSUADE library.

Inheritance diagram for PSUADEDesignCompExp:



Public Member Functions

- PSUADEDesignCompExp (ProblemDescDB &problem_db, Model &model)

  primary constructor for building a standard DACE iterator

- ~PSUADEDesignCompExp ()

  destructor

- void pre_run ()

  pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

- void post_input ()

  read tabular data for post-run mode

- void extract_trends ()

  Mapping of the core_run() virtual function for the PStudy/DACE branch.

- void post_run (std::ostream &s)

  post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way

- int num_samples () const
- void sampling_reset (int min_samples, bool all_data_flag, bool stats_flag)

  reset sampling iterator to use at least min_samples

- unsigned short sampling_scheme () const

  return sampling name

- void vary_pattern (bool pattern_flag)

  sets varyPattern in derived classes that support it

- void get_parameter_sets (Model &model)

  Returns one block of samples (ndim ∗ num_samples)

Private Member Functions

- void enforce_input_rules ()

  enforce sanity checks/modifications for the user input specification

Private Attributes

- int samplesSpec

  initial specification of number of samples
- int numSamples

  current number of samples to be evaluated
- const UShortArray & varPartitionsSpec

  number of partitions in each variable direction
- int numPartitions

  number of partitions to pass to PSUADE (levels = partitions + 1)
- bool allDataFlag

  flag which triggers the update of allVars/allResponses for use by Iterator::all_variables() and Iterator::all_responses()
- size_t numDACERuns

  counter for number of executions for this object
- bool varyPattern

  flag for generating a sequence of seed values within multiple get_parameter_sets() calls so that the sample sets are not repeated, but are still repeatable
- const int seedSpec

  the user seed specification for the random number generator (allows repeatable results)
- int randomSeed

  current seed for the random number generator

Additional Inherited Members

13.131.1 Detailed Description

Wrapper class for the PSUADE library.

The PSUADEDesignCompExp class provides a wrapper for PSUADE, a C++ design of experiments library from Lawrence Livermore National Laboratory. Currently this class only includes the PSUADE Morris One-at-a-time (MOAT) method to uniformly sample the parameter space spanned by the active bounds of the current Model. It returns all generated samples and their corresponding responses as well as the best sample found.

13.131.2 Constructor & Destructor Documentation

PSUADEDesignCompExp (ProblemDescDB & *problem_db*, Model & *model*)

primary constructor for building a standard DACE iterator

This constructor is called for a standard iterator built with data from probDescDB.

References Dakota::abort_handler(), Iterator::maxEvalConcurrency, Iterator::methodName, and PSUADEDesignCompExp::numSamples.

13.131.3 Member Function Documentation

void pre_run ( ) `[virtual]`

pre-run portion of run (optional); re-implemented by Iterators which can generate all Variables (parameter sets) a priori

pre-run phase, which a derived iterator may optionally reimplement; when not present, pre-run is likely integrated into the derived run function. This is a virtual function; when re-implementing, a derived class must call its nearest parent's pre_run(), if implemented, typically *before* performing its own implementation steps.

Reimplemented from Iterator.

References PSUADEDesignCompExp::get_parameter_sets(), and Iterator::iteratedModel.

void post_run (std::ostream & *s*) `[virtual]`

post-run portion of run (optional); verbose to print results; re-implemented by Iterators that can read all Variables/-Responses and perform final analysis phase in a standalone way

Post-run phase, which a derived iterator may optionally reimplement; when not present, post-run is likely integrated into run. This is a virtual function; when re-implementing, a derived class must call its nearest parent's post_run(), typically *after* performing its own implementation steps.

Reimplemented from Analyzer.

References Dakota::abort_handler(), Analyzer::allResponses, Analyzer::allSamples, Model::continuous_lower-_bounds(), Model::continuous_upper_bounds(), Iterator::iteratedModel, Analyzer::numContinuousVars, Analyzer-::numFunctions, PSUADEDesignCompExp::numSamples, and Analyzer::post_run().

int num_samples ( ) const `[inline], [virtual]`

Return current number of evaluation points. Since the calculation of samples, collocation points, etc. might be costly, provide a default implementation here that backs out from the maxEvalConcurrency.

Reimplemented from Analyzer.

References PSUADEDesignCompExp::numSamples.

void enforce_input_rules ( ) `[private]`

enforce sanity checks/modifications for the user input specification

Users may input a variety of quantities, but this function must enforce any restrictions imposed by the sampling algorithms.

References Dakota::abort_handler(), Iterator::methodName, Analyzer::numContinuousVars, PSUADEDesign-CompExp::numPartitions, PSUADEDesignCompExp::numSamples, and PSUADEDesignCompExp::varPartitions-Spec.

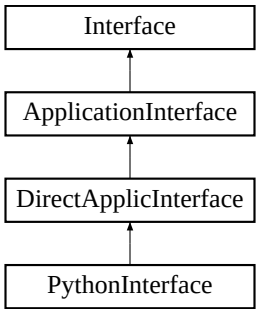
Referenced by PSUADEDesignCompExp::get_parameter_sets().

The documentation for this class was generated from the following files:

- PSUADEDesignCompExp.hpp
- PSUADEDesignCompExp.cpp

13.132 PythonInterface Class Reference

Inheritance diagram for PythonInterface:



Public Member Functions

- PythonInterface (const ProblemDescDB &problem_db)

  constructor
- ~PythonInterface ()

  destructor

Protected Member Functions

- virtual int derived_map_ac (const String &ac_name)

  execute an analysis code portion of a direct evaluation invocation
- int python_run (const String &ac_name)

  direct interface to Python via API, BMA 07/02/07
- template<class ArrayT , class Size >
  bool python_convert_int (const ArrayT &src, Size size, PyObject **dst)

  convert arrays of integer types to Python list or numpy array
- bool python_convert (const RealVector &src, PyObject **dst)

  convert RealVector to Python list or numpy array
- bool python_convert (const RealVector &c_src, const IntVector &di_src, const RealVector &dr_src, PyObject **dst)

  convert RealVector + IntVector + RealVector to Python mixed list or numpy double array
- template<class StringArrayT >
  bool python_convert_strlist (const StringArrayT &src, PyObject **dst)

  convert labels
- bool python_convert (const StringMultiArray &c_src, const StringMultiArray &di_src, const StringMultiArray &dr_src, PyObject **dst)

  convert all labels to single list
- bool python_convert (PyObject *pyv, RealVector &rv, const int &dim)

  convert python [list of int or float] or [numpy array of double] to RealVector (for fns)
- bool python_convert (PyObject *pyv, double *rv, const int &dim)

  convert python [list of int or float] or [numpy array of double] to double[], for use as helper in converting gradients
- bool python_convert (PyObject *pym, RealMatrix &rm)

  convert python [list of lists of int or float] or [numpy array of dbl] to RealMatrix (for gradients)
- bool python_convert (PyObject *pym, RealSymMatrix &rm)

convert python [list of lists of int or float] or [numpy array of dbl] to RealMatrix (used as helper in Hessian conversion)

- bool python_convert (PyObject *pyma, RealSymMatrixArray &rma)

  convert python [list of lists of lists of int or float] or [numpy array of double] to RealSymMatrixArray (for Hessians)

Protected Attributes

- bool userNumpyFlag

  whether the user requested numpy data structures in the input file
- bool ownPython

  true if this class created the interpreter instance

13.132.1 Detailed Description

Specialization of DirectApplicInterface to link to Python analysis drivers. Includes convenience functions to map data to/from Python

13.132.2 Member Function Documentation

int derived_map_ac (const String & *ac_name*) `[protected], [virtual]`

execute an analysis code portion of a direct evaluation invocation

Python specialization of derived analysis components.

Reimplemented from DirectApplicInterface.

References ApplicationInterface::analysisServerId, and PythonInterface::python_run().

bool python_convert_int (const ArrayT & *src*, Size *sz*, PyObject ** *dst*) `[protected]`

convert arrays of integer types to Python list or numpy array

convert all integer array types including IntVector, ShortArray, and SizetArray to Python list of ints or numpy array of ints

References PythonInterface::userNumpyFlag.

Referenced by PythonInterface::python_run().

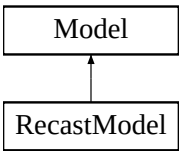
The documentation for this class was generated from the following files:

- PythonInterface.hpp
- PythonInterface.cpp

13.133 RecastModel Class Reference

Derived model class which provides a thin wrapper around a sub-model in order to recast the form of its inputs and/or outputs.

Inheritance diagram for RecastModel:



Public Member Functions

- RecastModel (const Model &sub_model, const Sizet2DArray &vars_map_indices, const SizetArray &vars_comps_total, const BitArray &all_relax_di, const BitArray &all_relax_dr, bool nonlinear_vars_mapping, void(∗variables_map)(const Variables &recast_vars, Variables &sub_model_vars), void(∗set_map)(const Variables &recast_vars, const ActiveSet &recast_set, ActiveSet &sub_model_set), const Sizet2DArray &primary_resp_map_indices, const Sizet2DArray &secondary_resp_map_indices, size_t recast_secondary_offset, const BoolDequeArray &nonlinear_resp_mapping, void(∗primary_resp_map)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response), void(∗secondary_resp_map)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response))

  standard constructor
- RecastModel (const Model &sub_model, const SizetArray &vars_comps_totals, const BitArray &all_relax_di, const BitArray &all_relax_dr, size_t num_recast_primary_fns, size_t num_recast_secondary_fns, size_t recast_secondary_offset)

  alternate constructor
- ∼RecastModel ()

  destructor
- void initialize (const Sizet2DArray &vars_map_indices, bool nonlinear_vars_mapping, void(∗variables_map)(const Variables &recast_vars, Variables &sub_model_vars), void(∗set_map)(const Variables &recast_vars, const ActiveSet &recast_set, ActiveSet &sub_model_set), const Sizet2DArray &primary_resp_map_indices, const Sizet2DArray &secondary_resp_map_indices, const BoolDequeArray &nonlinear_resp_mapping, void(∗primary_resp_map)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response), void(∗secondary_resp_map)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response))

  completes initialization of the RecastModel after alternate construction
- void inverse_mappings (void(∗inv_vars_map)(const Variables &sub_model_vars, Variables &recast_vars), void(∗inv_set_map)(const Variables &sub_model_vars, const ActiveSet &sub_model_set, ActiveSet &recast_set), void(∗inv_pri_resp_map)(const Variables &recast_vars, const Variables &sub_model_vars, const Response &recast_resp, Response &sub_model_resp), void(∗inv_sec_resp_map)(const Variables &recast_vars, const Variables &sub_model_vars, const Response &recast_resp, Response &sub_model_resp))

  provide optional inverse mappings
- void transform_variables (const Variables &recast_vars, Variables &sub_model_vars)

  perform transformation of Variables (recast –> sub-model)
- void transform_set (const Variables &recast_vars, const ActiveSet &recast_set, ActiveSet &sub_model_set)

  into sub_model_set for use with subModel.
- void transform_response (const Variables &recast_vars, const Variables &sub_model_vars, const Response &sub_model_resp, Response &recast_resp)

  perform transformation of Response (sub-model –> recast)
- void inverse_transform_variables (const Variables &sub_model_vars, Variables &recast_vars)

  perform inverse transformation of Variables (sub-model –> recast)
- void inverse_transform_set (const Variables &sub_model_vars, const ActiveSet &sub_model_set, ActiveSet &recast_set)

  into sub_model_set for use with subModel.
- void inverse_transform_response (const Variables &sub_model_vars, const Variables &recast_vars, const Response &recast_resp, Response &sub_model_resp)

  perform inverse transformation of Response (recast –> sub-model)

- void submodel_supports_derivative_estimation (bool sed_flag)

  override the submodel's derivative estimation behavior

Protected Member Functions

- void derived_compute_response (const ActiveSet &set)

  portion of compute_response() specific to RecastModel (forward to subModel.compute_response())
- void derived_asynch_compute_response (const ActiveSet &set)

  portion of asynch_compute_response() specific to RecastModel (forward to subModel.asynch_compute_response())
- const IntResponseMap & derived_synchronize ()

  portion of synchronize() specific to RecastModel (forward to subModel.synchronize())
- const IntResponseMap & derived_synchronize_nowait ()

  portion of synchronize_nowait() specific to RecastModel (forward to subModel.synchronize_nowait())
- Iterator & subordinate_iterator ()

  return sub-iterator, if present, within subModel
- Model & subordinate_model ()

  return subModel
- Model & surrogate_model ()

  return surrogate model, if present, within subModel
- Model & truth_model ()

  return truth model, if present, within subModel
- void derived_subordinate_models (ModelList &ml, bool recurse_flag)

  add subModel to list and recurse into subModel
- void update_from_subordinate_model (bool recurse_flag=true)

  pass request to subModel if recursing and then update from it
- Interface & derived_interface ()

  return subModel interface
- void primary_response_fn_weights (const RealVector &wts, bool recurse_flag=true)

  set the relative weightings for multiple objective functions or least squares terms and optionally recurses into subModel
- void surrogate_function_indices (const IntSet &surr_fn_indices)

  update the subModel's surrogate response function indices (DataFitSurrModel::surrogateFnIndices)
- void surrogate_response_mode (short mode)

  update the subModel's surrogate response mode (SurrogateModel::responseMode)
- void build_approximation ()

  builds the subModel approximation
- bool build_approximation (const Variables &vars, const IntResponsePair &response_pr)

  builds the subModel approximation
- void update_approximation (bool rebuild_flag)

  replaces data in the subModel approximation
- void update_approximation (const Variables &vars, const IntResponsePair &response_pr, bool rebuild_flag)

  replaces data in the subModel approximation
- void update_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map, bool rebuild_flag)

replaces data in the subModel approximation

- void append_approximation (bool rebuild_flag)

  appends data to the subModel approximation
- void append_approximation (const Variables &vars, const IntResponsePair &response_pr, bool rebuild_flag)

  appends data to the subModel approximation
- void append_approximation (const VariablesArray &vars_array, const IntResponseMap &resp_map, bool rebuild_flag)

  appends data to the subModel approximation
- void pop_approximation (bool save_surr_data, bool rebuild_flag=false)

  remove the previous data set addition to a surrogate (e.g., due to a previous append_approximation() call); flag manages storing of surrogate data for use in a subsequent restore_approximation()
- void restore_approximation ()

  restore a previous approximation data state within a surrogate
- bool restore_available ()

  query for whether a trial increment is restorable within a surrogate
- void finalize_approximation ()

  finalize an approximation by applying all previous trial increments
- void store_approximation ()

  move the current approximation into storage for later combination
- void combine_approximation (short corr_type)

  combine the current approximation with one previously stored
- std::vector< Approximation > & approximations ()

  retrieve the set of Approximations from the subModel
- const RealVectorArray & approximation_coefficients (bool normalized=false)

  retrieve the approximation coefficients from the subModel
- void approximation_coefficients (const RealVectorArray &approx_coeffs, bool normalized=false)

  set the approximation coefficients within the subModel
- const RealVector & approximation_variances (const Variables &vars)

  retrieve the approximation variances from the subModel
- const Pecos::SurrogateData & approximation_data (size_t index)

  retrieve the approximation data from the subModel
- void component_parallel_mode (short mode)

  RecastModel only supports parallelism in subModel, so this virtual function redefinition is simply a sanity check.
- size_t mi_parallel_level_index () const

  return subModel's MI parallel level index
- short local_eval_synchronization ()

  return subModel local synchronization setting
- int local_eval_concurrency ()

  return subModel local evaluation concurrency
- bool derived_master_overload () const

  flag which prevents overloading the master with a multiprocessor evaluation (request forwarded to subModel)
- void derived_init_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set up RecastModel for parallel operations (request forwarded to subModel)

- void derived_init_serial ()

  set up RecastModel for serial operations (request forwarded to subModel).
- void derived_set_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set active parallel configuration within subModel
- void derived_free_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  deallocate communicator partitions for the RecastModel (request forwarded to subModel)
- void serve_run (ParLevLIter pl_iter, int max_eval_concurrency)

  Service subModel job requests received from the master. Completes when a termination message is received from stop_servers().
- void stop_servers ()

  executed by the master to terminate subModel server operations when RecastModel iteration is complete.
- void inactive_view (short view, bool recurse_flag=true)

  update the Model's inactive view based on higher level (nested) context and optionally recurse into subModel
- const String & interface_id () const

  return the subModel interface identifier
- int evaluation_id () const

  return the current evaluation id for the RecastModel (request forwarded to subModel)
- void set_evaluation_reference ()

  set the evaluation counter reference points for the RecastModel (request forwarded to subModel)
- void fine_grained_evaluation_counters ()

  request fine-grained evaluation reporting within subModel
- void print_evaluation_summary (std::ostream &s, bool minimal_header=false, bool relative_count=true) const

  print the evaluation summary for the RecastModel (request forwarded to subModel)
- void eval_tag_prefix (const String &eval_id_str)

  set the hierarchical eval ID tag prefix

Private Member Functions

- void initialize_data_from_submodel ()

  code shared among constructors to initialize base class data from submodel
- void update_from_sub_model ()

  update current variables/labels/bounds/targets from subModel

Private Attributes

- Model subModel

  the sub-model underlying the function pointers
- Sizet2DArray varsMapIndices

  For each subModel variable, identifies the indices of the recast variables used to define it (maps RecastModel variables to subModel variables; data is packed with only the variable indices employed rather than a sparsely filled N_sm x N_r matrix)
- bool nonlinearVarsMapping

boolean set to true if the variables mapping involves a nonlinear transformation. Used in transform_set() to manage the requirement for gradients within the Hessian transformations. This does not require a BoolDeque for each individual variable, since response gradients and Hessians are managed per function, not per variable.

- bool respMapping

  set to true if non-NULL primaryRespMapping or secondaryRespMapping are supplied

- Sizet2DArray primaryRespMapIndices

  For each recast primary function, identifies the indices of the subModel functions used to define it (maps subModel response to RecastModel Response).

- Sizet2DArray secondaryRespMapIndices

  For each recast secondary function, identifies the indices of the subModel functions used to define it (maps subModel response to RecastModel response).

- BoolDequeArray nonlinearRespMapping

  array of BoolDeques, one for each recast response function. Each BoolDeque defines which subModel response functions contribute to the recast function using a nonlinear mapping. Used in transform_set() to augment the subModel function value/gradient requirements.

- IntActiveSetMap recastSetMap

  map of recast active set passed to derived_asynch_compute_response(). Needed for currentResponse update in synchronization routines.

- IntVariablesMap recastVarsMap

  map of recast variables used by derived_asynch_compute_response(). Needed for primaryRespMapping() and secondaryRespMapping() in synchronization routines.

- IntVariablesMap subModelVarsMap

  map of subModel variables used by derived_asynch_compute_response(). Needed for primaryRespMapping() and secondaryRespMapping() in synchronization routines.

- IntResponseMap recastResponseMap

  map of recast responses used by RecastModel::derived_synchronize() and RecastModel::derived_synchronize_nowait()

- void(∗ variablesMapping)(const Variables &recast_vars, Variables &sub_model_vars)

  holds pointer for variables mapping function passed in ctor/initialize

- void(∗ setMapping)(const Variables &recast_vars, const ActiveSet &recast_set, ActiveSet &sub_model_set)

  holds pointer for set mapping function passed in ctor/initialize

- void(∗ primaryRespMapping)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  holds pointer for primary response mapping function passed in ctor/initialize

- void(∗ secondaryRespMapping)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response)

  holds pointer for secondary response mapping function passed in ctor/initialize

- void(∗ invVarsMapping)(const Variables &sub_model_vars, Variables &recast_vars)

  holds pointer for optional inverse variables mapping function passed in inverse_mappings()

- void(∗ invSetMapping)(const Variables &sub_model_vars, const ActiveSet &sub_model_set, ActiveSet &recast_set)

  holds pointer for optional inverse set mapping function passed in inverse_mappings()

- void(∗ invPriRespMapping)(const Variables &recast_vars, const Variables &sub_model_vars, const Response &recast_resp, Response &sub_model_resp)

  holds pointer for optional inverse primary response mapping function passed in inverse_mappings()

- void(∗ invSecRespMapping)(const Variables &recast_vars, const Variables &sub_model_vars, const Response &recast_resp, Response &sub_model_resp)

  holds pointer for optional inverse secondary response mapping function passed in inverse_mappings()

Additional Inherited Members

13.133.1 Detailed Description

Derived model class which provides a thin wrapper around a sub-model in order to recast the form of its inputs and/or outputs.

The RecastModel class uses function pointers to allow recasting of the subModel input/output into new problem forms. This is currently used to recast SBO approximate subproblems, but can be used for multiobjective, input/output scaling, and other problem modifications in the future.

13.133.2 Constructor & Destructor Documentation

RecastModel (const Model & *sub_model,* const Sizet2DArray & *vars_map_indices,* const SizetArray & *vars_comps_totals,* const BitArray & *all_relax_di,* const BitArray & *all_relax_dr,* bool *nonlinear_vars_mapping,* void(∗)(const Variables &recast_vars, Variables &sub_model_vars) *variables_map,* void(∗)(const Variables &recast_vars, const ActiveSet &recast_set, ActiveSet &sub_model_set) *set_map,* const Sizet2DArray & *primary_resp_map_indices,* const Sizet2DArray & *secondary_resp_map_indices,* size_t *recast_secondary_offset,* const BoolDequeArray & *nonlinear_resp_mapping,* void(∗)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response) *primary_resp_map,* void(∗)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response) *secondary_resp_map*)

standard constructor

Default recast model constructor. Requires full definition of the transformation. Parameter vars_comps_totals indicates the number of each type of variable {4 types} x {3 domains} in the recast variable space. Note: recast_secondary_offset is the start index for equality constraints, typically num nonlinear ineq constraints.

References Dakota::abort_handler(), SharedVariablesData::all_relaxed_discrete_int(), SharedVariablesData::all_relaxed_discrete_real(), SharedVariablesData::components_totals(), Constraints::copy(), Response::copy(), Variables::copy(), Model::current_response(), Model::current_variables(), Model::currentResponse, Model::currentVariables, Variables::cv(), Response::function_gradients(), Response::function_hessians(), RecastModel::initialize_data_from_submodel(), Model::modelType, RecastModel::nonlinearRespMapping, Response::num_functions(), Model::num_functions(), Constraints::num_linear_eq_constraints(), Constraints::num_linear_ineq_constraints(), Constraints::num_nonlinear_eq_constraints(), Constraints::num_nonlinear_ineq_constraints(), Model::numDerivVars, Model::numFns, RecastModel::primaryRespMapIndices, RecastModel::primaryRespMapping, Constraints::reshape(), Response::reshape(), RecastModel::respMapping, RecastModel::secondaryRespMapIndices, RecastModel::secondaryRespMapping, Variables::shared_data(), RecastModel::subModel, Model::supportsEstimDerivs, Model::user_defined_constraints(), Model::userDefinedConstraints, RecastModel::variablesMapping, and Variables::view().

RecastModel (const Model & *sub_model,* const SizetArray & *vars_comps_totals,* const BitArray & *all_relax_di,* const BitArray & *all_relax_dr,* size_t *num_recast_primary_fns,* size_t *num_recast_secondary_fns,* size_t *recast_secondary_offset*)

alternate constructor

This alternate constructor defers initialization of the function pointers until a separate call to initialize(), and accepts the minimum information needed to construct currentVariables, currentResponse, and userDefined-

Constraints. The resulting model is sufficiently complete for passing to an Iterator. Parameter vars_comps_totals indicates the number of each type of variable {4 types} x {3 domains} in the recast variable space. Note: recast_secondary_offset is the start index for equality constraints, typically num nonlinear ineq constraints.

References SharedVariablesData::all_relaxed_discrete_int(), SharedVariablesData::all_relaxed_discrete_real(), SharedVariablesData::components_totals(), Constraints::copy(), Response::copy(), Variables::copy(), Model::current_response(), Model::current_variables(), Model::currentResponse, Model::currentVariables, Variables::cv(), Response::function_gradients(), Response::function_hessians(), RecastModel::initialize_data_from_submodel(), Model::modelType, Model::num_functions(), Constraints::num_linear_eq_constraints(), Constraints::num_linear_ineq_constraints(), Constraints::num_nonlinear_eq_constraints(), Constraints::num_nonlinear_ineq_constraints(), Model::numDerivVars, Model::numFns, Constraints::reshape(), Response::reshape(), Variables::shared_data(), RecastModel::subModel, Model::supportsEstimDerivs, Model::user_defined_constraints(), Model::userDefinedConstraints, RecastModel::variablesMapping, and Variables::view().

13.133.3 Member Function Documentation

void initialize (const Sizet2DArray & *vars_map_indices,* bool *nonlinear_vars_mapping,* void(∗)(const Variables &recast_vars, Variables &sub_model_vars) *variables_map,* void(∗)(const Variables &recast_vars, const ActiveSet &recast_set, ActiveSet &sub_model_set) *set_map,* const Sizet2DArray & *primary_resp_map_indices,* const Sizet2DArray & *secondary_resp_map_indices,* const BoolDequeArray & *nonlinear_resp_mapping,* void(∗)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response) *primary_resp_map,* void(∗)(const Variables &sub_model_vars, const Variables &recast_vars, const Response &sub_model_response, Response &recast_response) *secondary_resp_map*)

completes initialization of the RecastModel after alternate construction

This function is used for late initialization of the recasting functions. It is used in concert with the alternate constructor.

References Dakota::abort_handler(), RecastModel::nonlinearRespMapping, RecastModel::nonlinearVarsMapping, RecastModel::primaryRespMapIndices, RecastModel::primaryRespMapping, RecastModel::respMapping, RecastModel::secondaryRespMapIndices, RecastModel::secondaryRespMapping, RecastModel::setMapping, RecastModel::variablesMapping, and RecastModel::varsMapIndices.

Referenced by EffGlobalMinimizer::minimize_surrogates_on_model(), NonDLocalReliability::mpp_search(), NonDGlobalReliability::optimize_gaussian_process(), NonDLocalInterval::quantify_uncertainty(), NonDGlobalInterval::quantify_uncertainty(), and Minimizer::scale_model().

void derived_compute_response (const ActiveSet & *set*) `[protected], [virtual]`

portion of compute_response() specific to RecastModel (forward to subModel.compute_response())

The RecastModel is evaluated by an Iterator for a recast problem formulation. Therefore, the currentVariables, incoming active set, and output currentResponse all correspond to the recast inputs/outputs.

Reimplemented from Model.

References Response::active_set(), Model::compute_response(), Model::current_response(), Model::current_variables(), Model::currentResponse, Model::currentVariables, RecastModel::respMapping, RecastModel::subModel, RecastModel::transform_response(), RecastModel::transform_set(), RecastModel::transform_variables(), and Response::update().

void eval_tag_prefix (const String & *eval_id_str*) `[inline], [protected], [virtual]`

set the hierarchical eval ID tag prefix

RecastModel just forwards any tags to its subModel

Reimplemented from Model.

References Model::eval_tag_prefix(), and RecastModel::subModel.

void update_from_sub_model ( ) [private]

update current variables/labels/bounds/targets from subModel

Update inactive values and labels in currentVariables and inactive bound constraints in userDefinedConstraints from variables and constraints data within subModel.

References Model::aleatDistParams, Model::aleatory_distribution_parameters(), Constraints::continuous_lower_bounds(), Model::continuous_lower_bounds(), Constraints::continuous_upper_bounds(), Model::continuous_upper_bounds(), Variables::continuous_variable_labels(), Model::continuous_variable_labels(), Variables::continuous_variables(), Model::continuous_variables(), Model::currentResponse, Model::currentVariables, Model::discrete_design_set_int_values(), Model::discrete_design_set_real_values(), Constraints::discrete_int_lower_bounds(), Model::discrete_int_lower_bounds(), Constraints::discrete_int_upper_bounds(), Model::discrete_int_upper_bounds(), Variables::discrete_int_variable_labels(), Model::discrete_int_variable_labels(), Variables::discrete_int_variables(), Model::discrete_int_variables(), Constraints::discrete_real_lower_bounds(), Model::discrete_real_lower_bounds(), Constraints::discrete_real_upper_bounds(), Model::discrete_real_upper_bounds(), Variables::discrete_real_variable_labels(), Model::discrete_real_variable_labels(), Variables::discrete_real_variables(), Model::discrete_real_variables(), Model::discrete_state_set_int_values(), Model::discrete_state_set_real_values(), Model::discreteDesignSetIntValues, Model::discreteDesignSetRealValues, Model::discreteStateSetIntValues, Model::discreteStateSetRealValues, Model::epistDistParams, Model::epistemic_distribution_parameters(), Response::function_label(), Constraints::inactive_continuous_lower_bounds(), Model::inactive_continuous_lower_bounds(), Constraints::inactive_continuous_upper_bounds(), Model::inactive_continuous_upper_bounds(), Variables::inactive_continuous_variable_labels(), Model::inactive_continuous_variable_labels(), Variables::inactive_continuous_variables(), Model::inactive_continuous_variables(), Constraints::inactive_discrete_int_lower_bounds(), Model::inactive_discrete_int_lower_bounds(), Constraints::inactive_discrete_int_upper_bounds(), Model::inactive_discrete_int_upper_bounds(), Variables::inactive_discrete_int_variable_labels(), Model::inactive_discrete_int_variable_labels(), Variables::inactive_discrete_int_variables(), Model::inactive_discrete_int_variables(), Constraints::inactive_discrete_real_lower_bounds(), Model::inactive_discrete_real_lower_bounds(), Constraints::inactive_discrete_real_upper_bounds(), Model::inactive_discrete_real_upper_bounds(), Variables::inactive_discrete_real_variable_labels(), Model::inactive_discrete_real_variable_labels(), Variables::inactive_discrete_real_variables(), Model::inactive_discrete_real_variables(), Constraints::linear_eq_constraint_coeffs(), Model::linear_eq_constraint_coeffs(), Constraints::linear_eq_constraint_targets(), Model::linear_eq_constraint_targets(), Constraints::linear_ineq_constraint_coeffs(), Model::linear_ineq_constraint_coeffs(), Constraints::linear_ineq_constraint_lower_bounds(), Model::linear_ineq_constraint_lower_bounds(), Constraints::linear_ineq_constraint_upper_bounds(), Model::linear_ineq_constraint_upper_bounds(), Constraints::nonlinear_eq_constraint_targets(), Model::nonlinear_eq_constraint_targets(), Constraints::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_lower_bounds(), Constraints::nonlinear_ineq_constraint_upper_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Model::num_functions(), Model::num_linear_eq_constraints(), Model::num_linear_ineq_constraints(), Constraints::num_nonlinear_eq_constraints(), Model::num_nonlinear_eq_constraints(), Constraints::num_nonlinear_ineq_constraints(), Model::num_nonlinear_ineq_constraints(), Model::numFns, Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), Model::primaryRespFnSense, Model::primaryRespFnWts, RecastModel::primaryRespMapping, Model::response_labels(), RecastModel::secondaryRespMapping, RecastModel::subModel, Model::userDefinedConstraints, and RecastModel::variablesMapping.

Referenced by RecastModel::update_from_subordinate_model().

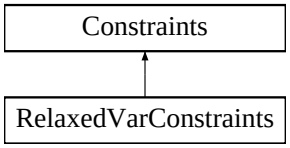
The documentation for this class was generated from the following files:

- RecastModel.hpp
- RecastModel.cpp

13.134 RelaxedVarConstraints Class Reference

Derived class within the Constraints hierarchy which employs relaxation of discrete variables.

Inheritance diagram for RelaxedVarConstraints:



Public Member Functions

- RelaxedVarConstraints (const SharedVariablesData &svd)

  lightweight constructor
- RelaxedVarConstraints (const ProblemDescDB &problem_db, const SharedVariablesData &svd)

  standard constructor
- ~RelaxedVarConstraints ()

  destructor
- void write (std::ostream &s) const

  write a variable constraints object to an std::ostream
- void read (std::istream &s)

  read a variable constraints object from an std::istream

Additional Inherited Members

13.134.1 Detailed Description

Derived class within the Constraints hierarchy which employs relaxation of discrete variables.

Derived variable constraints classes take different views of the design, uncertain, and state variable types and the continuous and discrete domain types. The RelaxedVarConstraints derived class combines continuous and discrete domain types through integer relaxation. The branch and bound method uses this approach (see Variables::get_variables(problem_db) for variables type selection; variables type is passed to the Constraints constructor in Model).

13.134.2 Constructor & Destructor Documentation

RelaxedVarConstraints (const ProblemDescDB & *problem_db*, const SharedVariablesData & *svd*)

standard constructor

In this class, a relaxed data approach is used in which continuous and discrete arrays are combined into a single continuous array (integrality is relaxed; the converse of truncating reals is not currently supported but could be in the future if needed). Iterators which use this class include: BranchBndOptimizer.

References SharedVariablesData::all_relaxed_discrete_int(), SharedVariablesData::all_relaxed_discrete_real(), Constraints::allContinuousLowerBnds, Constraints::allContinuousUpperBnds, Constraints::allDiscreteIntLowerBnds, Constraints::allDiscreteIntUpperBnds, Constraints::allDiscreteRealLowerBnds, Constraints::allDiscreteRealUpperBnds, Dakota::copy_data_partial(), ProblemDescDB::get_iv(), ProblemDescDB::get_rv(), Constraints::sharedVarsData, and SharedVariablesData::view().

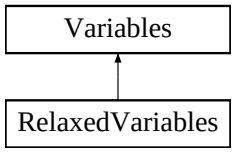
The documentation for this class was generated from the following files:

- RelaxedVarConstraints.hpp
- RelaxedVarConstraints.cpp

13.135 RelaxedVariables Class Reference

Derived class within the Variables hierarchy which employs the relaxation of discrete variables.

Inheritance diagram for RelaxedVariables:



Public Member Functions

- RelaxedVariables (const ProblemDescDB &problem_db, const std::pair< short, short > &view)

  standard constructor
- RelaxedVariables (const SharedVariablesData &svd)

  lightweight constructor
- ~RelaxedVariables ()

  destructor

Protected Member Functions

- void read (std::istream &s)

  read a variables object from an std::istream
- void write (std::ostream &s) const

  write a variables object to an std::ostream
- void write_aprepro (std::ostream &s) const

  write a variables object to an std::ostream in aprepro format
- void read_tabular (std::istream &s, bool active_only=false)
- void write_tabular (std::ostream &s, bool active_only=false) const

  write a variables object in tabular format to an std::ostream

Additional Inherited Members

13.135.1 Detailed Description

Derived class within the Variables hierarchy which employs the relaxation of discrete variables.

Derived variables classes take different views of the design, uncertain, and state variable types and the continuous and discrete domain types. The RelaxedVariables derived class combines continuous and discrete domain types but separates design, uncertain, and state variable types. The branch and bound method uses this approach (see Variables::get_variables(problem_db)).

13.135.2 Constructor & Destructor Documentation

RelaxedVariables (const ProblemDescDB & *problem_db*, const std::pair< short, short > & *view*)

standard constructor

In this class, a relaxed data approach is used in which continuous and discrete arrays are combined into a single continuous array (integrality is relaxed; the converse of truncating reals is not currently supported but could be in the future if needed). Iterators/strategies which use this class include: BranchBndOptimizer. Extract fundamental variable types and labels and merge continuous and discrete domains to create aggregate arrays and views.

References SharedVariablesData::all_relaxed_discrete_int(), SharedVariablesData::all_relaxed_discrete_real(), Variables::allContinuousVars, Variables::allDiscreteIntVars, Variables::allDiscreteRealVars, Variables::allDiscreteStringVars, Dakota::copy_data_partial(), ProblemDescDB::get_iv(), ProblemDescDB::get_rv(), ProblemDescDB::get_sa(), Variables::sharedVarsData, and SharedVariablesData::view().

13.135.3 Member Function Documentation

void read_tabular (std::istream & *s*, bool *active_only* = `false`) `[protected], [virtual]`

Presumes variables object is appropriately sized to receive data

Reimplemented from Variables.

References SharedVariablesData::active_components_totals(), SharedVariablesData::all_relaxed_discrete_int(), SharedVariablesData::all_relaxed_discrete_real(), Variables::allContinuousVars, Variables::allDiscreteIntVars, Variables::allDiscreteRealVars, Variables::allDiscreteStringVars, SharedVariablesData::components_totals(), and Variables::sharedVarsData.

The documentation for this class was generated from the following files:

- RelaxedVariables.hpp
- RelaxedVariables.cpp

13.136 Response Class Reference

Container class for response functions and their derivatives. Response provides the handle class.

Public Member Functions

- Response ()

  default constructor
- Response (const Variables &vars, const ProblemDescDB &problem_db)

  standard constructor built from problem description database
- Response (const ActiveSet &set)

  alternate constructor using limited data
- Response (const Response &response)

  copy constructor
- ~Response ()

  destructor
- Response operator= (const Response &response)

  assignment operator
- size_t num_functions () const

return the number of response functions

- size_t num_scalar_responses () const

  number of scalar responses
- size_t num_field_responses () const

  number of field responses
- size_t num_total_responses () const

  number of total responses (may be redundant, leaving in for now)
- IntVector field_lengths () const

  index of field lengths for field data
- IntVectorArray num_coords_per_field ()

  dimensions of each function
- const ActiveSet & active_set () const

  return the active set
- void active_set (const ActiveSet &set)

  set the active set
- const ShortArray & active_set_request_vector () const

  return the active set request vector
- void active_set_request_vector (const ShortArray &asrv)

  set the active set request vector
- const SizetArray & active_set_derivative_vector () const

  return the active set derivative vector
- void active_set_derivative_vector (const SizetArray &asdv)

  set the active set derivative vector
- const String & responses_id () const

  return the response identifier
- const String & function_label (size_t i) const

  return a response function identifier string
- const StringArray & function_labels () const

  return the response function identifier strings
- void function_label (const String &label, size_t i)

  set a response function identifier string
- void function_labels (const StringArray &labels)

  set the response function identifier strings
- const Real & function_value (size_t i) const

  return a function value
- Real & function_value_view (size_t i)

  return a "view" of a function value for updating in place
- const RealVector & function_values () const

  return all function values
- RealVector function_values_view ()

  return all function values as a view for updating in place
- void function_value (const Real &function_val, size_t i)

  set a function value

- void function_values (const RealVector &function_vals)

  set all function values
- const Real * function_gradient (const int &i) const

  *return the i-th function gradient as a const Real**
- RealVector function_gradient_view (const int &i) const

  return the i-th function gradient as a SerialDenseVector Teuchos::View (shallow copy) for updating in place
- RealVector function_gradient_copy (const int &i) const

  return the i-th function gradient as a SerialDenseVector Teuchos::Copy (deep copy)
- const RealMatrix & function_gradients () const

  return all function gradients
- RealMatrix function_gradients_view ()

  return all function gradients as a view for updating in place
- void function_gradient (const RealVector &function_grad, const int &i)

  set a function gradient
- void function_gradients (const RealMatrix &function_grads)

  set all function gradients
- const RealSymMatrix & function_hessian (size_t i) const

  return the i-th function Hessian
- RealSymMatrix function_hessian_view (size_t i) const

  return the i-th function Hessian as a Teuchos::View (shallow copy) for updating in place
- const RealSymMatrixArray & function_hessians () const

  return all function Hessians
- RealSymMatrixArray function_hessians_view ()

  return all function Hessians as Teuchos::Views (shallow copies) for updating in place
- void function_hessian (const RealSymMatrix &function_hessian, size_t i)

  set a function Hessian
- void function_hessians (const RealSymMatrixArray &function_hessians)

  set all function Hessians
- RealVector field_values_view (size_t i)

  return a "view" of a field value for updating in place
- void field_values (const RealVector &field_val, size_t i)

  set a field value
- void read (std::istream &s)

  read a response object from an std::istream
- void write (std::ostream &s) const

  write a response object to an std::ostream
- void read_annotated (std::istream &s)

  read a response object in annotated format from an std::istream
- void write_annotated (std::ostream &s) const

  write a response object in annotated format to an std::ostream
- void read_tabular (std::istream &s)

  read responseRep::functionValues in tabular format from an std::istream
- void write_tabular (std::ostream &s) const

write responseRep::functionValues in tabular format to an std::ostream

- void write_tabular_labels (std::ostream &s) const

  write the labels to a tabular data stream
- void read (MPIUnpackBuffer &s)

  read a response object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a response object to a packed MPI buffer
- Response copy () const

  a deep copy for use in history mechanisms
- int data_size ()

  handle class forward to corresponding body class member function
- void read_data (double *response_data)

  handle class forward to corresponding body class member function
- void write_data (double *response_data)

  handle class forward to corresponding body class member function
- void overlay (const Response &response)

  handle class forward to corresponding body class member function
- void update (const Response &response)

  Used in place of operator= when only results data updates are desired (functionValues/functionGradients/functionHessians are updated, ASV/labels/id's/etc. are not). Care is taken to allow different derivative array sizing between the two response objects.
- void update (const RealVector &source_fn_vals, const RealMatrix &source_fn_grads, const RealSymMatrixArray &source_fn_hessians, const ActiveSet &source_set)

  Overloaded form which allows update from components of a response object. Care is taken to allow different derivative array sizing.
- void update_partial (size_t start_index_target, size_t num_items, const Response &response, size_t start_index_source)

  partial update of this response object from another response object. The response objects may have different numbers of response functions.
- void update_partial (size_t start_index_target, size_t num_items, const RealVector &source_fn_vals, const RealMatrix &source_fn_grads, const RealSymMatrixArray &source_fn_hessians, const ActiveSet &source_set, size_t start_index_source)

  Overloaded form which allows partial update from components of a response object. The response objects may have different numbers of response functions.
- void reshape (size_t num_fns, size_t num_params, bool grad_flag, bool hess_flag)

  rehapes response data arrays
- void reset ()

  handle class forward to corresponding body class member function
- void reset_inactive ()

  handle class forward to corresponding body class member function
- bool is_null () const

  function to check responseRep (does this handle contain a body)
- template<class Archive >
  void load (Archive &ar, const unsigned int version)
- template<class Archive >
  void save (Archive &ar, const unsigned int version) const

Private Member Functions

- template<class Archive >
  void load (Archive &ar, const unsigned int version)

  read a Response from an archive
- template<class Archive >
  void save (Archive &ar, const unsigned int version) const

  write a Response to an archive
- BOOST_SERIALIZATION_SPLIT_MEMBER () ResponseRep *responseRep

  pointer to the body (handle-body idiom)

Friends

- class **boost::serialization::access**
- bool operator== (const Response &resp1, const Response &resp2)

  equality operator
- bool operator!= (const Response &resp1, const Response &resp2)

  inequality operator

13.136.1 Detailed Description

Container class for response functions and their derivatives. Response provides the handle class.

The Response class is a container class for an abstract set of functions (functionValues) and their first (functionGradients) and second (functionHessians) derivatives. The functions may involve objective and constraint functions (optimization data set), least squares terms (parameter estimation data set), or generic response functions (uncertainty quantification data set). It is not currently part of a class hierarchy, since the abstraction has been sufficiently general and has not required specialization. For memory efficiency, it employs the "handle-body idiom" approach to reference counting and representation sharing (see Coplien "Advanced C++", p. 58), for which Response serves as the handle and ResponseRep serves as the body.

13.136.2 Member Function Documentation

void load (Archive & *ar*, const unsigned int *version*)

Implementation of serialization load for the Response handle

void save (Archive & *ar*, const unsigned int *version*) const

Implementation of serialization save for the Response handle

The documentation for this class was generated from the following files:

- DakotaResponse.hpp
- DakotaResponse.cpp

13.137 ResponseRep Class Reference

Container class for response functions and their derivatives. ResponseRep provides the body class.

Public Member Functions

- template<class Archive , typename OrdinalType , typename ScalarType >
  void write_sdm_col (Archive &ar, int col, const Teuchos::SerialDenseMatrix< OrdinalType, ScalarType > &sdm) const

  convenience fnction to write a serial dense matrix column to an Archive
- template<class Archive , typename OrdinalType , typename ScalarType >
  void read_sdm_col (Archive &ar, int col, Teuchos::SerialDenseMatrix< OrdinalType, ScalarType > &sdm)

  convenience fnction to read a serial dense matrix column from an Archive
- template<class Archive >
  void load (Archive &ar, const unsigned int version)
- template<class Archive >
  void save (Archive &ar, const unsigned int version) const

Private Member Functions

- ResponseRep ()

  default constructor
- ResponseRep (const Variables &vars, const ProblemDescDB &problem_db)

  standard constructor built from problem description database
- ResponseRep (const ActiveSet &set)

  alternate constructor using limited data
- ~ResponseRep ()

  destructor
- void read (std::istream &s)

  read a responseRep object from an std::istream
- void write (std::ostream &s) const

  write a responseRep object to an std::ostream
- void read_annotated (std::istream &s)

  read a responseRep object from an std::istream (annotated format)
- void write_annotated (std::ostream &s) const

  write a responseRep object to an std::ostream (annotated format)
- void read_tabular (std::istream &s)

  read functionValues from an std::istream (tabular format)
- void write_tabular (std::ostream &s) const

  write functionValues to an std::ostream (tabular format)
- void write_tabular_labels (std::ostream &s) const

  write the labels to a tabular data stream
- void read (MPIUnpackBuffer &s)

  read a responseRep object from a packed MPI buffer
- void write (MPIPackBuffer &s) const

  write a responseRep object to a packed MPI buffer
- template<class Archive , typename OrdinalType , typename ScalarType >
  void write_sdm_col (Archive &ar, int col, const Teuchos::SerialDenseMatrix< OrdinalType, ScalarType > &sdm) const

write a column of a SerialDenseMatrix

- template<class Archive , typename OrdinalType , typename ScalarType >
  void read_sdm_col (Archive &ar, int col, Teuchos::SerialDenseMatrix< OrdinalType, ScalarType > &sdm)

  read a column of a SerialDenseMatrix
- template<class Archive >
  void load (Archive &ar, const unsigned int version)

  read a ResponseRep from an archive
- template<class Archive >
  void save (Archive &ar, const unsigned int version) const

  write a ResponseRep to an archive
- BOOST_SERIALIZATION_SPLIT_MEMBER () int data_size()

  return the number of doubles active in response. Used for sizing double∗ response_data arrays passed into read_data and write_data.
- void read_data (double ∗response_data)

  read from an incoming double∗ array
- void write_data (double ∗response_data)

  write to an incoming double∗ array
- void overlay (const Response &response)

  add incoming response to functionValues/Gradients/Hessians
- void update (const RealVector &source_fn_vals, const RealMatrix &source_fn_grads, const RealSymMatrixArray &source_fn_hessians, const ActiveSet &source_set)

  update this response object from components of another response object
- void update_partial (size_t start_index_target, size_t num_items, const RealVector &source_fn_vals, const RealMatrix &source_fn_grads, const RealSymMatrixArray &source_fn_hessians, const ActiveSet &source_set, size_t start_index_source)

  partially update this response object partial components of another response object
- void reshape (size_t num_fns, size_t num_params, bool grad_flag, bool hess_flag)

  rehapes response data arrays
- void reset ()

  resets all response data to zero
- void reset_inactive ()

  resets all inactive response data to zero
- void active_set_request_vector (const ShortArray &asrv)

  set the active set request vector and verify consistent number of response functions
- void active_set_derivative_vector (const SizetArray &asdv)

  set the active set derivative vector and reshape functionGradients/functionHessians if needed

Private Attributes

- int referenceCount

  number of handle objects sharing responseRep
- size_t numScalarResponses

  number of scalar responses
- size_t numFieldResponses

number of field responses

- size_t numTotalResponses

  number of total responses (may be redundant, leaving in for now)
- IntVector fieldLengths

  index of field lengths for field data
- IntVectorArray numCoordsPerField

  dimensions of each function
- RealVectorArray coordsValuesPerField

  independent coordinate values
- RealVector functionValues

  abstract set of response functions
- RealMatrix functionGradients

  first derivatives of the response functions
- RealSymMatrixArray functionHessians

  second derivatives of the response functions
- ActiveSet responseActiveSet

  copy of the ActiveSet used by the Model to generate a Response instance
- StringArray functionLabels

  response function identifiers used to improve output readability
- String responsesId

  response identifier string from the input file

Friends

- class boost::serialization::access

  for serializing private data members
- class Response

  the handle class can access attributes of the body class directly
- bool operator== (const ResponseRep &rep1, const ResponseRep &rep2)

  equality operator

13.137.1 Detailed Description

Container class for response functions and their derivatives. ResponseRep provides the body class.

The ResponseRep class is the "representation" of the response container class. It is the "body" portion of the "handle-body idiom" (see Coplien "Advanced C++", p. 58). The handle class (Response) provides for memory efficiency in management of multiple response objects through reference counting and representation sharing. The body class (ResponseRep) actually contains the response data (functionValues, functionGradients, functionHessians, etc.). The representation is hidden in that an instance of ResponseRep may only be created by Response. Therefore, programmers create instances of the Response handle class, and only need to be aware of the handle/body mechanisms when it comes to managing shallow copies (shared representation) versus deep copies (separate representation used for history mechanisms).

13.137.2 Constructor & Destructor Documentation

ResponseRep (const Variables & *vars,* const ProblemDescDB & *problem_db*) [private]

standard constructor built from problem description database

The standard constructor used by Dakota::ModelRep.

References Dakota::build_label(), Variables::continuous_variable_ids(), Variables::cv(), ActiveSet::derivative_vector(), ResponseRep::fieldLengths, ResponseRep::functionGradients, ResponseRep::functionHessians, ResponseRep::functionLabels, ResponseRep::functionValues, ProblemDescDB::get_iv(), ProblemDescDB::get_sizet(), ProblemDescDB::get_string(), ResponseRep::numFieldResponses, ResponseRep::numScalarResponses, ResponseRep::numTotalResponses, ActiveSet::request_vector(), and ResponseRep::responseActiveSet.

ResponseRep (const ActiveSet & *set*) [private]

alternate constructor using limited data

Used for building a response object of the correct size on the fly (e.g., by slave analysis servers performing execute() on a local_response). functionLabels is not needed for this purpose since it's not passed in the MPI send/recv buffers. However, NPSOLOptimizer's user-defined functions option uses this constructor to build bestResponseArray.front() and bestResponseArray.front() needs functionLabels for I/O, so construction of functionLabels has been added.

References Dakota::build_labels(), ActiveSet::derivative_vector(), ResponseRep::functionGradients, ResponseRep::functionHessians, ResponseRep::functionLabels, and ActiveSet::request_vector().

13.137.3 Member Function Documentation

void read (std::istream & *s*) [private]

read a responseRep object from an std::istream

ASCII version of read needs capabilities for capturing data omissions or formatting errors (resulting from user error or asynch race condition) and analysis failures (resulting from nonconvergence, instability, etc.).

References ResponseRep::functionGradients, ResponseRep::functionHessians, ResponseRep::functionValues, Dakota::re_match(), ActiveSet::request_vector(), ResponseRep::reset(), and ResponseRep::responseActiveSet.

void read_annotated (std::istream & *s*) [private]

read a responseRep object from an std::istream (annotated format)

read_annotated() is used for neutral file translation of restart files. Since objects are built solely from this data, annotations are used. This version closely mirrors the BiStream version.

References ResponseRep::functionGradients, ResponseRep::functionHessians, ResponseRep::functionLabels, ResponseRep::functionValues, ActiveSet::request_vector(), ResponseRep::reset(), ActiveSet::reshape(), ResponseRep::reshape(), and ResponseRep::responseActiveSet.

void write_annotated (std::ostream & *s*) const [private]

write a responseRep object to an std::ostream (annotated format)

write_annotated() is used for neutral file translation of restart files. Since objects need to be build solely from this data, annotations are used. This version closely mirrors the BoStream version, with the exception of the use of white space between fields.

References ActiveSet::derivative_vector(), ResponseRep::functionGradients, ResponseRep::functionHessians, ResponseRep::functionValues, ActiveSet::request_vector(), ResponseRep::responseActiveSet, and ActiveSet::write_annotated().

void read_tabular (std::istream & *s*) [private]

read functionValues from an std::istream (tabular format)

read_tabular is used to read functionValues in tabular format. It is currently only used by ApproximationInterfaces in reading samples from a file. There is insufficient data in a tabular file to build complete response objects; rather, the response object must be constructed a priori and then its functionValues can be set.

References ResponseRep::functionValues.

void write_tabular (std::ostream & *s*) const [private]

write functionValues to an std::ostream (tabular format)

write_tabular is used for output of functionValues in a tabular format for convenience in post-processing/plotting of DAKOTA results.

References ResponseRep::functionValues, ActiveSet::request_vector(), ResponseRep::responseActiveSet, and Dakota::write_precision.

void read (MPIUnpackBuffer & *s*) [private]

read a responseRep object from a packed MPI buffer

UnpackBuffer version differs from BiStream version in the omission of functionLabels. Master processor retains labels and interface ids and communicates asv and response data only with slaves.

References ResponseRep::functionGradients, ResponseRep::functionHessians, ResponseRep::functionValues, ResponseRep::reset(), ResponseRep::reshape(), and ResponseRep::responseActiveSet.

void update (const RealVector & *source_fn_vals,* const RealMatrix & *source_fn_grads,* const RealSymMatrixArray & *source_fn_hessians,* const ActiveSet & *source_set*) [private]

update this response object from components of another response object

Copy function values/gradients/Hessians data *only*. Prevents unwanted overwriting of responseActiveSet, functionLabels, etc. Also, care is taken to account for differences in derivative variable matrix sizing.

References Dakota::abort_handler(), ActiveSet::derivative_vector(), ResponseRep::functionGradients, ResponseRep::functionHessians, ActiveSet::request_vector(), and ResponseRep::reset_inactive().

void update_partial (size_t *start_index_target,* size_t *num_items,* const RealVector & *source_fn_vals,* const RealMatrix & *source_fn_grads,* const RealSymMatrixArray & *source_fn_hessians,* const ActiveSet & *source_set,* size_t *start_index_source*) [private]

partially update this response object partial components of another response object

Copy function values/gradients/Hessians data *only*. Prevents unwanted overwriting of responseActiveSet, functionLabels, etc. Also, care is taken to account for differences in derivative variable matrix sizing.

References Dakota::abort_handler(), ActiveSet::derivative_vector(), ResponseRep::functionGradients, ResponseRep::functionHessians, ActiveSet::request_vector(), and ResponseRep::reset_inactive().

void reshape (size_t *num_fns,* size_t *num_params,* bool *grad_flag,* bool *hess_flag*) [private]

rehapes response data arrays

Reshape functionValues, functionGradients, and functionHessians according to num_fns, num_params, grad_flag, and hess_flag.

References Dakota::build_labels(), ResponseRep::functionGradients, and ResponseRep::functionHessians.

Referenced by ResponseRep::active_set_derivative_vector(), ResponseRep::load(), ResponseRep::read(), and ResponseRep::read_annotated().

void reset ( ) [private]

resets all response data to zero

Reset all numerical response data (not labels, ids, or active set) to zero.

References ResponseRep::functionGradients, and ResponseRep::functionHessians.

Referenced by ResponseRep::load(), ResponseRep::read(), and ResponseRep::read_annotated().

void reset_inactive ( ) [private]

resets all inactive response data to zero

Used to clear out any inactive data left over from previous evaluations.

References ResponseRep::functionGradients, and ResponseRep::functionHessians.

Referenced by ResponseRep::update(), and ResponseRep::update_partial().

void load (Archive & *ar,* const unsigned int *version*)

Binary version differs from ASCII version in 2 primary ways: (1) it lacks formatting. (2) the Response has not been sized a priori. In reading data from the binary restart file, a ParamResponsePair was constructed with its default constructor which called the Response default constructor. Therefore, we must first read sizing data and resize all of the arrays.

References ResponseRep::functionGradients, ResponseRep::functionHessians, ResponseRep::functionLabels, ResponseRep::read_sdm_col(), ResponseRep::reset(), ResponseRep::reshape(), and ResponseRep::responseActiveSet.

void save (Archive & *ar,* const unsigned int *version*) const

Binary version differs from ASCII version in 2 primary ways: (1) It lacks formatting. (2) In reading data from the binary restart file, ParamResponsePairs are constructed with their default constructor which calls the Response default constructor. Therefore, we must first write sizing data so that ResponseRep::read(BoStream& s) can resize the arrays.

References ResponseRep::functionGradients, ResponseRep::functionHessians, ResponseRep::functionLabels, ResponseRep::responseActiveSet, and ResponseRep::write_sdm_col().

13.137.4 Member Data Documentation

RealMatrix functionGradients [private]

first derivatives of the response functions

the gradient vectors (plural) are column vectors in the matrix (singular) with (row, col) = (variable index, response fn index).

Referenced by ResponseRep::active_set_derivative_vector(), ResponseRep::load(), ResponseRep::overlay(), ResponseRep::read(), ResponseRep::read_annotated(), ResponseRep::read_data(), ResponseRep::reset(), ResponseRep::reset_inactive(), ResponseRep::reshape(), ResponseRep::ResponseRep(), ResponseRep::save(), ResponseRep::update(), ResponseRep::update_partial(), ResponseRep::write_annotated(), and ResponseRep::write_data().

The documentation for this class was generated from the following files:

- DakotaResponse.hpp
- DakotaResponse.cpp

13.138 RestartWriter Class Reference

Public Member Functions

- RestartWriter ()

  optional default ctor allowing a non-outputting RestartWriter

- RestartWriter (const String &write_restart_filename)

  typical ctor taking a filename

- const String & filename ()

  output filename for this writer

- void append_prp (const ParamResponsePair &prp_in)

  add the passed pair to the restart file

- void flush ()

  flush the restart stream so we have a complete restart record should Dakota abort

Private Member Functions

- RestartWriter (const RestartWriter &)

  copy constructor is disallowed due to file stream

- const RestartWriter & operator= (const RestartWriter &)

  assignment is disallowed due to file stream

Private Attributes

- String restartOutputFilename

  the name of the restart output file

- std::ofstream restartOutputFS

  Binary stream to which restart data is written.

- boost::scoped_ptr
  < boost::archive::binary_oarchive > restartOutputArchive

  Binary output archive to which data is written (pointer since no default ctor for oarchive and may not be initialized);.

13.138.1 Detailed Description

Component for writing restart files. Creation and destruction of archive and associated stream are managed here.

The documentation for this class was generated from the following files:

- OutputManager.hpp
- OutputManager.cpp

13.139 ResultsDBAny Class Reference

Public Member Functions

- template<typename StoredType >
  void array_allocate (const StrStrSizet &iterator_id, const std::string &data_name, size_t array_size, const MetaDataType &metadata)

  allocate an entry with sized array of the StoredType, e.g., array across response functions or optimization results sets
- template<typename StoredType >
  void array_insert (const StrStrSizet &iterator_id, const std::string &data_name, size_t index, const StoredType &sent_data)

  insert sent_data in specified position in previously allocated array
- template<typename StoredType >
  StoredType get_data (const StrStrSizet &iterator_id, const std::string &data_name) const

  return requested data by value in StoredType
- template<typename StoredType >
  StoredType get_array_data (const StrStrSizet &iterator_id, const std::string &data_name, size_t index) const

  return requested data from array by value in StoredType
- template<typename StoredType >
  const StoredType * get_data_ptr (const StrStrSizet &iterator_id, const std::string &result_key) const

  return pointer to stored data entry
- template<typename StoredType >
  const StoredType * get_array_data_ptr (const StrStrSizet &iterator_id, const std::string &data_name, size_t index) const

  return pointer to stored data at given array location
- void insert (const StrStrSizet &iterator_id, const std::string &data_name, const boost::any &result, const MetaDataType &metadata)

  record addition with metadata map
- void dump_data (std::ostream &output_stream)

  coarsely dump the data to the passed output stream
- void print_data (std::ostream &output_stream)

  pretty print the data to the passed output stream

Private Member Functions

- const ResultsValueType & lookup_data (const StrStrSizet &iterator_id, const std::string &data_name) const

  attempt to find the requested data, erroring if not found
- template<typename StoredType >
  StoredType cast_data (const boost::any &dataholder) const

  cast the reference to the any data to the requested type
- template<typename StoredType >
  const StoredType * cast_data_ptr (const boost::any *dataholder) const

  cast the pointer to the any data to the requested type

- void print_metadata (std::ostream &os, const MetaDataType &md)

  print metadata to ostream
- void extract_data (const boost::any &dataholder, std::ostream &os)

  determine the type of contained data and output it to ostream
- void output_data (const std::vector< double > &data, std::ostream &os)

  output data to ostream
- void output_data (const std::vector< RealVector > &data, std::ostream &os)

  output data to ostream
- void output_data (const std::vector< std::string > &data, std::ostream &os)

  output data to ostream
- void output_data (const std::vector< std::vector< std::string > > &data, std::ostream &os)

  output data to ostream
- void output_data (const std::vector< RealMatrix > &data, std::ostream &os)

  output data to ostream
- void output_data (const RealMatrix &data, std::ostream &os)

  output data to ostream

Private Attributes

- std::map< ResultsKeyType, ResultsValueType > iteratorData

  core data storage (map from key to value type)

13.139.1 Detailed Description

Class: ResultsDBAny Description: A map-based container to store DAKOTA Iterator results in underlying boost::anys, with optional metadata

13.139.2 Member Function Documentation

void array_insert (const StrStrSizet & *iterator_id,* const std::string & *data_name,* size_t *index,* const StoredType & *sent_data*)

insert sent_data in specified position in previously allocated array

insert requires previous allocation, and does not allow metadata update

References Dakota::abort_handler(), ResultsDBAny::iteratorData, and Dakota::make_key().

void insert (const StrStrSizet & *iterator_id,* const std::string & *data_name,* const boost::any & *result,* const MetaDataType & *metadata*)

record addition with metadata map

Add or update existing entry

References ResultsDBAny::iteratorData, and Dakota::make_key().

Referenced by ResultsDBAny::array_allocate().

void extract_data (const boost::any & *dataholder*, std::ostream & *os*) [private]

determine the type of contained data and output it to ostream

Extract the data from the held any and map to supported concrete types int double RealVector (Teuchos::-SerialDenseVector<int,double) RealMatrix (Teuchos::SerialDenseMatrix<int,double)

References ResultsDBAny::output_data().

Referenced by ResultsDBAny::dump_data(), and ResultsDBAny::print_data().

The documentation for this class was generated from the following files:

- ResultsDBAny.hpp
- ResultsDBAny.cpp

13.140 ResultsEntry< StoredType > Class Template Reference

Class to manage in-core vs. file database lookups.

Public Member Functions

- ResultsEntry (const ResultsManager &results_mgr, const StrStrSizet &iterator_id, const std::string &data_-name)

  Construct ResultsEntry containing retrieved item of StoredType.
- ResultsEntry (const ResultsManager &results_mgr, const StrStrSizet &iterator_id, const std::string &data_-name, size_t array_index)

  Construct ResultsEntry to retrieve item array_index from array of StoredType.

Private Member Functions

- ResultsEntry ()

  return a reference to the stored data, whether from core or file

Private Attributes

- bool coreActive

  whether the ResultsManager has an active in-core database
- StoredType dbData

  data retrieved from file data base
- const StoredType * dbDataPtr

  non-const pointer to const data we don't own in the core case

13.140.1 Detailed Description

template<typename StoredType>class Dakota::ResultsEntry< StoredType >

Class to manage in-core vs. file database lookups.

ResultsEntry manages database lookups. If a core database is available, will return a reference directly to the stored data; if disk, will return reference to a local copy contained in this class. Allows disk-stored data to persist for minimum time during lookup to support true out-of-core use cases.

13.140.2 Constructor & Destructor Documentation

ResultsEntry ( ) [private]

return a reference to the stored data, whether from core or file
default construction disallowed: data must be initialized from DB lookup if needed
The documentation for this class was generated from the following file:

- ResultsManager.hpp

13.141 ResultsID Class Reference

Get a globally unique 1-based execution number for a given iterator name (combination of methodName and methodID) for use in results DB. Each Iterator::run() call creates or increments this count for its string identifier.

Public Member Functions

- size_t increment_id (const std::string &method_name, const std::string &method_id)

  explicitly increment the iterator results ID, init to 1 if needed
- size_t get_id (const std::string &method_name, const std::string &method_id)

  get (possibly creating) a unique iterator results ID for the passed name
- size_t get_id (const std::string &method_name, const std::string &method_id) const

  get a unique iterator results ID for the passed name (const version errors if not found)

Static Public Member Functions

- static ResultsID & instance ()

  get the single unique instance of ResultsID

Private Member Functions

- ResultsID ()

  Private constructor for ResultsID.
- ~ResultsID ()

  Private destructor for ResultsID.
- ResultsID (ResultsID const &)

  Private copy constructor for ResultsID.
- ResultsID & operator= (ResultsID const &)

  Private assignment operator for ResultsID.

Private Attributes

- std::map< std::pair
  < std::string, std::string >
  , size_t > idMap

  storage for the results IDs

13.141.1 Detailed Description

Get a globally unique 1-based execution number for a given iterator name (combination of methodName and methodID) for use in results DB. Each Iterator::run() call creates or increments this count for its string identifier.

The documentation for this class was generated from the following files:

- ResultsManager.hpp
- ResultsManager.cpp

13.142 ResultsManager Class Reference

Results manager for iterator final data.

Public Member Functions

- ResultsManager ()

  default constructor: no databases active until initialize called
- void initialize (const std::string &base_filename)

  initialize the results manager to manage an in-core database, writing to the specified file name
- bool active () const

  whether any databases are active
- void write_databases ()

  Write in-core databases to file.
- template<typename StoredType >
  void insert (const StrStrSizet &iterator_id, const std::string &data_name, const StoredType &sent_data, const MetaDataType metadata=MetaDataType())

  insert data
- void **insert** (const StrStrSizet &iterator_id, const std::string &data_name, StringMultiArrayConstView sma_labels, const MetaDataType metadata=MetaDataType())
- template<typename StoredType >
  void array_allocate (const StrStrSizet &iterator_id, const std::string &data_name, size_t array_size, const MetaDataType metadata=MetaDataType())

  allocate an entry with array of StoredType of array_size for future insertion; likely move to non-templated accessors for these
- template<typename StoredType >
  void array_insert (const StrStrSizet &iterator_id, const std::string &data_name, size_t index, const StoredType &sent_data)

  insert into a previously allocated array of StoredType at index specified; metadata must be specified at allocation
- template<typename StoredType >
  void array_insert (const StrStrSizet &iterator_id, const std::string &data_name, size_t index, StringMultiArrayConstView sent_data)

  specialization: insert a SMACV into a previously allocated array of StringArrayStoredType at index specified; metadata must be specified at allocation

Public Attributes

- ResultsNames results_names

  Copy of valid results names for when manager is passed around.

Private Member Functions

- template<typename StoredType >
  StoredType core_lookup (const StrStrSizet &iterator_id, const std::string &data_name) const

  retrieve in-core entry given by id and name
- template<typename StoredType >
  StoredType * core_lookup_ptr (const StrStrSizet &iterator_id, const std::string &data_name) const

  retrieve data via pointer to avoid copy; work-around for Boost any use of pointer (could use utilib::Any)
- template<typename StoredType >
  StoredType core_lookup (const StrStrSizet &iterator_id, const std::string &data_name, size_t index) const

  retrieve data from in-core array of StoredType at given index
- template<typename StoredType >
  const StoredType * core_lookup_ptr (const StrStrSizet &iterator_id, const std::string &data_name, size_t index) const

  retrieve data via pointer to entry in in-core array
- template<typename StoredType >
  void file_lookup (StoredType &db_data, const StrStrSizet &iterator_id, const std::string &data_name) const

  retrieve requested data into provided db_data StoredType

Private Attributes

- bool coreDBActive

  whether the in-core database in active
- std::string coreDBFilename

  filename for the in-core database
- bool hdf5DBActive

  whether the file database is active
- boost::scoped_ptr< ResultsDBAny > coreDB

  In-core database, with option to flush to file at end.
- boost::shared_ptr< ResultsDBHDF5 > hdf5DB

  File-based database; using shared_ptr due to potentially incomplete type and requirements for checked_delete in debug builds.

Friends

- template<typename StoredType >
  class ResultsEntry

  ResultsEntry is a friend of ResultsManager.

13.142.1 Detailed Description

Results manager for iterator final data.

The results manager provides the API for posting and retrieving iterator results data (and eventually run config/statistics). It can manage a set of underlying results databases, in or out of core, depending on configuration

The key for a results entry is documented in results_types.hpp, e.g., tuple<std::string, std::string, size_t, std::string>

For now, using concrete types for most insertion, since underlying databases like HDF5 might need concrete types; though template parameter for array allocation and retrieval.

All insertions overwrite any previous data.

The documentation for this class was generated from the following files:

- ResultsManager.hpp
- ResultsManager.cpp

13.143 ResultsNames Class Reference

List of valid names for iterator results.

Public Member Functions

- ResultsNames ()

  Default constructor initializes all valid names.

Public Attributes

- size_t **namesVersion**
- std::string **best_cv**
- std::string **best_div**
- std::string **best_dsv**
- std::string **best_drv**
- std::string **best_fns**
- std::string **moments_std**
- std::string **moments_central**
- std::string **moments_std_num**
- std::string **moments_central_num**
- std::string **moments_std_exp**
- std::string **moments_central_exp**
- std::string **moment_cis**
- std::string **extreme_values**
- std::string **map_resp_prob**
- std::string **map_resp_rel**
- std::string **map_resp_genrel**
- std::string **map_prob_resp**
- std::string **map_rel_resp**
- std::string **map_genrel_resp**
- std::string **pdf_histograms**
- std::string **correl_simple_all**
- std::string **correl_simple_io**
- std::string **correl_partial_io**
- std::string **correl_simple_rank_all**
- std::string **correl_simple_rank_io**
- std::string **correl_partial_rank_io**
- std::string **pce_coeffs**

- std::string **pce_coeff_labels**
- std::string **cv_labels**
- std::string **div_labels**
- std::string **dsv_labels**
- std::string **drv_labels**
- std::string **fn_labels**

13.143.1 Detailed Description

List of valid names for iterator results.

All data in the ResultsNames class is public, basically just a struct

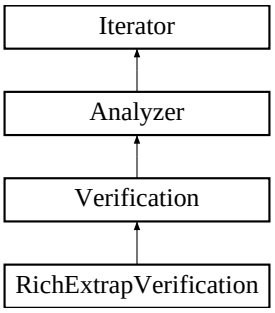
The documentation for this class was generated from the following file:

- ResultsManager.hpp

13.144 RichExtrapVerification Class Reference

Class for Richardson extrapolation for code and solution verification.

Inheritance diagram for RichExtrapVerification:



Public Member Functions

- RichExtrapVerification (ProblemDescDB &problem_db, Model &model)

  constructor
- ~RichExtrapVerification ()

  destructor
- void perform_verification ()

  Redefines the core_run() virtual function for the Verification branch.
- void print_results (std::ostream &s)

  print the final iterator results

Private Member Functions

- void estimate_order ()

  perform a single estimation of convOrder using extrapolation()
- void converge_order ()

iterate using extrapolation() until convOrder stabilizes

- void converge_qoi ()

  iterate using extrapolation() until QOIs stabilize

- void extrapolation (const RealVector &refine_triple, RealMatrix &qoi_triples)

  estimate convOrder from refinement and quantity of interest (QOI) triples

- void extrapolate_result (const RealVector &refine_triple, const RealMatrix &qoi_triples)

  predict the converged value based on the convergence rate and the value of Phi

Private Attributes

- unsigned short studyType

  internal code for extrapolation study type: SUBMETHOD_{CONVERGE_ORDER,CONVERGE_QOI,ESTIMATE-_ORDER}

- size_t numFactors

  number of refinement factors defined from active state variables

- RealVector initialCVars

  initial reference values for refinement factors

- size_t factorIndex

  the index of the active factor

- Real refinementRate

  rate of mesh refinement (default = 2.)

- RealMatrix convOrder

  the orders of convergence of the QOIs (numFunctions by numFactors)

- RealMatrix extrapQOI

  the extrapolated value of the QOI (numFunctions by numFactors)

- RealMatrix numErrorQOI

  the numerical uncertainty associated with level of refinement (numFunctions by numFactors)

- RealVector refinementRefPt

  This is a reference point reported for the converged extrapQOI and numErrorQOI. It currently corresponds to the coarsest mesh in the final refinement triple.

Additional Inherited Members

13.144.1 Detailed Description

Class for Richardson extrapolation for code and solution verification.

The RichExtrapVerification class contains several algorithms for performing Richardson extrapolation.

13.144.2 Member Function Documentation

void print_results (std::ostream & *s*) `[virtual]`

print the final iterator results

This virtual function provides additional iterator-specific final results outputs beyond the function evaluation summary printed in finalize_run().

Reimplemented from Verification.

References Model::continuous_variable_labels(), RichExtrapVerification::convOrder, Dakota::copy_data(), RichExtrapVerification::extrapQOI, Iterator::iteratedModel, RichExtrapVerification::numErrorQOI, Verification::print_results(), RichExtrapVerification::refinementRate, RichExtrapVerification::refinementRefPt, and Model::response_labels().

void estimate_order ( ) [private]

perform a single estimation of convOrder using extrapolation()

This algorithm executes a single refinement triple and returns convergence order estimates.

References RichExtrapVerification::extrapolate_result(), RichExtrapVerification::extrapolation(), RichExtrapVerification::extrapQOI, RichExtrapVerification::factorIndex, RichExtrapVerification::initialCVars, RichExtrapVerification::numErrorQOI, RichExtrapVerification::numFactors, Analyzer::numFunctions, RichExtrapVerification::refinementRate, and RichExtrapVerification::refinementRefPt.

Referenced by RichExtrapVerification::perform_verification().

void converge_order ( ) [private]

iterate using extrapolation() until convOrder stabilizes

This algorithm continues to refine until the convergence order estimate converges.

References Iterator::convergenceTol, RichExtrapVerification::convOrder, Dakota::copy_data(), RichExtrapVerification::extrapolate_result(), RichExtrapVerification::extrapolation(), RichExtrapVerification::extrapQOI, RichExtrapVerification::factorIndex, RichExtrapVerification::initialCVars, Iterator::maxIterations, RichExtrapVerification::numErrorQOI, RichExtrapVerification::numFactors, Analyzer::numFunctions, Iterator::outputLevel, RichExtrapVerification::refinementRate, and RichExtrapVerification::refinementRefPt.

Referenced by RichExtrapVerification::perform_verification().

void converge_qoi ( ) [private]

iterate using extrapolation() until QOIs stabilize

This algorithm continues to refine until the discretization error lies within a prescribed tolerance.

References Iterator::convergenceTol, RichExtrapVerification::extrapolate_result(), RichExtrapVerification::extrapolation(), RichExtrapVerification::extrapQOI, RichExtrapVerification::factorIndex, RichExtrapVerification::initialCVars, Iterator::maxIterations, RichExtrapVerification::numErrorQOI, RichExtrapVerification::numFactors, Analyzer::numFunctions, Iterator::outputLevel, RichExtrapVerification::refinementRate, and RichExtrapVerification::refinementRefPt.

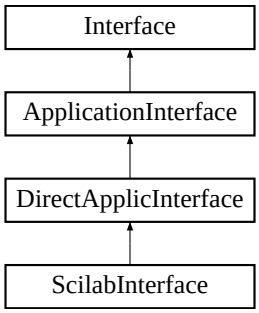
Referenced by RichExtrapVerification::perform_verification().

The documentation for this class was generated from the following files:

- RichExtrapVerification.hpp
- RichExtrapVerification.cpp

13.145 ScilabInterface Class Reference

Inheritance diagram for ScilabInterface:



Public Member Functions

- ScilabInterface (const ProblemDescDB &problem_db)

  Constructor: start Matlab engine.
- ~ScilabInterface ()

  Destructor: close Matlab engine.

Protected Member Functions

- virtual int derived_map_ac (const String &ac_name)

  execute an analysis code portion of a direct evaluation invocation
- int scilab_engine_run (const String &ac_name)

  principal Scilab execute function

Protected Attributes

- int scilabEngine

  identifier for the running Scilab enginer

13.145.1 Detailed Description

Specialization of DirectApplicInterface to link to Scilab analysis drivers. Includes convenience functions to map data to/from Scilab

The documentation for this class was generated from the following files:

- ScilabInterface.hpp
- ScilabInterface.cpp

13.146 SensAnalysisGlobal Class Reference

Class for a utility class containing correlation calculations and variance-based decomposition.

Public Member Functions

- SensAnalysisGlobal ()

  constructor
- ~SensAnalysisGlobal ()

  destructor
- void compute_correlations (const VariablesArray &vars_samples, const IntResponseMap &resp_samples, const StringSetArray &dss_vals)

  computes four correlation matrices for input and output data simple, partial, simple rank, and partial rank
- void compute_correlations (const RealMatrix &vars_samples, const IntResponseMap &resp_samples)

  computes four correlation matrices for input and output data simple, partial, simple rank, and partial rank
- void archive_correlations (const StrStrSizet &run_identifier, ResultsManager &iterator_results, StringMultiArrayConstView cv_labels, StringMultiArrayConstView div_labels, StringMultiArrayConstView dsv_labels, StringMultiArrayConstView drv_labels, const StringArray &resp_labels) const

  save correlations to database
- bool correlations_computed () const

  returns corrComputed to indicate whether compute_correlations() has been invoked
- void print_correlations (std::ostream &s, StringMultiArrayConstView cv_labels, StringMultiArrayConstView div_labels, StringMultiArrayConstView dsv_labels, StringMultiArrayConstView drv_labels, const StringArray &resp_labels) const

  prints the correlations computed in compute_correlations()

Private Member Functions

- void simple_corr (RealMatrix &total_data, bool rank_on, const int &num_in)

  computes simple correlations
- void partial_corr (RealMatrix &total_data, bool rank_on, const int &num_in)

  computes partial correlations

Static Private Member Functions

- static bool rank_sort (const int &x, const int &y)

  sort algorithm to compute ranks for rank correlations

Private Attributes

- RealMatrix simpleCorr

  matrix to hold simple raw correlations
- RealMatrix simpleRankCorr

  matrix to hold simple rank correlations
- RealMatrix partialCorr

  matrix to hold partial raw correlations
- RealMatrix partialRankCorr

  matrix to hold partial rank correlations
- size_t numFns

  number of responses

- size_t numVars

  number of inputs
- bool numericalIssuesRaw

  flag indicating numerical issues in partial raw correlation calculations
- bool numericalIssuesRank

  flag indicating numerical issues in partial rank correlation calculations
- bool corrComputed

  flag indictaing whether correlations have been computed

Static Private Attributes

- static RealArray rawData = RealArray()

  array to hold temporary data before sort

13.146.1 Detailed Description

Class for a utility class containing correlation calculations and variance-based decomposition.

This class provides code for several of the sampling methods both in the NonD branch and in the PStudyDACE branch. Currently, the utility functions provide global sensitivity analysis through correlation calculations (e.g. simple, partial, rank, raw) as well as variance-based decomposition.

13.146.2 Member Function Documentation

void compute_correlations (const VariablesArray & *vars_samples,* const IntResponseMap & *resp_samples,* const StringSetArray & *dss_vals*)

computes four correlation matrices for input and output data simple, partial, simple rank, and partial rank

This version is used when full variables objects are being processed

References Dakota::abort_handler(), Variables::continuous_variable(), SensAnalysisGlobal::corrComputed, Variables::discrete_int_variable(), Variables::discrete_real_variable(), Variables::discrete_string_variable(), Response::function_value(), SensAnalysisGlobal::numFns, SensAnalysisGlobal::numVars, SensAnalysisGlobal::partial_corr(), SensAnalysisGlobal::rank_sort(), SensAnalysisGlobal::rawData, Dakota::set_value_to_index(), and SensAnalysisGlobal::simple_corr().

Referenced by NonDSampling::compute_statistics(), FSUDesignCompExp::post_run(), ParamStudy::post_run(), and DDACEDesignCompExp::post_run().

void compute_correlations (const RealMatrix & *vars_samples,* const IntResponseMap & *resp_samples*)

computes four correlation matrices for input and output data simple, partial, simple rank, and partial rank

This version is used when compact samples matrix is being processed

References Dakota::abort_handler(), SensAnalysisGlobal::corrComputed, Response::function_value(), SensAnalysisGlobal::numFns, SensAnalysisGlobal::numVars, SensAnalysisGlobal::partial_corr(), SensAnalysisGlobal::rank_sort(), SensAnalysisGlobal::rawData, and SensAnalysisGlobal::simple_corr().

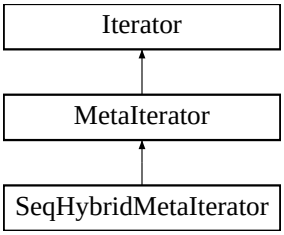
The documentation for this class was generated from the following files:

- SensAnalysisGlobal.hpp
- SensAnalysisGlobal.cpp

13.147 SeqHybridMetaIterator Class Reference

Method for sequential hybrid iteration using multiple optimization and nonlinear least squares methods on multiple models of varying fidelity.

Inheritance diagram for SeqHybridMetaIterator:



Public Member Functions

- SeqHybridMetaIterator (ProblemDescDB &problem_db)

  standard constructor
- SeqHybridMetaIterator (ProblemDescDB &problem_db, Model &model)

  alternate constructor
- ~SeqHybridMetaIterator ()

  destructor

Protected Member Functions

- void core_run ()

  Performs the hybrid iteration by executing a sequence of iterators, using a similar sequence of models that may vary in fidelity.
- void print_results (std::ostream &s)

  print the final iterator results
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- const Variables & variables_results () const

  return the final solution from selectedIterators (variables)
- const Response & response_results () const

  return the final solution from selectedIterators (response)
- void initialize_iterator (int job_index)

  used by IteratorScheduler to set the starting data for a run
- void pack_parameters_buffer (MPIPackBuffer &send_buffer, int job_index)

  used by IteratorScheduler to pack starting data for an iterator run
- void unpack_parameters_initialize (MPIUnpackBuffer &recv_buffer)

used by IteratorScheduler to unpack starting data and initialize an iterator run

- void pack_results_buffer (MPIPackBuffer &send_buffer, int job_index)

  used by IteratorScheduler to pack results data from an iterator run
- void unpack_results_buffer (MPIUnpackBuffer &recv_buffer, int job_index)

  used by IteratorScheduler to unpack results data from an iterator run
- void update_local_results (int job_index)

  used by IteratorScheduler to update local results arrays

Private Member Functions

- void run_sequential ()

  run a sequential hybrid
- void run_sequential_adaptive ()

  run a sequential adaptive hybrid
- void partition_sets (size_t num_sets, int job_index, size_t &start_index, size_t &job_size)

  convert num_sets and job_index into a start_index and job_size for extraction from parameterSets
- void extract_parameter_sets (int job_index, VariablesArray &partial_param_sets)

  extract partial_param_sets from parameterSets based on job_index
- void update_local_results (PRPArray &prp_results, int job_id)

  update the partial set of final results from the local iterator execution
- void initialize_iterator (const VariablesArray &param_sets)

  called by unpack_parameters_initialize(MPIUnpackBuffer) and initialize_iterator(int) to update the active Model and Iterator

Private Attributes

- String seqHybridType

  empty (default) or "adaptive"
- StringArray methodList

  the list of method name identifiers
- bool lightwtCtor

  use of lightweight Iterator construction by name
- IteratorArray selectedIterators

  the set of iterators, one for each entry in methodList
- ModelArray selectedModels

  the set of models, one for each iterator (if not lightweight construction)
- size_t seqCount

  hybrid sequence counter: 0 to numIterators-1
- Real progressThreshold

  when the progress metric falls below this threshold, the sequential adaptive hybrid switches to the next method
- PRP2DArray prpResults

  2-D array of results corresponding to numIteratorJobs, one set of results per job (iterators may return multiple final solutions)
- VariablesArray parameterSets

  1-D array of variable starting points for the iterator jobs

Friends

- class IteratorScheduler

  protect scheduler callback functions from general access

Additional Inherited Members

13.147.1 Detailed Description

Method for sequential hybrid iteration using multiple optimization and nonlinear least squares methods on multiple models of varying fidelity.

Sequential hybrid meta-iteration supports two approaches: (1) the non-adaptive sequential hybrid runs one method to completion, passes its best results as the starting point for a subsequent method, and continues this succession until all methods have been executed (the stopping rules are controlled internally by each iterator), and (2) the adaptive sequential hybrid uses adaptive stopping rules for the iterators that are controlled externally by this method. Any iterator may be used so long as it defines the notion of a final solution which can be passed as starting data for subsequent iterators.

13.147.2 Member Function Documentation

void print_results (std::ostream & *s*) [protected], [virtual]

print the final iterator results

This virtual function provides additional iterator-specific final results outputs beyond the function evaluation summary printed in finalize_run().

Reimplemented from Iterator.

References Response::function_values(), Response::is_null(), Variables::is_null(), MetaIterator::iterSched, IteratorScheduler::messagePass, and SeqHybridMetaIterator::prpResults.

void run_sequential ( ) [private]

run a sequential hybrid

In the sequential nonadaptive case, there is no interference with the iterators. Each runs until its own convergence criteria is satisfied. Status: fully operational.

References Iterator::accepts_multiple_points(), ParallelLibrary::bcast(), ParallelLibrary::bcast_hs(), Iterator::initialize_graphics(), Model::interface_id(), Iterator::iteratedModel, IteratorScheduler::iterator_message_lengths(), IteratorScheduler::iteratorCommRank, IteratorScheduler::iteratorCommSize, IteratorScheduler::iteratorScheduling, IteratorScheduler::iteratorServerId, MetaIterator::iterSched, SeqHybridMetaIterator::lightwtCtor, IteratorScheduler::messagePass, SeqHybridMetaIterator::methodList, Iterator::methodPCIter, IteratorScheduler::miPLIndex, Iterator::num_final_solutions(), IteratorScheduler::numIteratorJobs, IteratorScheduler::numIteratorServers, SeqHybridMetaIterator::pack_parameters_buffer(), Iterator::parallelLib, SeqHybridMetaIterator::parameterSets, SeqHybridMetaIterator::prpResults, ParallelLibrary::recv(), Iterator::response_results(), IteratorScheduler::schedule_iterators(), SeqHybridMetaIterator::selectedIterators, SeqHybridMetaIterator::selectedModels, ParallelLibrary::send(), SeqHybridMetaIterator::seqCount, MPIPackBuffer::size(), Iterator::summaryOutputFlag, and Iterator::variables_results().

Referenced by SeqHybridMetaIterator::core_run().

void run_sequential_adaptive ( ) [private]

run a sequential adaptive hybrid

In the sequential adaptive case, there is interference with the iterators through the use of the ++ overloaded operator. iterator++ runs the iterator for one cycle, after which a progress_metric is computed. This progress metric is used to dictate method switching instead of each iterator's internal convergence criteria. Status: incomplete.

References Iterator::finalize_run(), Iterator::initialize_graphics(), Iterator::initialize_run(), IteratorScheduler::iteratorCommRank, IteratorScheduler::iteratorServerId, MetaIterator::iterSched, SeqHybridMetaIterator::methodList, IteratorScheduler::numIteratorServers, SeqHybridMetaIterator::progressThreshold, Iterator::response_results(), SeqHybridMetaIterator::selectedIterators, SeqHybridMetaIterator::selectedModels, SeqHybridMetaIterator::seqCount, Iterator::summaryOutputFlag, and Iterator::variables_results().

Referenced by SeqHybridMetaIterator::core_run().

void extract_parameter_sets (int *job_index,* VariablesArray & *partial_param_sets*) `[inline], [private]`

extract partial_param_sets from parameterSets based on job_index

This convenience function is executed on an iterator master (static scheduling) or a meta-iterator master (self scheduling) at run initialization time and has access to the full parameterSets array (this is All-Reduced for all peers at the completion of each cycle in run_sequential()).

References SeqHybridMetaIterator::parameterSets, and SeqHybridMetaIterator::partition_sets().

Referenced by SeqHybridMetaIterator::initialize_iterator(), and SeqHybridMetaIterator::pack_parameters_buffer().

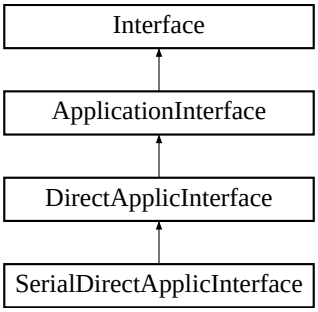
The documentation for this class was generated from the following files:

- SeqHybridMetaIterator.hpp
- SeqHybridMetaIterator.cpp

13.148 SerialDirectApplicInterface Class Reference

Sample derived interface class for testing serial simulator plug-ins using assign_rep().

Inheritance diagram for SerialDirectApplicInterface:



Public Member Functions

- SerialDirectApplicInterface (const Dakota::ProblemDescDB &problem_db)

  constructor
- ~SerialDirectApplicInterface ()

  destructor

Protected Member Functions

- int derived_map_ac (const Dakota::String &ac_name)

  execute an analysis code portion of a direct evaluation invocation
- void derived_map_asynch (const Dakota::ParamResponsePair &pair)

  no-op hides base error; job batching occurs within wait_local_evaluations()
- void wait_local_evaluations (Dakota::PRPQueue &prp_queue)

  evaluate the batch of jobs contained in prp_queue
- void test_local_evaluations (Dakota::PRPQueue &prp_queue)

  invokes wait_local_evaluations() (no special nowait support)
- void set_communicators_checks (int max_eval_concurrency)

  no-op hides default run-time error checks at DirectApplicInterface level

Private Member Functions

- int rosenbrock (const Dakota::RealVector &c_vars, short asv, Dakota::Real &fn_val, Dakota::RealVector &fn_grad, Dakota::RealSymMatrix &fn_hess)

  Rosenbrock plug-in test function.

Additional Inherited Members

13.148.1 Detailed Description

Sample derived interface class for testing serial simulator plug-ins using assign_rep().

The plug-in SerialDirectApplicInterface resides in namespace SIM and uses a copy of rosenbrock() to perform serial parameter to response mappings. It is used to demonstrate plugging in a serial direct analysis driver into Dakota in library mode. Test input files can then use an analysis_driver of "plugin_rosenbrock".

13.148.2 Member Function Documentation

void test_local_evaluations (Dakota::PRPQueue & *prp_queue*) `[inline], [protected]`

invokes wait_local_evaluations() (no special nowait support)

For use by ApplicationInterface::serve_evaluations_asynch(), which can provide a batch processing capability within message passing schedulers (called using chain IteratorScheduler::run_iterator() –> Model::serve() –> ApplicationInterface::serve_evaluations() –> ApplicationInterface::serve_evaluations_asynch()).

References SerialDirectApplicInterface::wait_local_evaluations().

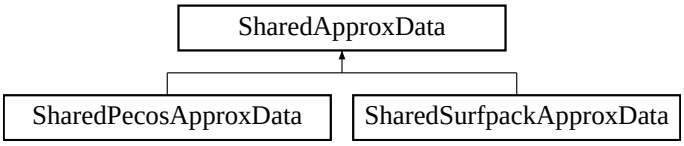
The documentation for this class was generated from the following files:

- PluginSerialDirectApplicInterface.hpp
- PluginSerialDirectApplicInterface.cpp

13.149 SharedApproxData Class Reference

Base class for the shared approximation data class hierarchy.

Inheritance diagram for SharedApproxData:



Public Member Functions

- SharedApproxData ()

  default constructor
- SharedApproxData (ProblemDescDB &problem_db, size_t num_vars)

  standard constructor for envelope
- SharedApproxData (const String &approx_type, const UShortArray &approx_order, size_t num_vars, short data_order, short output_level)

  alternate constructor for envelope
- SharedApproxData (const SharedApproxData &approx)

  copy constructor
- virtual ~SharedApproxData ()

  destructor
- SharedApproxData operator= (const SharedApproxData &approx)

  assignment operator
- virtual void build ()

  builds the shared approximation data from scratch
- virtual void rebuild ()

  rebuilds the shared approximation data incrementally
- virtual void pop (bool save_surr_data)

  back out the previous increment to the shared approximation data
- virtual bool restore_available ()

  queries availability of restoration for trial set
- virtual size_t restoration_index ()

  return index of trial set within restorable bookkeeping sets
- virtual void pre_restore ()

  restore a previous state of the shared approximation data
- virtual void post_restore ()

  clean up saved storage following restoration
- virtual size_t finalization_index (size_t i)

  return index of i-th trailing trial set within restorable bookkeeping sets
- virtual void pre_finalize ()

  finalize the shared approximation data following a set of increments
- virtual void post_finalize ()

  clean up saved storage following aggregation
- virtual void store ()

  store the current state of the shared approximation data for later combination

- virtual void pre_combine (short corr_type)

  aggregate the shared approximation data from current and saved states
- virtual void post_combine (short corr_type)

  clean up saved storage after aggregation
- void set_bounds (const RealVector &c_l_bnds, const RealVector &c_u_bnds, const IntVector &di_l_bnds, const IntVector &di_u_bnds, const RealVector &dr_l_bnds, const RealVector &dr_u_bnds)

  set approximation lower and upper bounds (currently only used by graphics)
- SharedApproxData * data_rep () const

  returns dataRep for access to derived class member functions that are not mapped to the top SharedApproxData level

Protected Member Functions

- SharedApproxData (BaseConstructor, ProblemDescDB &problem_db, size_t num_vars)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
- SharedApproxData (NoDBBaseConstructor, const String &approx_type, size_t num_vars, short data_order, short output_level)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

Protected Attributes

- size_t numVars

  number of variables in the approximation
- String approxType

  approximation type identifier
- short buildDataOrder

  order of the data used for surrogate construction, in ActiveSet request vector 3-bit format.
- short outputLevel

  output verbosity level: {SILENT,QUIET,NORMAL,VERBOSE,DEBUG}_OUTPUT
- RealVector approxCLowerBnds

  approximation continuous lower bounds (used by 3D graphics and Surfpack KrigingModel)
- RealVector approxCUpperBnds

  approximation continuous upper bounds (used by 3D graphics and Surfpack KrigingModel)
- IntVector approxDILowerBnds

  approximation continuous lower bounds
- IntVector approxDIUpperBnds

  approximation continuous upper bounds
- RealVector approxDRLowerBnds

  approximation continuous lower bounds
- RealVector approxDRUpperBnds

  approximation continuous upper bounds

Private Member Functions

- SharedApproxData * get_shared_data (ProblemDescDB &problem_db, size_t num_vars)

  Used only by the standard envelope constructor to initialize dataRep to the appropriate derived type.
- SharedApproxData * get_shared_data (const String &approx_type, const UShortArray &approx_order, size_t num_vars, short data_order, short output_level)

  Used only by the alternate envelope constructor to initialize dataRep to the appropriate derived type.

Private Attributes

- SharedApproxData * dataRep

  pointer to the letter (initialized only for the envelope)
- int referenceCount

  number of objects sharing dataRep

Friends

- class **Approximation**
- class **TaylorApproximation**
- class **TANA3Approximation**
- class **GaussProcApproximation**
- class **SurfpackApproximation**
- class **PecosApproximation**

13.149.1 Detailed Description

Base class for the shared approximation data class hierarchy.

The SharedApproxData class is the base class for the shared approximation data class hierarchy in DAKOTA. For memory efficiency and enhanced polymorphism, the approximation hierarchy employs the "letter/envelope idiom" (see Coplien "Advanced C++", p. 133), for which the base class (SharedApproxData) serves as the envelope and one of the derived classes (selected in SharedApproxData::get_shared_data()) serves as the letter.

13.149.2 Constructor & Destructor Documentation

SharedApproxData ( )

default constructor

For the default constructor, dataRep is NULL. This makes it necessary to check for NULL in the copy constructor, assignment operator, and destructor.

Referenced by SharedApproxData::get_shared_data().

SharedApproxData (ProblemDescDB & *problem_db*, size_t *num_vars*)

standard constructor for envelope

Envelope constructor only needs to extract enough data to properly execute get_shared_data, since SharedApproxData(BaseConstructor, problem_db) builds the actual base class data for the derived approximations.

References Dakota::abort_handler(), SharedApproxData::dataRep, and SharedApproxData::get_shared_data().

SharedApproxData (const String & *approx_type,* const UShortArray & *approx_order,* size_t *num_vars,* short *data_order,* short *output_level*)

alternate constructor for envelope

This is the alternate envelope constructor for instantiations on the fly. Since it does not have access to problem_db, it utilizes the NoDBBaseConstructor constructor chain.

References Dakota::abort_handler(), SharedApproxData::dataRep, and SharedApproxData::get_shared_data().

SharedApproxData (const SharedApproxData & *shared_data*)

copy constructor

Copy constructor manages sharing of dataRep and incrementing of referenceCount.

References SharedApproxData::dataRep, and SharedApproxData::referenceCount.

~SharedApproxData ( ) `[virtual]`

destructor

Destructor decrements referenceCount and only deletes dataRep when referenceCount reaches zero.

References SharedApproxData::dataRep, and SharedApproxData::referenceCount.

SharedApproxData (BaseConstructor , ProblemDescDB & *problem_db,* size_t *num_vars*) `[protected]`

constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor is the one which must build the base class data for all derived classes. get_shared_data() instantiates a derived class letter and the derived constructor selects this base class constructor in its initialization list (to avoid recursion in the base class constructor calling get_shared_data() again). Since the letter IS the representation, its rep pointer is set to NULL (an uninitialized pointer causes problems in ~SharedApproxData).

References SharedApproxData::approxType, SharedApproxData::buildDataOrder, ProblemDescDB::get_bool(), ProblemDescDB::get_db_model_node(), ProblemDescDB::get_string(), ProblemDescDB::set_db_model_nodes(), Dakota::strbegins(), and Dakota::strends().

SharedApproxData (NoDBBaseConstructor , const String & *approx_type,* size_t *num_vars,* short *data_order,* short *output_level*) `[protected]`

constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

This constructor is the one which must build the base class data for all derived classes. get_shared_data() instantiates a derived class letter and the derived constructor selects this base class constructor in its initialization list (to avoid recursion in the base class constructor calling get_shared_data() again). Since the letter IS the representation, its rep pointer is set to NULL (an uninitialized pointer causes problems in ~SharedApproxData).

References SharedApproxData::approxType, SharedApproxData::buildDataOrder, Dakota::strbegins(), and Dakota::strends().

13.149.3 Member Function Documentation

SharedApproxData operator= (const SharedApproxData & *shared_data*)

assignment operator

Assignment operator decrements referenceCount for old dataRep, assigns new dataRep, and increments referenceCount for new dataRep.

References SharedApproxData::dataRep, and SharedApproxData::referenceCount.

SharedApproxData * get_shared_data (ProblemDescDB & *problem_db*, size_t *num_vars*) [private]

Used only by the standard envelope constructor to initialize dataRep to the appropriate derived type.

Used only by the envelope constructor to initialize dataRep to the appropriate derived type.

References ProblemDescDB::get_string(), SharedApproxData::SharedApproxData(), and Dakota::strends().

Referenced by SharedApproxData::SharedApproxData().

SharedApproxData * get_shared_data (const String & *approx_type*, const UShortArray & *approx_order*, size_t *num_vars*, short *data_order*, short *output_level*) [private]

Used only by the alternate envelope constructor to initialize dataRep to the appropriate derived type.

Used only by the envelope constructor to initialize dataRep to the appropriate derived type.

References SharedApproxData::SharedApproxData(), and Dakota::strends().

13.149.4 Member Data Documentation

short buildDataOrder [protected]

order of the data used for surrogate construction, in ActiveSet request vector 3-bit format.

This setting distinguishes derivative data intended for use in construction (includes derivatives w.r.t. the build variables) from derivative data that may be approximated separately (excludes derivatives w.r.t. auxilliary variables). This setting should also not be inferred directly from the responses specification, since we may need gradient support for evaluating gradients at a single point (e.g., the center of a trust region), but not require gradient evaluations at every point.

Referenced by SharedSurfpackApproxData::add_sd_to_surfdata(), TaylorApproximation::build(), TaylorApproximation::gradient(), TaylorApproximation::hessian(), TaylorApproximation::min_coefficients(), Approximation::min_points(), Approximation::recommended_points(), SharedApproxData::SharedApproxData(), SharedPecosApproxData::SharedPecosApproxData(), SurfpackApproximation::SurfpackApproximation(), SurfpackApproximation::surrogates_to_surf_data(), TANA3Approximation::TANA3Approximation(), and TaylorApproximation::value().

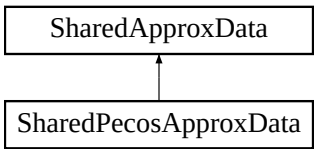
The documentation for this class was generated from the following files:

- SharedApproxData.hpp
- SharedApproxData.cpp

13.150 SharedPecosApproxData Class Reference

Derived approximation class for global basis polynomials.

Inheritance diagram for SharedPecosApproxData:



Public Member Functions

- SharedPecosApproxData ()

  default constructor
- SharedPecosApproxData (const String &approx_type, const UShortArray &approx_order, size_t num_vars, short data_order, short output_level)

  alternate constructor
- SharedPecosApproxData (ProblemDescDB &problem_db, size_t num_vars)

  standard ProblemDescDB-driven constructor
- ~SharedPecosApproxData ()

  destructor
- void random_variables_key (const Pecos::BitArray &random_vars_key)

  set pecosBasisApprox.randomVarsKey
- void integration_iterator (const Iterator &iterator)

  set pecosBasisApprox.driverRep
- void construct_basis (const Pecos::ShortArray &u_types, const Pecos::AleatoryDistParams &adp)

  invoke Pecos::SharedOrthogPolyApproxData::construct_basis()
- void polynomial_basis (const std::vector< Pecos::BasisPolynomial > &poly_basis)

  set Pecos::SharedOrthogPolyApproxData::polynomialBasis
- const std::vector
  < Pecos::BasisPolynomial > & polynomial_basis () const

  get Pecos::SharedOrthogPolyApproxData::polynomialBasis
- void allocate (const UShort2DArray &mi)

  set Pecos::SharedOrthogPolyApproxData::multiIndex and allocate associated arrays
- const UShort2DArray & multi_index () const

  get Pecos::SharedOrthogPolyApproxData::multiIndex
- const Pecos::BitArrayULongMap & sobol_index_map () const

  return Pecos::SharedPolyApproxData::sobolIndexMap
- void coefficients_norms_flag (bool flag)

  invoke Pecos::SharedOrthogPolyApproxData::coefficients_norms_flag()
- size_t expansion_terms () const

  return Pecos::SharedOrthogPolyApproxData::expansion_terms()
- const UShortArray & expansion_order () const

  return Pecos::SharedOrthogPolyApproxData::expansion_order()
- void expansion_order (const UShortArray &order)

  invokes Pecos::SharedOrthogPolyApproxData::expansion_order(UShortArray&)
- void increment_order ()

  invokes Pecos::SharedOrthogPolyApproxData::increment_order()
- void configuration_options (const Pecos::ExpansionConfigOptions &ec_options)

  set the expansion configuration options within Pecos::SharedPolyApproxData
- void configuration_options (const Pecos::BasisConfigOptions &bc_options)

  set the basis configuration options within Pecos::SharedPolyApproxData
- void configuration_options (const Pecos::RegressionConfigOptions &rc_options)

  set the regression configuration options within Pecos::SharedRegressOrthogPolyApproxData

Protected Member Functions

- void build ()

  builds the shared approximation data from scratch
- void rebuild ()

  rebuilds the shared approximation data incrementally
- void pop (bool save_surr_data)

  back out the previous increment to the shared approximation data
- bool restore_available ()

  queries availability of restoration for trial set
- size_t restoration_index ()

  return index of trial set within restorable bookkeeping sets
- void pre_restore ()

  restore a previous state of the shared approximation data
- void post_restore ()

  clean up saved storage following restoration
- size_t finalization_index (size_t i)

  return index of i-th trailing trial set within restorable bookkeeping sets
- void pre_finalize ()

  finalize the shared approximation data following a set of increments
- void post_finalize ()

  clean up saved storage following aggregation
- void store ()

  store the current state of the shared approximation data for later combination
- void pre_combine (short corr_type)

  aggregate the shared approximation data from current and saved states
- void post_combine (short corr_type)

  clean up saved storage after aggregation

Private Member Functions

- Pecos::SharedBasisApproxData & pecos_shared_data ()

  return pecosSharedData
- void approx_type_to_basis_type (const String &approx_type, short &basis_type)

  utility to convert Dakota type string to Pecos type enumeration

Private Attributes

- Pecos::SharedBasisApproxData pecosSharedData

  the Pecos shared approximation data
- Pecos::SharedPolyApproxData * pecosSharedDataRep

  convenience pointer to derived letter within pecosSharedData

Friends

- class **PecosApproximation**

Additional Inherited Members

13.150.1 Detailed Description

Derived approximation class for global basis polynomials.

The SharedPecosApproxData class provides a global approximation based on basis polynomials. This includes orthogonal polynomials used for polynomial chaos expansions and interpolation polynomials used for stochastic collocation.

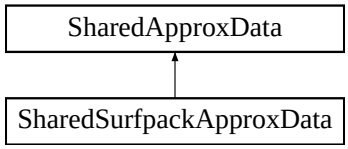
The documentation for this class was generated from the following files:

- SharedPecosApproxData.hpp
- SharedPecosApproxData.cpp

13.151 SharedSurfpackApproxData Class Reference

Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota.

Inheritance diagram for SharedSurfpackApproxData:



Public Member Functions

- SharedSurfpackApproxData ()

  default constructor
- SharedSurfpackApproxData (const String &approx_type, const UShortArray &approx_order, size_t num_vars, short data_order, short output_level)

  alternate constructor
- SharedSurfpackApproxData (ProblemDescDB &problem_db, size_t num_vars)

  standard constructor: Surfpack surface of appropriate type will be created
- ~SharedSurfpackApproxData ()

  destructor

Private Member Functions

- void add_sd_to_surfdata (const Pecos::SurrogateDataVars &sdv, const Pecos::SurrogateDataResp &sdr, short fail_code, SurfData &surf_data)

  add Pecos::SurrogateData::SurrogateData{Vars,Resp} to SurfData, accounting for buildDataOrder available
- void copy_matrix (const RealSymMatrix &rsm, SurfpackMatrix< Real > &surfpack_matrix)

  copy RealSymMatrix to SurfpackMatrix (Real type only)
- void merge_variable_arrays (const RealVector &cv, const IntVector &div, const RealVector &drv, RealArray &ra)

  merge cv, div, and drv vectors into a single ra array
- void sdv_to_realarray (const Pecos::SurrogateDataVars &sdv, RealArray &ra)

aggregate {continuous,discrete int,discrete real} variables from SurrogateDataVars into ra

- void vars_to_realarray (const Variables &vars, RealArray &ra)

  aggregate {active,all} {continuous,discrete int,discrete real} variables into ra

Private Attributes

- unsigned short approxOrder

  order of polynomial approximation
- String exportModelName

  A Surfpack model name for saving the surrogate model.
- StringArray diagnosticSet

  set of diagnostic metrics
- bool crossValidateFlag

  whether to perform cross validation
- unsigned numFolds

  number of folds for CV
- Real percentFold

  percentage of data for CV
- bool pressFlag

  whether to perform PRESS

Friends

- class **SurfpackApproximation**

Additional Inherited Members

13.151.1 Detailed Description

Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota.

The SharedSurfpackApproxData class is the interface between Dakota and Surfpack. Based on the information in the ProblemDescDB that is passed in through the constructor, SharedSurfpackApproxData builds a Surfpack Surface object that corresponds to one of the following data-fitting techniques: polynomial regression, kriging, artificial neural networks, radial basis function network, or multivariate adaptaive regression splines (MARS).

13.151.2 Constructor & Destructor Documentation

SharedSurfpackApproxData (const String & *approx_type,* const UShortArray & *approx_order,* size_t *num_vars,* short *data_order,* short *output_level*)

alternate constructor

On-the-fly constructor which uses mostly Surfpack model defaults.

References Dakota::abort_handler(), SharedSurfpackApproxData::approxOrder, and SharedApproxData::approxType.

SharedSurfpackApproxData (ProblemDescDB & *problem_db*, size_t *num_vars*)

standard constructor: Surfpack surface of appropriate type will be created

Initialize the embedded Surfpack surface object and configure it using the specifications from the input file. Data for the surface is created later.

References SharedSurfpackApproxData::approxOrder, SharedApproxData::approxType, ProblemDescDB::get_short(), and ProblemDescDB::get_string().

The documentation for this class was generated from the following files:

- SharedSurfpackApproxData.hpp
- SharedSurfpackApproxData.cpp

13.152 SharedVariablesData Class Reference

Container class encapsulating variables data that can be shared among a set of Variables instances.

Public Member Functions

- SharedVariablesData ()

  default constructor
- SharedVariablesData (const ProblemDescDB &problem_db, const std::pair< short, short > &view)

  standard constructor
- SharedVariablesData (const std::pair< short, short > &view, const std::map< unsigned short, size_t > &vars_comps, const BitArray &all_relax_di=BitArray(), const BitArray &all_relax_dr=BitArray())

  medium weight constructor providing detailed variable counts
- SharedVariablesData (const std::pair< short, short > &view, const SizetArray &vars_comps_totals, const BitArray &all_relax_di=BitArray(), const BitArray &all_relax_dr=BitArray())

  lightweight constructor providing variable count totals
- SharedVariablesData (const SharedVariablesData &svd)

  copy constructor
- ~SharedVariablesData ()

  destructor
- SharedVariablesData & operator= (const SharedVariablesData &svd)

  assignment operator
- SharedVariablesData copy () const

  create a deep copy of the current object and return by value
- void all_counts (size_t &num_acv, size_t &num_adiv, size_t &num_adsv, size_t &num_adrv) const

  compute all variables sums from SharedVariablesDataRep::variablesCompsTotals and SharedVariablesDataRep::allRelaxedDiscrete{Int,Real}
- void design_counts (size_t &num_cdv, size_t &num_ddiv, size_t &num_ddsv, size_t &num_ddrv) const

  compute design variables sums from SharedVariablesDataRep::variablesCompsTotals and SharedVariablesDataRep::allRelaxedDiscrete{Int,Real}
- void aleatory_uncertain_counts (size_t &num_cauv, size_t &num_dauiv, size_t &num_dausv, size_t &num_daurv) const

  compute aleatory uncertain variables sums from SharedVariablesDataRep::variablesCompsTotals and SharedVariablesDataRep::allRelaxedDiscrete{Int,Real}

- void epistemic_uncertain_counts (size_t &num_ceuv, size_t &num_deuiv, size_t &num_deusv, size_t &num_deurv) const

  compute epistemic uncertain variables sums from SharedVariablesDataRep::variablesCompsTotals and SharedVariablesDataRep::allRelaxedDiscrete{Int,Real}
- void uncertain_counts (size_t &num_cuv, size_t &num_duiv, size_t &num_dusv, size_t &num_durv) const

  compute uncertain variables sums from SharedVariablesDataRep::variablesCompsTotals and SharedVariablesDataRep::allRelaxedDiscrete{Int,Real}
- void state_counts (size_t &num_csv, size_t &num_dsiv, size_t &num_dssv, size_t &num_dsrv) const

  compute state variables sums from SharedVariablesDataRep::variablesCompsTotals and SharedVariablesDataRep::allRelaxedDiscrete{Int,Real}
- void initialize_active_start_counts ()

  initialize start index and counts for active variables
- void initialize_inactive_start_counts ()

  initialize start index and counts for inactive variables
- void initialize_active_components ()

  initialize the active components totals given active variable counts
- void initialize_inactive_components ()

  initialize the inactive components totals given inactive variable counts
- const BitArray & all_relaxed_discrete_int () const

  return allRelaxedDiscreteInt
- const BitArray & all_relaxed_discrete_real () const

  return allRelaxedDiscreteReal
- StringMultiArrayView all_continuous_labels (size_t start, size_t num_items) const

  get num_items continuous labels beginning at index start
- void all_continuous_labels (StringMultiArrayConstView cv_labels, size_t start, size_t num_items)

  set num_items continuous labels beginning at index start
- void all_continuous_label (const String &cv_label, size_t index)

  set continuous label at index start
- StringMultiArrayView all_discrete_int_labels (size_t start, size_t num_items) const

  get num_items discrete integer labels beginning at index start
- void all_discrete_int_labels (StringMultiArrayConstView div_labels, size_t start, size_t num_items)

  set num_items discrete integer labels beginning at index start
- void all_discrete_int_label (const String &div_label, size_t index)

  set discrete integer label at index start
- StringMultiArrayView all_discrete_string_labels (size_t start, size_t num_items) const

  get num_items discrete string labels beginning at index start
- void all_discrete_string_labels (StringMultiArrayConstView dsv_labels, size_t start, size_t num_items)

  set num_items discrete string labels beginning at index start
- void all_discrete_string_label (const String &dsv_label, size_t index)

  set discrete string label at index start
- StringMultiArrayView all_discrete_real_labels (size_t start, size_t num_items) const

  get num_items discrete real labels beginning at index start
- void all_discrete_real_labels (StringMultiArrayConstView drv_labels, size_t start, size_t num_items)

  set num_items discrete real labels beginning at index start

- void all_discrete_real_label (const String &drv_label, size_t index)

  set discrete real label at index start
- UShortMultiArrayConstView all_continuous_types (size_t start, size_t num_items) const

  get num_items continuous types beginning at index start
- void all_continuous_types (UShortMultiArrayConstView cv_types, size_t start, size_t num_items)

  set num_items continuous types beginning at index start
- void all_continuous_type (unsigned short cv_type, size_t index)

  set continuous type at index
- UShortMultiArrayConstView all_discrete_int_types (size_t start, size_t num_items) const

  get num_items discrete integer types beginning at index start
- void all_discrete_int_types (UShortMultiArrayConstView div_types, size_t start, size_t num_items)

  set num_items discrete integer types beginning at index start
- void all_discrete_int_type (unsigned short div_type, size_t index)

  set discrete integer type at index
- UShortMultiArrayConstView all_discrete_string_types (size_t start, size_t num_items) const

  get num_items discrete string types beginning at index start
- void all_discrete_string_types (UShortMultiArrayConstView dsv_types, size_t start, size_t num_items)

  set num_items discrete string types beginning at index start
- void all_discrete_string_type (unsigned short dsv_type, size_t index)

  set discrete string type at index
- UShortMultiArrayConstView all_discrete_real_types (size_t start, size_t num_items) const

  get num_items discrete real types beginning at index start
- void all_discrete_real_types (UShortMultiArrayConstView drv_types, size_t start, size_t num_items)

  set num_items discrete real types beginning at index start
- void all_discrete_real_type (unsigned short drv_type, size_t index)

  set discrete real type at index
- SizetMultiArrayConstView all_continuous_ids (size_t start, size_t num_items) const

  get num_items continuous ids beginning at index start
- void all_continuous_ids (SizetMultiArrayConstView cv_ids, size_t start, size_t num_items)

  set num_items continuous ids beginning at index start
- void all_continuous_id (size_t id, size_t index)

  set num_items continuous ids beginning at index start
- const String & id () const

  return the user-provided or default Variables identifier
- const SizetArray & components_totals () const

  return variable type counts for {continuous,discrete integer,discrete real} {design,aleatory uncertain,epistemic uncertain,state}
- const SizetArray & active_components_totals () const

  return active variable type counts for {continuous,discrete integer,discrete real} {design,aleatory uncertain,epistemic uncertain,state}
- const SizetArray & inactive_components_totals () const

  return inactive variable type counts for {continuous,discrete integer,discrete real} {design,aleatory uncertain,epistemic uncertain,state}
- size_t vc_lookup (unsigned short key) const

retrieve the variables type count within svdRep->variablesComponents corresponding to (a fine-grain variables type) key

- const std::pair< short, short > & view () const

  retreive the Variables view
- void inactive_view (short view2)

  set the inactive Variables view
- size_t cv () const

  get number of active continuous vars
- size_t cv_start () const

  get start index of active continuous vars
- size_t div () const

  get number of active discrete int vars
- size_t div_start () const

  get start index of active discrete int vars
- size_t dsv () const

  get number of active discrete string vars
- size_t dsv_start () const

  get start index of active discrete string vars
- size_t drv () const

  get number of active discrete real vars
- size_t drv_start () const

  get start index of active discrete real vars
- size_t icv () const

  get number of inactive continuous vars
- size_t icv_start () const

  get start index of inactive continuous vars
- size_t idiv () const

  get number of inactive discrete int vars
- size_t idiv_start () const

  get start index of inactive discrete int vars
- size_t idsv () const

  get number of inactive discrete string vars
- size_t idsv_start () const

  get start index of inactive discr string vars
- size_t idrv () const

  get number of inactive discrete real vars
- size_t idrv_start () const

  get start index of inactive discrete real vars
- void cv (size_t ncv)

  set number of active continuous vars
- void cv_start (size_t cvs)

  set start index of active continuous vars
- void div (size_t ndiv)

set number of active discrete int vars

- void div_start (size_t divs)

  set start index of active discrete int vars

- void dsv (size_t ndsv)

  set number of active discrete string vars

- void dsv_start (size_t dsvs)

  set start index of active discr string vars

- void drv (size_t ndrv)

  set number of active discrete real vars

- void drv_start (size_t drvs)

  set start index of active discrete real vars

- void icv (size_t nicv)

  set number of inactive continuous vars

- void icv_start (size_t icvs)

  set start index of inactive continuous vars

- void idiv (size_t nidiv)

  set number of inactive discrete int vars

- void idiv_start (size_t idivs)

  set start index of inactive discr int vars

- void idsv (size_t nidsv)

  set number of inactive discr string vars

- void idsv_start (size_t idsvs)

  set start index of inact discr string vars

- void idrv (size_t nidrv)

  set number of inactive discrete real vars

- void idrv_start (size_t idrvs)

  set start index of inact discr real vars

Private Attributes

- SharedVariablesDataRep * svdRep

  pointer to the body (handle-body idiom)

13.152.1 Detailed Description

Container class encapsulating variables data that can be shared among a set of Variables instances.

An array of Variables objects (e.g., Analyzer::allVariables) contains repeated configuration data (id's, labels, counts). SharedVariablesData employs a handle-body idiom to allow this shared data to be managed in a single object with many references to it, one per Variables object in the array. This allows scaling to larger sample sets.

13.152.2 Member Function Documentation

SharedVariablesData copy ( ) const

create a deep copy of the current object and return by value

Deep copies are used when recasting changes the nature of a Variables set.

References SharedVariablesDataRep::activeVarsCompsTotals, SharedVariablesDataRep::allContinuousIds, SharedVariablesDataRep::allContinuousLabels, SharedVariablesDataRep::allContinuousTypes, SharedVariablesDataRep::allDiscreteIntLabels, SharedVariablesDataRep::allDiscreteIntTypes, SharedVariablesDataRep::allDiscreteRealLabels, SharedVariablesDataRep::allDiscreteRealTypes, SharedVariablesDataRep::allDiscreteStringLabels, SharedVariablesDataRep::allDiscreteStringTypes, SharedVariablesDataRep::allRelaxedDiscreteInt, SharedVariablesDataRep::allRelaxedDiscreteReal, SharedVariablesDataRep::cvStart, SharedVariablesDataRep::divStart, SharedVariablesDataRep::drvStart, SharedVariablesDataRep::dsvStart, SharedVariablesDataRep::icvStart, SharedVariablesDataRep::idivStart, SharedVariablesDataRep::idrvStart, SharedVariablesDataRep::idsvStart, SharedVariablesDataRep::inactiveVarsCompsTotals, SharedVariablesDataRep::numCV, SharedVariablesDataRep::numDIV, SharedVariablesDataRep::numDRV, SharedVariablesDataRep::numDSV, SharedVariablesDataRep::numICV, SharedVariablesDataRep::numIDIV, SharedVariablesDataRep::numIDRV, SharedVariablesDataRep::numIDSV, SharedVariablesData::svdRep, SharedVariablesDataRep::variablesComponents, SharedVariablesDataRep::variablesCompsTotals, SharedVariablesDataRep::variablesId, and SharedVariablesDataRep::variablesView.

The documentation for this class was generated from the following files:

- SharedVariablesData.hpp
- SharedVariablesData.cpp

13.153 SharedVariablesDataRep Class Reference

The representation of a SharedVariablesData instance. This representation, or body, may be shared by multiple SharedVariablesData handle instances.

Private Member Functions

- SharedVariablesDataRep (const ProblemDescDB &problem_db, const std::pair< short, short > &view)

  standard constructor
- SharedVariablesDataRep (const std::pair< short, short > &view, const std::map< unsigned short, size_t > &vars_comps, const BitArray &all_relax_di, const BitArray &all_relax_dr)

  medium weight constructor providing detailed variable counts
- SharedVariablesDataRep (const std::pair< short, short > &view, const SizetArray &vars_comps_totals, const BitArray &all_relax_di, const BitArray &all_relax_dr)

  lightweight constructor providing variable count totals
- SharedVariablesDataRep ()

  default constructor
- ~SharedVariablesDataRep ()

  destructor
- void initialize_components_totals (const ProblemDescDB &problem_db)

  populate variables{Components,CompsTotals} from user variable type and count specifications
- void components_to_totals ()

  update variablesCompsTotals from variablesComponents
- void relax_noncategorical (const ProblemDescDB &problem_db)

populate allRelaxedDiscrete{Int,Real} from user specifications (relax variables that are not declared as categorical)

- void all_counts (size_t &num_acv, size_t &num_adiv, size_t &num_adsv, size_t &num_adrv) const

  compute all variables sums from variablesCompsTotals

- void relax_counts (size_t &num_cv, size_t &num_div, size_t &num_drv, size_t offset_di, size_t offset_dr) const

  adjust counts based on allRelaxedDiscrete{Int,Real}

- void design_counts (size_t &num_cdv, size_t &num_ddiv, size_t &num_ddsv, size_t &num_ddrv) const

  compute design variables sums from variablesCompsTotals

- void aleatory_uncertain_counts (size_t &num_cauv, size_t &num_dauiv, size_t &num_dausv, size_t &num_-daurv) const

  compute aleatory uncertain variables sums from variablesCompsTotals

- void epistemic_uncertain_counts (size_t &num_ceuv, size_t &num_deuiv, size_t &num_deusv, size_t &num-_deurv) const

  compute epistemic uncertain variables sums from variablesCompsTotals

- void uncertain_counts (size_t &num_cuv, size_t &num_duiv, size_t &num_dusv, size_t &num_durv) const

  compute uncertain variables sums from variablesCompsTotals

- void state_counts (size_t &num_csv, size_t &num_dsiv, size_t &num_dssv, size_t &num_dsrv) const

  compute state variables sums from variablesCompsTotals

- void view_start_counts (short view, size_t &cv_start, size_t &div_start, size_t &dsv_start, size_t &drv_start, size_t &num_cv, size_t &num_div, size_t &num_dsv, size_t &num_drv) const

  define start indices and counts for active variables based on view

- void size_all_labels ()

  size all{Continuous,DiscreteInt,DiscreteString,DiscreteReal}Labels, with or without discrete relaxation

- void size_all_types ()

  size all{Continuous,DiscreteInt,DiscreteString,DiscreteReal}Types, with or without discrete relaxation

- void initialize_all_labels (const ProblemDescDB &problem_db)

  aggregate all{Continuous,DiscreteInt,DiscreteString,DiscreteReal}Labels from user specification or defaults

- void initialize_all_types ()

  initialize all{Continuous,DiscreteInt,DiscreteString,DiscreteReal}Types, with or without discrete relaxation

- void initialize_all_ids ()

  initialize allContinuousIds (discrete not currently needed), with or without discrete relaxation

- void initialize_active_start_counts ()

  initialize {c,di,ds,dr}vStart and num{D,DI,DS,DR}V

- void initialize_inactive_start_counts ()

  initialize i{c,di,ds,dr}vStart and numI{D,DI,DS,DR}V

- void initialize_active_components ()

  initialize activeVarsCompsTotals given {c,di,dr}vStart and num{C,DI,DR}V

- void initialize_inactive_components ()

  initialize inactiveVarsCompsTotals given i{c,di,dr}vStart and numI{C,DI,DR}V

- size_t vc_lookup (unsigned short key) const

  retrieve the count within variablesComponents corresponding to key

Private Attributes

- String variablesId

  variables identifier string from the input file
- std::map< unsigned short, size_t > variablesComponents

  map linking variable types to counts
- SizetArray variablesCompsTotals

  totals for variable type counts for {continuous,discrete integer,discrete string,discrete real} {design,aleatory uncertain,epistemic uncertain,state}.
- SizetArray activeVarsCompsTotals

  totals for active variable type counts for {continuous,discrete integer,discrete string,discrete real} {design,aleatory uncertain,epistemic uncertain,state}.
- SizetArray inactiveVarsCompsTotals

  totals for inactive variable type counts for {continuous,discrete integer,discrete string,discrete real} {design,aleatory uncertain,epistemic uncertain,state}.
- std::pair< short, short > variablesView

  the variables view pair containing active (first) and inactive (second) view enumerations
- size_t cvStart

  start index of active continuous variables within allContinuousVars
- size_t divStart

  start index of active discrete integer variables within allDiscreteIntVars
- size_t dsvStart

  start index of active discrete string vars within allDiscreteStringVars
- size_t drvStart

  start index of active discrete real variables within allDiscreteRealVars
- size_t icvStart

  start index of inactive continuous variables within allContinuousVars
- size_t idivStart

  start index of inactive discrete integer vars within allDiscreteIntVars
- size_t idsvStart

  start index of inactive discrete string vars within allDiscreteStringVars
- size_t idrvStart

  start index of inactive discrete real variables within allDiscreteRealVars
- size_t numCV

  number of active continuous variables
- size_t numDIV

  number of active discrete integer variables
- size_t numDSV

  number of active discrete string variables
- size_t numDRV

  number of active discrete real variables
- size_t numICV

  number of inactive continuous variables
- size_t numIDIV

number of inactive discrete integer variables

- size_t numIDSV

  number of inactive discrete string variables

- size_t numIDRV

  number of inactive discrete real variables

- StringMultiArray allContinuousLabels

  array of variable labels for all of the continuous variables

- StringMultiArray allDiscreteIntLabels

  array of variable labels for all of the discrete integer variables

- StringMultiArray allDiscreteStringLabels

  array of variable labels for all of the discrete string variables

- StringMultiArray allDiscreteRealLabels

  array of variable labels for all of the discrete real variables

- UShortMultiArray allContinuousTypes

  array of variable types for all of the continuous variables

- UShortMultiArray allDiscreteIntTypes

  array of variable types for all of the discrete integer variables

- UShortMultiArray allDiscreteStringTypes

  array of variable types for all of the discrete string variables

- UShortMultiArray allDiscreteRealTypes

  array of variable types for all of the discrete real variables

- SizetMultiArray allContinuousIds

  array of 1-based position identifiers for the all continuous variables array

- BitArray allRelaxedDiscreteInt

  array of booleans to indicate relaxation (promotion from DiscreteInt to Continuous) for all specified discrete int variables

- BitArray allRelaxedDiscreteReal

  array of booleans to indicate relaxation (promotion from DiscreteReal to Continuous) for all specified discrete real variables

- int referenceCount

  number of handle objects sharing svdRep

Friends

- class **SharedVariablesData**

13.153.1 Detailed Description

The representation of a SharedVariablesData instance. This representation, or body, may be shared by multiple SharedVariablesData handle instances.

The SharedVariablesData/SharedVariablesDataRep pairs utilize a handle-body idiom (Coplien, Advanced C++).

13.153.2 Constructor & Destructor Documentation

SharedVariablesDataRep (const ProblemDescDB & *problem_db,* const std::pair< short, short > & *view*) [private]

standard constructor

This constructor is the one which must build the base class data for all derived classes. get_variables() instantiates a derived class letter and the derived constructor selects this base class constructor in its initialization list (to avoid the recursion of the base class constructor calling get_variables() again). Since the letter IS the representation, its representation pointer is set to NULL (an uninitialized pointer causes problems in ~Variables).

References SharedVariablesDataRep::initialize_all_ids(), SharedVariablesDataRep::initialize_all_labels(), SharedVariablesDataRep::initialize_all_types(), SharedVariablesDataRep::initialize_components_totals(), and SharedVariablesDataRep::relax_noncategorical().

13.153.3 Member Data Documentation

SizetArray variablesCompsTotals [private]

totals for variable type counts for {continuous,discrete integer,discrete string,discrete real} {design,aleatory uncertain,epistemic uncertain,state}.

This data reflects the variable counts as originally specified and is not altered by relaxation.

Referenced by SharedVariablesDataRep::aleatory_uncertain_counts(), SharedVariablesDataRep::all_counts(), SharedVariablesDataRep::components_to_totals(), SharedVariablesData::components_totals(), SharedVariablesData::copy(), SharedVariablesDataRep::design_counts(), SharedVariablesDataRep::epistemic_uncertain_counts(), SharedVariablesDataRep::initialize_active_components(), SharedVariablesDataRep::initialize_all_ids(), SharedVariablesDataRep::initialize_components_totals(), SharedVariablesDataRep::initialize_inactive_components(), SharedVariablesDataRep::relax_noncategorical(), SharedVariablesDataRep::state_counts(), SharedVariablesDataRep::uncertain_counts(), and SharedVariablesDataRep::view_start_counts().

SizetArray activeVarsCompsTotals [private]

totals for active variable type counts for {continuous,discrete integer,discrete string,discrete real} {design,aleatory uncertain,epistemic uncertain,state}.

This data reflects the variable counts as originally specified and is not altered by relaxation.

Referenced by SharedVariablesData::active_components_totals(), SharedVariablesData::copy(), and SharedVariablesDataRep::initialize_active_components().

SizetArray inactiveVarsCompsTotals [private]

totals for inactive variable type counts for {continuous,discrete integer,discrete string,discrete real} {design,aleatory uncertain,epistemic uncertain,state}.

This data reflects the variable counts as originally specified and is not altered by relaxation.

Referenced by SharedVariablesData::copy(), SharedVariablesData::inactive_components_totals(), and SharedVariablesDataRep::initialize_inactive_components().

SizetMultiArray allContinuousIds [private]

array of 1-based position identifiers for the all continuous variables array

These identifiers define positions of the all continuous variables array within the total variable sequence. A primary use case is for defining derivative ids (DVV) based on an active subset.

Referenced by SharedVariablesData::all_continuous_id(), SharedVariablesData::all_continuous_ids(), SharedVariablesData::copy(), and SharedVariablesDataRep::initialize_all_ids().

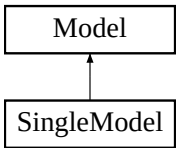
The documentation for this class was generated from the following files:

- SharedVariablesData.hpp
- SharedVariablesData.cpp

13.154 SingleModel Class Reference

Derived model class which utilizes a single interface to map variables into responses.

Inheritance diagram for SingleModel:



Public Member Functions

- SingleModel (ProblemDescDB &problem_db)

  constructor
- ~SingleModel ()

  destructor

Protected Member Functions

- Interface & derived_interface ()

  return userDefinedInterface
- void derived_compute_response (const ActiveSet &set)

  portion of compute_response() specific to SingleModel (invokes a synchronous map() on userDefinedInterface)
- void derived_asynch_compute_response (const ActiveSet &set)

  portion of asynch_compute_response() specific to SingleModel (invokes an asynchronous map() on userDefinedInterface)
- const IntResponseMap & derived_synchronize ()

  portion of synchronize() specific to SingleModel (invokes synch() on userDefinedInterface)
- const IntResponseMap & derived_synchronize_nowait ()

  portion of synchronize_nowait() specific to SingleModel (invokes synch_nowait() on userDefinedInterface)
- void component_parallel_mode (short mode)

  SingleModel only supports parallelism in userDefinedInterface, so this virtual function redefinition is simply a sanity check.
- short local_eval_synchronization ()

  return userDefinedInterface synchronization setting
- int local_eval_concurrency ()

  return userDefinedInterface asynchronous evaluation concurrency
- bool derived_master_overload () const

  flag which prevents overloading the master with a multiprocessor evaluation (request forwarded to userDefinedInterface)

- void derived_init_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set up SingleModel for parallel operations (request forwarded to userDefinedInterface)
- void derived_init_serial ()

  set up SingleModel for serial operations (request forwarded to userDefinedInterface).
- void derived_set_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  set active parallel configuration for the SingleModel (request forwarded to userDefinedInterface)
- void derived_free_communicators (ParLevLIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)

  deallocate communicator partitions for the SingleModel (request forwarded to userDefinedInterface)
- void serve_run (ParLevLIter pl_iter, int max_eval_concurrency)

  Service userDefinedInterface job requests received from the master. Completes when a termination message is received from stop_servers().
- void stop_servers ()

  executed by the master to terminate userDefinedInterface server operations when SingleModel iteration is complete.
- const String & interface_id () const

  return the userDefinedInterface identifier
- int evaluation_id () const

  return the current evaluation id for the SingleModel (request forwarded to userDefinedInterface)
- bool evaluation_cache () const

  return flag indicated usage of an evaluation cache by the SingleModel (request forwarded to userDefinedInterface)
- void set_evaluation_reference ()

  set the evaluation counter reference points for the SingleModel (request forwarded to userDefinedInterface)
- void fine_grained_evaluation_counters ()

  request fine-grained evaluation reporting within the userDefinedInterface
- void print_evaluation_summary (std::ostream &s, bool minimal_header=false, bool relative_count=true) const

  print the evaluation summary for the SingleModel (request forwarded to userDefinedInterface)
- void eval_tag_prefix (const String &eval_id_str)

  set the hierarchical eval ID tag prefix

Private Attributes

- Interface userDefinedInterface

  the interface used for mapping variables to responses

Additional Inherited Members

13.154.1 Detailed Description

Derived model class which utilizes a single interface to map variables into responses.

The SingleModel class is the simplest of the derived model classes. It provides the capabilities of the original Model class, prior to the development of surrogate and nested model extensions. The derived response computation and synchronization functions utilize a single interface to perform the function evaluations.

13.154.2 Member Function Documentation

void eval_tag_prefix (const String & *eval_id_str*) [protected], [virtual]

set the hierarchical eval ID tag prefix

SingleModel doesn't need to change the tagging, so just forward to Interface

Reimplemented from Model.

References Interface::eval_tag_prefix(), and SingleModel::userDefinedInterface.

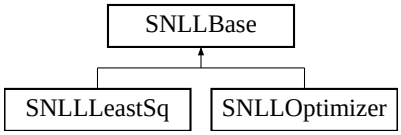
The documentation for this class was generated from the following files:

- SingleModel.hpp
- SingleModel.cpp

13.155 SNLLBase Class Reference

Base class for OPT++ optimization and least squares methods.

Inheritance diagram for SNLLBase:



Public Member Functions

- SNLLBase ()

  default constructor
- SNLLBase (ProblemDescDB &problem_db)

  standard constructor
- ~SNLLBase ()

  destructor

Protected Member Functions

- void copy_con_vals_dak_to_optpp (const RealVector &local_fn_vals, RealVector &g, size_t offset)

  convenience function for copying local_fn_vals to g; used by constraint evaluator functions
- void copy_con_vals_optpp_to_dak (const RealVector &g, RealVector &local_fn_vals, size_t offset)

  convenience function for copying g to local_fn_vals; used in final solution logging
- void copy_con_grad (const RealMatrix &local_fn_grads, RealMatrix &grad_g, size_t offset)

  convenience function for copying local_fn_grads to grad_g; used by constraint evaluator functions
- void copy_con_hess (const RealSymMatrixArray &local_fn_hessians, OPTPP::OptppArray< RealSymMatrix > &hess_g, size_t offset)

  convenience function for copying local_fn_hessians to hess_g; used by constraint evaluator functions
- void snll_pre_instantiate (bool bound_constr_flag, int num_constr)

  convenience function for setting OPT++ options prior to the method instantiation

- void snll_post_instantiate (int num_cv, bool vendor_num_grad_flag, const String &finite_diff_type, const RealVector &fdss, int max_iter, int max_fn_evals, Real conv_tol, Real grad_tol, Real max_step, bool bound_constr_flag, int num_constr, short output_lev, OPTPP::OptimizeClass *the_optimizer, OPTPP::NLP0 *nlf_objective, OPTPP::FDNLF1 *fd_nlf1, OPTPP::FDNLF1 *fd_nlf1_con)

  convenience function for setting OPT++ options after the method instantiation
- void snll_initialize_run (OPTPP::NLP0 *nlf_objective, OPTPP::NLP *nlp_constraint, const RealVector &init_pt, bool bound_constr_flag, const RealVector &lower_bnds, const RealVector &upper_bnds, const RealMatrix &lin_ineq_coeffs, const RealVector &lin_ineq_l_bnds, const RealVector &lin_ineq_u_bnds, const RealMatrix &lin_eq_coeffs, const RealVector &lin_eq_targets, const RealVector &nln_ineq_l_bnds, const RealVector &nln_ineq_u_bnds, const RealVector &nln_eq_targets)

  convenience function for OPT++ configuration prior to the method invocation
- void snll_post_run (OPTPP::NLP0 *nlf_objective)

  convenience function for setting OPT++ options after the method instantiations

Static Protected Member Functions

- static void init_fn (int n, RealVector &x)

  An initialization mechanism provided by OPT++ (not currently used).

Protected Attributes

- String searchMethod

  value_based_line_search, gradient_based_line_search, trust_region, or tr_pds
- OPTPP::SearchStrategy searchStrat

  enum: LineSearch, TrustRegion, or TrustPDS
- OPTPP::MeritFcn meritFn

  enum: NormFmu, ArgaezTapia, or VanShanno
- Real maxStep

  value from `max_step` *specification*
- Real stepLenToBndry

  value from `steplength_to_boundary` *specification*
- Real centeringParam

  value from `centering_parameter` *specification*
- bool constantASVFlag

  flags a user selection of active_set_vector == constant. By mapping this into mode override, reliance on duplicate detection can be avoided.

Static Protected Attributes

- static Minimizer * optLSqInstance

  pointer to the active base class object instance used within the static evaluator functions in order to avoid the need for static data
- static bool modeOverrideFlag

  flags OPT++ mode override (for combining value, gradient, and Hessian requests)
- static EvalType lastFnEvalLocn

an enum used to track whether an nlf evaluator or a constraint evaluator was the last location of a function evaluation

- static int lastEvalMode

  copy of mode from constraint evaluators

- static RealVector lastEvalVars

  copy of variables from constraint evaluators

13.155.1 Detailed Description

Base class for OPT++ optimization and least squares methods.

The SNLLBase class provides a common base class for SNLLOptimizer and SNLLLeastSq, both of which are wrappers for OPT++, a C++ optimization library from the Computational Sciences and Mathematics Research (CSMR) department at Sandia's Livermore CA site.

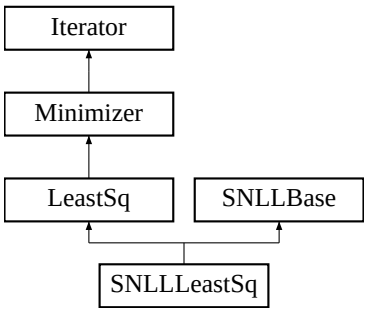
The documentation for this class was generated from the following files:

- SNLLBase.hpp
- SNLLBase.cpp

13.156 SNLLLeastSq Class Reference

Wrapper class for the OPT++ optimization library.

Inheritance diagram for SNLLLeastSq:



Public Member Functions

- SNLLLeastSq (ProblemDescDB &problem_db, Model &model)

  standard constructor

- SNLLLeastSq (const String &method_name, Model &model)

  alternate constructor for instantiations without ProblemDescDB support

- ~SNLLLeastSq ()

  destructor

- void minimize_residuals ()

  Performs the iterations to determine the least squares solution.

Protected Member Functions

- void initialize_run ()

  invokes LeastSq::initialize_run(), SNLLBase::snll_initialize_run(), and performs other set-up
- void post_run (std::ostream &s)

  invokes snll_post_run and re-implements post_run (does not call parent) and performs other solution processing
- void finalize_run ()

  restores instances

Static Private Member Functions

- static void nlf2_evaluator_gn (int mode, int n, const RealVector &x, double &f, RealVector &grad_f, RealSymMatrix &hess_f, int &result_mode)

  objective function evaluator function which obtains values and gradients for least square terms and computes objective function value, gradient, and Hessian using the Gauss-Newton approximation.
- static void constraint1_evaluator_gn (int mode, int n, const RealVector &x, RealVector &g, RealMatrix &grad_g, int &result_mode)

  constraint evaluator function which provides constraint values and gradients to OPT++ Gauss-Newton methods.
- static void constraint2_evaluator_gn (int mode, int n, const RealVector &x, RealVector &g, RealMatrix &grad_g, OPTPP::OptppArray< RealSymMatrix > &hess_g, int &result_mode)

  constraint evaluator function which provides constraint values, gradients, and Hessians to OPT++ Gauss-Newton methods.

Private Attributes

- SNLLLeastSq * prevSnllLSqInstance

  pointer to the previously active object instance used for restoration in the case of iterator/model recursion
- OPTPP::NLP0 * nlfObjective

  objective NLF base class pointer
- OPTPP::NLP0 * nlfConstraint

  constraint NLF base class pointer
- OPTPP::NLP * nlpConstraint

  constraint NLP pointer
- OPTPP::NLF2 * nlf2

  pointer to objective NLF for full Newton optimizers
- OPTPP::NLF2 * nlf2Con

  pointer to constraint NLF for full Newton optimizers
- OPTPP::NLF1 * nlf1Con

  pointer to constraint NLF for Quasi Newton optimizers
- OPTPP::OptimizeClass * theOptimizer

  optimizer base class pointer
- OPTPP::OptNewton * optnewton

  Newton optimizer pointer.
- OPTPP::OptBCNewton * optbcnewton

  Bound constrained Newton optimizer ptr.
- OPTPP::OptDHNIPS * optdhnips

  Disaggregated Hessian NIPS optimizer ptr.

Static Private Attributes

- static SNLLLeastSq * snllLSqInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.156.1 Detailed Description

Wrapper class for the OPT++ optimization library.

The SNLLLeastSq class provides a wrapper for OPT++, a C++ optimization library of nonlinear programming and pattern search techniques from the Computational Sciences and Mathematics Research (CSMR) department at Sandia's Livermore CA site. It uses a function pointer approach for which passed functions must be either global functions or static member functions. Any attribute used within static member functions must be either local to that function, a static member, or accessed by static pointer.

The user input mappings are as follows: `max_iterations`, `max_function_evaluations`, `convergence_tolerance`, `max_step`, `gradient_tolerance`, `search_method`, and `search_scheme_size` are set using OPT++'s setMaxIter(), setMaxFeval(), setFcnTol(), setMaxStep(), setGradTol(), setSearchStrategy(), and setSSS() member functions, respectively; `output` verbosity is used to toggle OPT++'s debug mode using the setDebug() member function. Internal to OPT++, there are 3 search strategies, while the DAKOTA `search_method` specification supports 4 (`value_based_line_search`, `gradient_based_line_search`, `trust_region`, or `tr_pds`). The difference stems from the "is_expensive" flag in OPT++. If the search strategy is LineSearch and "is_expensive" is turned on, then the `value_based_line_search` is used. Otherwise (the "is_expensive" default is off), the algorithm will use the `gradient_based_line_search`. Refer to [Meza, J.C., 1994] and to the OPT++ source in the Dakota/packages/OPTPP directory for information on OPT++ class member functions.

13.156.2 Member Function Documentation

void post_run (std::ostream & *s*) `[protected], [virtual]`

invokes snll_post_run and re-implements post_run (does not call parent) and performs other solution processing

SNLLLeastSq requires fn DB lookup, so overrides LeastSq::post_run and directly invokes Iterator::post_run when complete.

Reimplemented from LeastSq.

References Iterator::activeSet, Iterator::bestResponseArray, Iterator::bestVariablesArray, SNLLBase::copy_con_vals_optpp_to_dak(), Dakota::copy_data_partial(), Minimizer::cvScaleMultipliers, Minimizer::cvScaleOffsets, Minimizer::cvScaleTypes, Dakota::data_pairs, Minimizer::expData, LeastSq::get_confidence_intervals(), Model::interface_id(), Iterator::iteratedModel, Dakota::lookup_by_val(), Minimizer::modify_s2n(), Minimizer::need_resp_trans_byvars(), SNLLLeastSq::nlfObjective, Minimizer::numExperiments, Minimizer::numFunctions, LeastSq::numLeastSqTerms, Minimizer::numNonlinearConstraints, Minimizer::numUserPrimaryFns, Minimizer::obsDataFlag, Minimizer::post_run(), ActiveSet::request_values(), ActiveSet::request_vector(), Minimizer::responseScaleMultipliers, Minimizer::responseScaleOffsets, Minimizer::responseScaleTypes, ExperimentData::scalar_data(), Minimizer::secondaryRespScaleFlag, SNLLBase::snll_post_run(), SNLLLeastSq::theOptimizer, and Minimizer::varsScaleFlag.

void nlf2_evaluator_gn (int *mode,* int *n,* const RealVector & *x,* double & *f,* RealVector & *grad_f,* RealSymMatrix & *hess_f,* int & *result_mode*) `[static], [private]`

objective function evaluator function which obtains values and gradients for least square terms and computes objective function value, gradient, and Hessian using the Gauss-Newton approximation.

This nlf2 evaluator function is used for the Gauss-Newton method in order to exploit the special structure of the nonlinear least squares problem. Here, fx = sum (T_i - Tbar_i)^2 and Response is made up of residual functions and their gradients along with any nonlinear constraints. The objective function and its gradient vector and Hessian matrix are computed directly from the residual functions and their derivatives (which are returned from the Response object).

References Dakota::abort_handler(), Iterator::activeSet, Model::compute_response(), Model::continuous_variables(), Model::current_response(), Response::function_gradients(), Response::function_values(), Iterator::iteratedModel, SNLLBase::lastEvalMode, SNLLBase::lastEvalVars, SNLLBase::lastFnEvalLocn, Minimizer::numFunctions, LeastSq::numLeastSqTerms, Minimizer::numNonlinearConstraints, Iterator::outputLevel, ActiveSet::request_vector(), SNLLLeastSq::snlllLSqInstance, and Dakota::write_precision.

Referenced by SNLLLeastSq::SNLLLeastSq().

void constraint1_evaluator_gn (int *mode,* int *n,* const RealVector & *x,* RealVector & *g,* RealMatrix & *grad_g,* int & *result_mode*) `[static], [private]`

constraint evaluator function which provides constraint values and gradients to OPT++ Gauss-Newton methods.

While it does not employ the Gauss-Newton approximation, it is distinct from constraint1_evaluator() due to its need to anticipate the required modes for the least squares terms. This constraint evaluator function is used with diaggregated Hessian NIPS and is currently active.

References Dakota::abort_handler(), Iterator::activeSet, Model::compute_response(), Model::continuous_variables(), SNLLBase::copy_con_grad(), SNLLBase::copy_con_vals_dak_to_optpp(), Model::current_response(), Response::function_gradients(), Response::function_values(), Iterator::iteratedModel, SNLLBase::lastEvalMode, SNLLBase::lastEvalVars, SNLLBase::lastFnEvalLocn, Minimizer::numFunctions, LeastSq::numLeastSqTerms, Iterator::outputLevel, ActiveSet::request_vector(), and SNLLLeastSq::snlllLSqInstance.

Referenced by SNLLLeastSq::SNLLLeastSq().

void constraint2_evaluator_gn (int *mode,* int *n,* const RealVector & *x,* RealVector & *g,* RealMatrix & *grad_g,* OPTPP::OptppArray< RealSymMatrix > & *hess_g,* int & *result_mode*) `[static], [private]`

constraint evaluator function which provides constraint values, gradients, and Hessians to OPT++ Gauss-Newton methods.

While it does not employ the Gauss-Newton approximation, it is distinct from constraint2_evaluator() due to its need to anticipate the required modes for the least squares terms. This constraint evaluator function is used with full Newton NIPS and is currently inactive.

References Dakota::abort_handler(), Iterator::activeSet, Model::compute_response(), Model::continuous_variables(), SNLLBase::copy_con_grad(), SNLLBase::copy_con_hess(), SNLLBase::copy_con_vals_dak_to_optpp(), Model::current_response(), Response::function_gradients(), Response::function_hessians(), Response::function_values(), Iterator::iteratedModel, SNLLBase::lastEvalMode, SNLLBase::lastEvalVars, SNLLBase::lastFnEvalLocn, SNLLBase::modeOverrideFlag, Minimizer::numFunctions, LeastSq::numLeastSqTerms, Iterator::outputLevel, ActiveSet::request_vector(), and SNLLLeastSq::snlllLSqInstance.

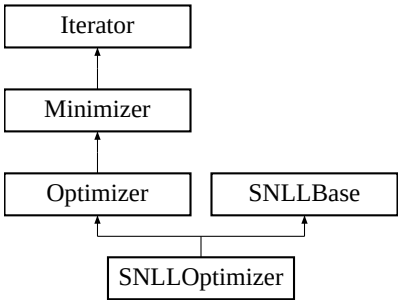
The documentation for this class was generated from the following files:

- SNLLLeastSq.hpp
- SNLLLeastSq.cpp

13.157 SNLLOptimizer Class Reference

Wrapper class for the OPT++ optimization library.

Inheritance diagram for SNLLOptimizer:



Public Member Functions

- SNLLOptimizer (ProblemDescDB &problem_db, Model &model)

  standard constructor

- SNLLOptimizer (const String &method_string, Model &model)

  alternate constructor for instantiations "on the fly"

- SNLLOptimizer (const RealVector &initial_pt, const RealVector &var_l_bnds, const RealVector &var_u_bnds, const RealMatrix &lin_ineq_coeffs, const RealVector &lin_ineq_l_bnds, const RealVector &lin_ineq_u_bnds, const RealMatrix &lin_eq_coeffs, const RealVector &lin_eq_tgts, const RealVector &nln_ineq_l_bnds, const RealVector &nln_ineq_u_bnds, const RealVector &nln_eq_tgts, void(∗user_obj_eval)(int mode, int n, const RealVector &x, double &f, RealVector &grad_f, int &result_mode), void(∗user_con_eval)(int mode, int n, const RealVector &x, RealVector &g, RealMatrix &grad_g, int &result_mode))

  alternate constructor for instantiations "on the fly"

- ∼SNLLOptimizer ()

  destructor

- void find_optimum ()

  Performs the iterations to determine the optimal solution.

Protected Member Functions

- void initialize_run ()

  invokes Optimizer::initialize_run(), SNLLBase::snll_initialize_run(), and performs other set-up

- void post_run (std::ostream &s)

  performs data recovery and calls Optimizer::post_run()

- void finalize_run ()

  performs cleanup, restores instances and calls parent finalize

Static Private Member Functions

- static void nlf0_evaluator (int n, const RealVector &x, double &f, int &result_mode)

  objective function evaluator function for OPT++ methods which require only function values.
- static void nlf1_evaluator (int mode, int n, const RealVector &x, double &f, RealVector &grad_f, int &result_mode)

  objective function evaluator function which provides function values and gradients to OPT++ methods.
- static void nlf2_evaluator (int mode, int n, const RealVector &x, double &f, RealVector &grad_f, RealSymMatrix &hess_f, int &result_mode)

  objective function evaluator function which provides function values, gradients, and Hessians to OPT++ methods.
- static void constraint0_evaluator (int n, const RealVector &x, RealVector &g, int &result_mode)

  constraint evaluator function for OPT++ methods which require only constraint values.
- static void constraint1_evaluator (int mode, int n, const RealVector &x, RealVector &g, RealMatrix &grad_g, int &result_mode)

  constraint evaluator function which provides constraint values and gradients to OPT++ methods.
- static void constraint2_evaluator (int mode, int n, const RealVector &x, RealVector &g, RealMatrix &grad_g, OPTPP::OptppArray< RealSymMatrix > &hess_g, int &result_mode)

  constraint evaluator function which provides constraint values, gradients, and Hessians to OPT++ methods.

Private Attributes

- SNLLOptimizer * prevSnllOptInstance

  pointer to the previously active object instance used for restoration in the case of iterator/model recursion
- OPTPP::NLP0 * nlfObjective

  objective NLF base class pointer
- OPTPP::NLP0 * nlfConstraint

  constraint NLF base class pointer
- OPTPP::NLP * nlpConstraint

  constraint NLP pointer
- OPTPP::NLF0 * nlf0

  pointer to objective NLF for nongradient optimizers
- OPTPP::NLF1 * nlf1

  pointer to objective NLF for (analytic) gradient-based optimizers
- OPTPP::NLF1 * nlf1Con

  pointer to constraint NLF for (analytic) gradient-based optimizers
- OPTPP::FDNLF1 * fdnlf1

  pointer to objective NLF for (finite diff) gradient-based optimizers
- OPTPP::FDNLF1 * fdnlf1Con

  pointer to constraint NLF for (finite diff) gradient-based optimizers
- OPTPP::NLF2 * nlf2

  pointer to objective NLF for full Newton optimizers
- OPTPP::NLF2 * nlf2Con

  pointer to constraint NLF for full Newton optimizers
- OPTPP::OptimizeClass * theOptimizer

  optimizer base class pointer

- OPTPP::OptPDS * optpds

  PDS optimizer pointer.

- OPTPP::OptCG * optcg

  CG optimizer pointer.

- OPTPP::OptLBFGS * optlbfgs

  L-BFGS optimizer pointer.

- OPTPP::OptNewton * optnewton

  Newton optimizer pointer.

- OPTPP::OptQNewton * optqnewton

  Quasi-Newton optimizer pointer.

- OPTPP::OptFDNewton * optfdnewton

  Finite Difference Newton opt pointer.

- OPTPP::OptBCNewton * optbcnewton

  Bound constrained Newton opt pointer.

- OPTPP::OptBCQNewton * optbcqnewton

  Bnd constrained Quasi-Newton opt ptr.

- OPTPP::OptBCFDNewton * optbcfdnewton

  Bnd constrained FD-Newton opt ptr.

- OPTPP::OptNIPS * optnips

  NIPS optimizer pointer.

- OPTPP::OptQNIPS * optqnips

  Quasi-Newton NIPS optimizer pointer.

- OPTPP::OptFDNIPS * optfdnips

  Finite Difference NIPS opt pointer.

- String setUpType

  flag for iteration mode: "model" (normal usage) or "user_functions" (user-supplied functions mode for "on the fly" instantiations). NonDReliability currently uses the user_functions mode.

- RealVector initialPoint

  holds initial point passed in for "user_functions" mode.

- RealVector lowerBounds

  holds variable lower bounds passed in for "user_functions" mode.

- RealVector upperBounds

  holds variable upper bounds passed in for "user_functions" mode.

Static Private Attributes

- static SNLLOptimizer * snllOptInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

Additional Inherited Members

13.157.1 Detailed Description

Wrapper class for the OPT++ optimization library.

The SNLLOptimizer class provides a wrapper for OPT++, a C++ optimization library of nonlinear programming and pattern search techniques from the Computational Sciences and Mathematics Research (CSMR) department at Sandia's Livermore CA site. It uses a function pointer approach for which passed functions must be either global functions or static member functions. Any attribute used within static member functions must be either local to that function, a static member, or accessed by static pointer.

The user input mappings are as follows: `max_iterations`, `max_function_evaluations`, `convergence_tolerance`, `max_step`, `gradient_tolerance`, `search_method`, and `search_scheme_size` are set using OPT++'s setMaxIter(), setMaxFeval(), setFcnTol(), setMaxStep(), setGradTol(), setSearchStrategy(), and setSSS() member functions, respectively; `output` verbosity is used to toggle OPT++'s debug mode using the setDebug() member function. Internal to OPT++, there are 3 search strategies, while the DAKOTA `search_method` specification supports 4 (`value_based_line_search`, `gradient_based_line_search`, `trust_region`, or `tr_pds`). The difference stems from the "is_expensive" flag in OPT++. If the search strategy is LineSearch and "is_expensive" is turned on, then the `value_based_line_search` is used. Otherwise (the "is_expensive" default is off), the algorithm will use the `gradient_based_line_search`. Refer to [Meza, J.C., 1994] and to the OPT++ source in the Dakota/packages/OPTPP directory for information on OPT++ class member functions.

13.157.2 Constructor & Destructor Documentation

SNLLOptimizer (ProblemDescDB & *problem_db,* Model & *model*)

standard constructor

This constructor is used for normal instantiations using data from the ProblemDescDB.

References Dakota::abort_handler(), Minimizer::boundConstraintFlag, SNLLBase::centeringParam, SNLLOptimizer::constraint0_evaluator(), SNLLOptimizer::constraint1_evaluator(), SNLLOptimizer::constraint2_evaluator(), Iterator::convergenceTol, Model::fd_gradient_step_size(), SNLLOptimizer::fdnlf1, SNLLOptimizer::fdnlf1Con, ProblemDescDB::get_int(), ProblemDescDB::get_real(), SNLLBase::init_fn(), Model::interval_type(), Iterator::iteratedModel, Dakota::LARGE_SCALE, Iterator::maxEvalConcurrency, Iterator::maxFunctionEvals, Iterator::maxIterations, SNLLBase::maxStep, SNLLBase::meritFn, Iterator::method_enum_to_string(), Iterator::methodName, SNLLOptimizer::nlf0, SNLLOptimizer::nlf0_evaluator(), SNLLOptimizer::nlf1, SNLLOptimizer::nlf1_evaluator(), SNLLOptimizer::nlf1Con, SNLLOptimizer::nlf2, SNLLOptimizer::nlf2_evaluator(), SNLLOptimizer::nlf2Con, SNLLOptimizer::nlfConstraint, SNLLOptimizer::nlfObjective, SNLLOptimizer::nlpConstraint, Minimizer::numConstraints, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, SNLLOptimizer::optbcfdnewton, SNLLOptimizer::optbcnewton, SNLLOptimizer::optbcqnewton, SNLLOptimizer::optcg, SNLLOptimizer::optfdnewton, SNLLOptimizer::optfdnips, SNLLOptimizer::optlbfgs, SNLLOptimizer::optnewton, SNLLOptimizer::optnips, SNLLOptimizer::optpds, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, Iterator::outputLevel, Iterator::probDescDB, SNLLBase::searchStrat, SNLLBase::snll_post_instantiate(), SNLLBase::snll_pre_instantiate(), SNLLBase::stepLenToBndry, SNLLOptimizer::theOptimizer, and Minimizer::vendorNumericalGradFlag.

SNLLOptimizer (const String & *method_string,* Model & *model*)

alternate constructor for instantiations "on the fly"

This is an alternate constructor for instantiations on the fly using a Model but no ProblemDescDB.

References Minimizer::boundConstraintFlag, SNLLOptimizer::constraint1_evaluator(), Iterator::convergenceTol, Model::fd_gradient_step_size(), SNLLBase::init_fn(), Model::interval_type(), Iterator::iteratedModel, Dakota::LARGE_SCALE, Iterator::maxFunctionEvals, Iterator::maxIterations, SNLLBase::meritFn, Iterator::method-

Name, SNLLOptimizer::nlf1, SNLLOptimizer::nlf1_evaluator(), SNLLOptimizer::nlf1Con, SNLLOptimizer::nlfConstraint, SNLLOptimizer::nlfObjective, SNLLOptimizer::nlpConstraint, Minimizer::numConstraints, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, SNLLOptimizer::optbcqnewton, SNLLOptimizer::optlbfgs, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, Iterator::outputLevel, SNLLBase::searchStrat, SNLLBase::snll_post_instantiate(), SNLLBase::snll_pre_instantiate(), SNLLOptimizer::theOptimizer, and Minimizer::vendorNumericalGradFlag.

SNLLOptimizer (const RealVector & *initial_pt,* const RealVector & *var_l_bnds,* const RealVector & *var_u_bnds,* const RealMatrix & *lin_ineq_coeffs,* const RealVector & *lin_ineq_l_bnds,* const RealVector & *lin_ineq_u_bnds,* const RealMatrix & *lin_eq_coeffs,* const RealVector & *lin_eq_tgts,* const RealVector & *nln_ineq_l_bnds,* const RealVector & *nln_ineq_u_bnds,* const RealVector & *nln_eq_tgts,* void(∗)(int mode, int n, const RealVector &x, double &f, RealVector &grad_f, int &result_mode) *user_obj_eval,* void(∗)(int mode, int n, const RealVector &x, RealVector &g, RealMatrix &grad_g, int &result_mode) *user_con_eval*)

alternate constructor for instantiations "on the fly"

This is an alternate constructor for performing an optimization using the passed in objective function and constraint function pointers.

References Minimizer::bigRealBoundSize, Minimizer::boundConstraintFlag, SNLLBase::init_fn(), SNLLOptimizer::initialPoint, Dakota::LARGE_SCALE, SNLLOptimizer::lowerBounds, SNLLBase::meritFn, SNLLOptimizer::nlf1, SNLLOptimizer::nlf1Con, SNLLOptimizer::nlfConstraint, SNLLOptimizer::nlfObjective, SNLLOptimizer::nlpConstraint, Minimizer::numConstraints, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, SNLLOptimizer::optbcqnewton, SNLLOptimizer::optlbfgs, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, Iterator::outputLevel, SNLLBase::searchStrat, SNLLBase::snll_initialize_run(), SNLLBase::snll_post_instantiate(), SNLLBase::snll_pre_instantiate(), SNLLOptimizer::theOptimizer, and SNLLOptimizer::upperBounds.

13.157.3 Member Function Documentation

void nlf0_evaluator (int *n,* const RealVector & *x,* double & *f,* int & *result_mode*) `[static], [private]`

objective function evaluator function for OPT++ methods which require only function values.

For use when DAKOTA computes f and gradients are not directly available. This is used by nongradient-based optimizers such as PDS and by gradient-based optimizers in vendor numerical gradient mode (opt++'s internal finite difference routine is used).

References Model::compute_response(), Model::continuous_variables(), Model::current_response(), Response::function_value(), Iterator::iteratedModel, SNLLBase::lastEvalVars, SNLLBase::lastFnEvalLocn, Minimizer::numNonlinearConstraints, Iterator::outputLevel, Model::primary_response_fn_sense(), and SNLLOptimizer::snllOptInstance.

Referenced by SNLLOptimizer::SNLLOptimizer().

void nlf1_evaluator (int *mode,* int *n,* const RealVector & *x,* double & *f,* RealVector & *grad_f,* int & *result_mode*) `[static],[private]`

objective function evaluator function which provides function values and gradients to OPT++ methods.

For use when DAKOTA computes f and df/dX (regardless of gradient type). Vendor numerical gradient case is handled by nlf0_evaluator.

References Iterator::activeSet, Model::compute_response(), Model::continuous_variables(), Model::current_response(), Response::function_gradient_copy(), Response::function_value(), Iterator::iteratedModel, SNLLBase::lastEvalMode, SNLLBase::lastEvalVars, SNLLBase::lastFnEvalLocn, Minimizer::numNonlinearConstraints, Iterator::outputLevel, Model::primary_response_fn_sense(), ActiveSet::request_values(), and SNLLOptimizer::snllOptInstance.

Referenced by SNLLOptimizer::SNLLOptimizer().

void nlf2_evaluator (int *mode,* int *n,* const RealVector & *x,* double & *f,* RealVector & *grad_f,* RealSymMatrix & *hess_f,* int & *result_mode*) `[static], [private]`

objective function evaluator function which provides function values, gradients, and Hessians to OPT++ methods.

For use when DAKOTA receives f, df/dX, & d^2f/dx^2 from the ApplicationInterface (analytic only). Finite differencing does not make sense for a full Newton approach, since lack of analytic gradients & Hessian should dictate the use of quasi-newton or fd-newton. Thus, there is no fdnlf2_evaluator for use with full Newton approaches, since it is preferable to use quasi-newton or fd-newton with nlf1. Gauss-Newton does not fit this model; it uses nlf2_evaluator_gn instead of nlf2_evaluator.

References Iterator::activeSet, Model::compute_response(), Model::continuous_variables(), Model::current_response(), Response::function_gradient_copy(), Response::function_hessian(), Response::function_value(), Iterator::iteratedModel, SNLLBase::lastEvalMode, SNLLBase::lastEvalVars, SNLLBase::lastFnEvalLocn, Minimizer::numNonlinearConstraints, Iterator::outputLevel, Model::primary_response_fn_sense(), ActiveSet::request_values(), and SNLLOptimizer::snllOptInstance.

Referenced by SNLLOptimizer::SNLLOptimizer().

void constraint0_evaluator (int *n,* const RealVector & *x,* RealVector & *g,* int & *result_mode*) `[static], [private]`

constraint evaluator function for OPT++ methods which require only constraint values.

For use when DAKOTA computes g and gradients are not directly available. This is used by nongradient-based optimizers and by gradient-based optimizers in vendor numerical gradient mode (opt++'s internal finite difference routine is used).

References Model::compute_response(), Model::continuous_variables(), SNLLBase::copy_con_vals_dak_to_optpp(), Model::current_response(), Response::function_values(), Iterator::iteratedModel, SNLLBase::lastEvalVars, SNLLBase::lastFnEvalLocn, Optimizer::numObjectiveFns, Iterator::outputLevel, and SNLLOptimizer::snllOptInstance.

Referenced by SNLLOptimizer::SNLLOptimizer().

void constraint1_evaluator (int *mode,* int *n,* const RealVector & *x,* RealVector & *g,* RealMatrix & *grad_g,* int & *result_mode*) `[static], [private]`

constraint evaluator function which provides constraint values and gradients to OPT++ methods.

For use when DAKOTA computes g and dg/dX (regardless of gradient type). Vendor numerical gradient case is handled by constraint0_evaluator.

References Iterator::activeSet, Model::compute_response(), Model::continuous_variables(), SNLLBase::copy_con_grad(), SNLLBase::copy_con_vals_dak_to_optpp(), Model::current_response(), Response::function_gradients(), Response::function_values(), Iterator::iteratedModel, SNLLBase::lastEvalMode, SNLLBase::lastEvalVars, SNLLBase::lastFnEvalLocn, Optimizer::numObjectiveFns, Iterator::outputLevel, ActiveSet::request_values(), and SNLLOptimizer::snllOptInstance.

Referenced by SNLLOptimizer::SNLLOptimizer().

void constraint2_evaluator (int *mode,* int *n,* const RealVector & *x,* RealVector & *g,* RealMatrix & *grad_g,* OPTPP::OptppArray< RealSymMatrix > & *hess_g,* int & *result_mode*) `[static], [private]`

constraint evaluator function which provides constraint values, gradients, and Hessians to OPT++ methods.

For use when DAKOTA computes g, dg/dX, & d^2g/dx^2 (analytic only).

References Iterator::activeSet, Model::compute_response(), Model::continuous_variables(), SNLLBase::copy_con_grad(), SNLLBase::copy_con_hess(), SNLLBase::copy_con_vals_dak_to_optpp(), Model::current_response(), Response::function_gradients(), Response::function_hessians(), Response::function_values(), Iterator::iteratedModel, SNLLBase::lastEvalMode, SNLLBase::lastEvalVars, SNLLBase::lastFnEvalLocn, Optimizer::numObjectiveFns, Iterator::outputLevel, ActiveSet::request_values(), and SNLLOptimizer::snllOptInstance.

Referenced by SNLLOptimizer::SNLLOptimizer().

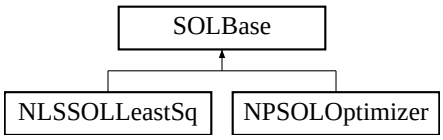
The documentation for this class was generated from the following files:

- SNLLOptimizer.hpp
- SNLLOptimizer.cpp

13.158 SOLBase Class Reference

Base class for Stanford SOL software.

Inheritance diagram for SOLBase:



Public Member Functions

- SOLBase ()

  default constructor
- SOLBase (Model &model)

  standard constructor
- ~SOLBase ()

  destructor

Protected Member Functions

- void allocate_arrays (int num_cv, size_t num_nln_con, const RealMatrix &lin_ineq_coeffs, const RealMatrix &lin_eq_coeffs)

  Allocates miscellaneous arrays for the SOL algorithms.
- void deallocate_arrays ()

  Deallocates memory previously allocated by allocate_arrays().
- void allocate_workspace (int num_cv, int num_nln_con, int num_lin_con, int num_lsq)

  Allocates real and integer workspaces for the SOL algorithms.
- void set_options (bool speculative_flag, bool vendor_num_grad_flag, short output_lev, int verify_lev, Real fn_prec, Real linesrch_tol, int max_iter, Real constr_tol, Real conv_tol, const std::string &grad_type, const RealVector &fdss)

  Sets SOL method options using calls to npoptn2.
- void augment_bounds (RealVector &augmented_l_bnds, RealVector &augmented_u_bnds, const RealVector &lin_ineq_l_bnds, const RealVector &lin_ineq_u_bnds, const RealVector &lin_eq_targets, const RealVector &nln_ineq_l_bnds, const RealVector &nln_ineq_u_bnds, const RealVector &nln_eq_targets)

  augments variable bounds with linear and nonlinear constraint bounds.

Static Protected Member Functions

- static void constraint_eval (int &mode, int &ncnln, int &n, int &nrowj, int *needc, double *x, double *c, double *cjac, int &nstate)

  CONFUN in NPSOL manual: computes the values and first derivatives of the nonlinear constraint functions.

Protected Attributes

- int realWorkSpaceSize

  size of realWorkSpace
- int intWorkSpaceSize

  size of intWorkSpace
- RealArray realWorkSpace

  real work space for NPSOL/NLSSOL
- IntArray intWorkSpace

  int work space for NPSOL/NLSSOL
- int nlnConstraintArraySize

  used for non-zero array sizing (nonlinear constraints)
- int linConstraintArraySize

  used for non-zero array sizing (linear constraints)
- RealArray cLambda

  CLAMBDA from NPSOL manual: Langrange multipliers.
- IntArray constraintState

  ISTATE from NPSOL manual: constraint status.
- int informResult

  INFORM from NPSOL manual: optimization status on exit.
- int numberIterations

  ITER from NPSOL manual: number of (major) iterations performed.
- int boundsArraySize

  length of augmented bounds arrays (variable bounds plus linear and nonlinear constraint bounds)
- double * linConstraintMatrixF77

  [A] matrix from NPSOL manual: linear constraint coefficients
- double * upperFactorHessianF77

  [R] matrix from NPSOL manual: upper Cholesky factor of the Hessian of the Lagrangian.
- double * constraintJacMatrixF77

  [CJAC] matrix from NPSOL manual: nonlinear constraint Jacobian
- int fnEvalCntr

  counter for testing against maxFunctionEvals
- size_t constrOffset

  used in constraint_eval() to bridge NLSSOLLeastSq::numLeastSqTerms and NPSOLOptimizer::numObjectiveFns

Static Protected Attributes

- static SOLBase * solInstance

  pointer to the active object instance used within the static evaluator functions in order to avoid the need for static data

- static Minimizer * optLSqInstance

  pointer to the active base class object instance used within the static evaluator functions in order to avoid the need for static data

13.158.1 Detailed Description

Base class for Stanford SOL software.

The SOLBase class provides a common base class for NPSOLOptimizer and NLSSOLLeastSq, both of which are Fortran 77 sequential quadratic programming algorithms from Stanford University marketed by Stanford Business Associates.

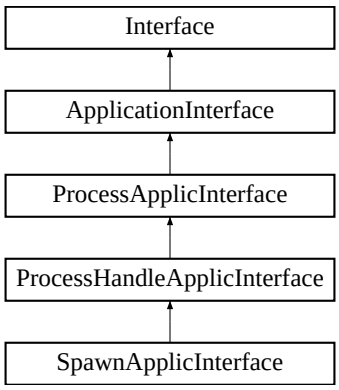
The documentation for this class was generated from the following files:

- SOLBase.hpp
- SOLBase.cpp

13.159 SpawnApplicInterface Class Reference

Derived application interface class which spawns simulation codes using spawnvp.

Inheritance diagram for SpawnApplicInterface:



Public Member Functions

- SpawnApplicInterface (const ProblemDescDB &problem_db)

  constructor

- ~SpawnApplicInterface ()

  destructor

Protected Member Functions

- void wait_local_evaluations (PRPQueue &prp_queue)

  For asynchronous function evaluations, this method is used to detect completion of jobs and process their results. It provides the processing code that is specific to derived classes. This version waits for at least one completion.
- void test_local_evaluations (PRPQueue &prp_queue)

  For asynchronous function evaluations, this method is used to detect completion of jobs and process their results. It provides the processing code that is specific to derived classes. This version is nonblocking and will return without any completions if none are immediately available.
- pid_t create_analysis_process (bool block_flag, bool new_group)

  spawn a child process for an analysis component within an evaluation
- size_t wait_local_analyses ()

  wait for asynchronous analyses on the local processor, completing at least one job
- size_t test_local_analyses_send (int analysis_id)

  test for asynchronous analysis completions on the local processor and return results for any completions by sending messages

Additional Inherited Members

13.159.1 Detailed Description

Derived application interface class which spawns simulation codes using spawnvp.

SpawnApplicInterface is used on Windows systems and is a peer to ForkApplicInterface for Unix systems.

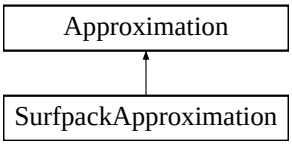
The documentation for this class was generated from the following files:

- SpawnApplicInterface.hpp
- SpawnApplicInterface.cpp

13.160 SurfpackApproximation Class Reference

Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota.

Inheritance diagram for SurfpackApproximation:



Public Member Functions

- SurfpackApproximation ()

  default constructor
- SurfpackApproximation (const ProblemDescDB &problem_db, const SharedApproxData &shared_data)

  standard constructor: Surfpack surface of appropriate type will be created
- SurfpackApproximation (const SharedApproxData &shared_data)

  alternate constructor
- ~SurfpackApproximation ()

  destructor

Protected Member Functions

- int min_coefficients () const

  return the minimum number of samples (unknowns) required to build the derived class approximation type in numVars dimensions
- int recommended_coefficients () const

  return the recommended number of samples (unknowns) required to build the derived class approximation type in numVars dimensions
- void build ()

  SurfData object will be created from Dakota's SurrogateData, and the appropriate Surfpack build method will be invoked.
- Real value (const Variables &vars)

  Return the value of the Surfpack surface for a given parameter vector x.
- const RealVector & gradient (const Variables &vars)

  retrieve the approximate function gradient for a given parameter vector x
- const RealSymMatrix & hessian (const Variables &vars)

  retrieve the approximate function Hessian for a given parameter vector x
- Real prediction_variance (const Variables &vars)

  retrieve the variance of the predicted value for a given parameter set x (KrigingModel only)
- bool diagnostics_available ()

  check if the diagnostics are available (true for the Surfpack types)
- Real diagnostic (const String &metric_type)

  retrieve a single diagnostic metric for the diagnostic type specified on the primary model and data
- Real diagnostic (const String &metric_type, const SurfpackModel &model, const SurfData &data)

  retrieve a single diagnostic metric for the diagnostic type specified on the given model and data
- void primary_diagnostics (int fn_index)

  compute and print all requested diagnostics and cross-validation
- void challenge_diagnostics (const RealMatrix &challenge_points, int fn_index)

  compute and print all requested diagnostics for user provided challenge pts

Private Member Functions

- SurfData * surrogates_to_surf_data ()

  copy from SurrogateData to SurfPoint/SurfData
- void add_anchor_to_surfdata (SurfData &surf_data)

  set the anchor point (including gradient and hessian if present) into surf_data

Private Attributes

- SurfpackModel * model

  The native Surfpack approximation.
- SurfpackModelFactory * factory

  factory for the SurfpackModel instance
- SurfData * surfData

  The data used to build the approximation, in Surfpack format.

Additional Inherited Members

13.160.1 Detailed Description

Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota.

The SurfpackApproximation class is the interface between Dakota and Surfpack. Based on the information in the ProblemDescDB that is passed in through the constructor, SurfpackApproximation builds a Surfpack Surface object that corresponds to one of the following data-fitting techniques: polynomial regression, kriging, artificial neural networks, radial basis function network, or multivariate adaptaive regression splines (MARS).

13.160.2 Constructor & Destructor Documentation

SurfpackApproximation (const ProblemDescDB & *problem_db,* const SharedApproxData & *shared_data*)

standard constructor: Surfpack surface of appropriate type will be created

Initialize the embedded Surfpack surface object and configure it using the specifications from the input file. Data for the surface is created later.

References Dakota::abort_handler(), SharedSurfpackApproxData::approxOrder, SharedApproxData::approxType, SharedApproxData::buildDataOrder, Dakota::copy_data(), SharedSurfpackApproxData::crossValidateFlag, SharedSurfpackApproxData::diagnosticSet, SharedSurfpackApproxData::exportModelName, SurfpackApproximation::factory, ProblemDescDB::get_real(), ProblemDescDB::get_rv(), ProblemDescDB::get_short(), ProblemDescDB::get_string(), SharedSurfpackApproxData::numFolds, SharedApproxData::numVars, SharedApproxData::outputLevel, SharedSurfpackApproxData::percentFold, Approximation::sharedDataRep, and Dakota::strends().

SurfpackApproximation (const SharedApproxData & *shared_data*)

alternate constructor

On-the-fly constructor which uses mostly Surfpack model defaults.

References Dakota::abort_handler(), SharedSurfpackApproxData::approxOrder, SharedApproxData::approxType, SharedApproxData::buildDataOrder, SurfpackApproximation::factory, SharedApproxData::numVars, SharedApproxData::outputLevel, and Approximation::sharedDataRep.

13.160.3 Member Function Documentation

void build ( ) `[protected], [virtual]`

SurfData object will be created from Dakota's SurrogateData, and the appropriate Surfpack build method will be invoked.

surfData will be deleted in dtor

Todo Right now, we're completely deleting the old data and then recopying the current data into a SurfData object. This was just the easiest way to arrive at a solution that would build and run. This function is frequently called from addPoint rebuild, however, and it's not good to go through this whole process every time one more data point is added.

Reimplemented from Approximation.

References Dakota::abort_handler(), SharedApproxData::approxCLowerBnds, SharedApproxData::approxCUpperBnds, SharedApproxData::approxDILowerBnds, SharedApproxData::approxDIUpperBnds, SharedApproxData::approxDRLowerBnds, SharedApproxData::approxDRUpperBnds, Approximation::build(), SharedSurfpackApproxData::exportModelName, SurfpackApproximation::factory, SharedSurfpackApproxData::merge_variable_arrays(), SurfpackApproximation::model, SharedApproxData::outputLevel, Approximation::sharedDataRep, SurfpackApproximation::surfData, and SurfpackApproximation::surrogates_to_surf_data().

const RealSymMatrix & hessian (const Variables & *vars*) [protected], [virtual]

retrieve the approximate function Hessian for a given parameter vector x

Todo Make this acceptably efficient

Reimplemented from Approximation.

References Dakota::abort_handler(), Approximation::approxHessian, SharedApproxData::approxType, Variables::cv(), SurfpackApproximation::model, and Approximation::sharedDataRep.

Referenced by SurfpackApproximation::add_anchor_to_surfdata().

SurfData * surrogates_to_surf_data ( ) [private]

copy from SurrogateData to SurfPoint/SurfData

Copy the data stored in Dakota-style SurrogateData into Surfpack-style SurfPoint and SurfData objects.

References SurfpackApproximation::add_anchor_to_surfdata(), SharedSurfpackApproxData::add_sd_to_surfdata(), Approximation::approxData, SharedApproxData::buildDataOrder, SurfpackApproximation::factory, SharedApproxData::outputLevel, and Approximation::sharedDataRep.

Referenced by SurfpackApproximation::build().

void add_anchor_to_surfdata (SurfData & *surf_data*) [private]

set the anchor point (including gradient and hessian if present) into surf_data

If there is an anchor point, add an equality constraint for its response value. Also add constraints for gradient and hessian, if applicable.

References Dakota::abort_handler(), Approximation::approxData, Dakota::copy_data(), SharedSurfpackApproxData::copy_matrix(), SurfpackApproximation::gradient(), SurfpackApproximation::hessian(), SharedApproxData::outputLevel, SharedSurfpackApproxData::sdv_to_realarray(), and Approximation::sharedDataRep.

Referenced by SurfpackApproximation::surrogates_to_surf_data().

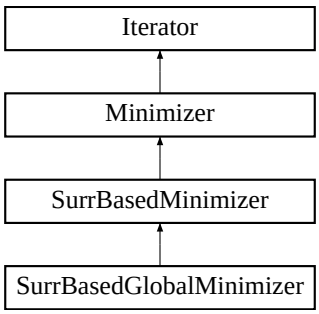
The documentation for this class was generated from the following files:

- SurfpackApproximation.hpp
- SurfpackApproximation.cpp

13.161 SurrBasedGlobalMinimizer Class Reference

The global surrogate-based minimizer which sequentially minimizes and updates a global surrogate model without trust region controls.

Inheritance diagram for SurrBasedGlobalMinimizer:



Public Member Functions

- SurrBasedGlobalMinimizer (ProblemDescDB &problem_db, Model &model)

  constructor

- ~SurrBasedGlobalMinimizer ()

  destructor

Protected Member Functions

- bool returns_multiple_points () const

  Global surrogate-based methods can return multiple points.

Private Member Functions

- void minimize_surrogates ()

  Performs global surrogate-based optimization by repeatedly optimizing on and improving surrogates of the response functions.

Private Attributes

- bool replacePoints

  flag for replacing the previous iteration's point additions, rather than continuing to append, during construction of the next surrogate

Additional Inherited Members

13.161.1 Detailed Description

The global surrogate-based minimizer which sequentially minimizes and updates a global surrogate model without trust region controls.

This method uses a SurrogateModel to perform minimization (optimization or nonlinear least squares) through a set of iterations. At each iteration, a surrogate is built, the surrogate is minimized, and the optimal points from the surrogate are then evaluated with the "true" function, to generate new points upon which the surrogate for the next iteration is built.

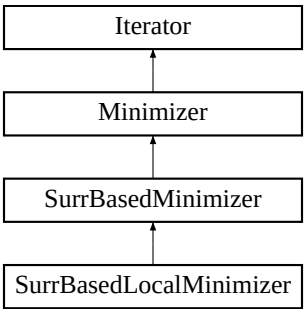
The documentation for this class was generated from the following files:

- SurrBasedGlobalMinimizer.hpp
- SurrBasedGlobalMinimizer.cpp

13.162 SurrBasedLocalMinimizer Class Reference

Class for provably-convergent local surrogate-based optimization and nonlinear least squares.

Inheritance diagram for SurrBasedLocalMinimizer:



Public Member Functions

- SurrBasedLocalMinimizer (ProblemDescDB &problem_db, Model &model)

  constructor
- ~SurrBasedLocalMinimizer ()

  destructor

Protected Member Functions

- void reset ()

  reset convergence controls in case of multiple SBLM executions

Private Member Functions

- void minimize_surrogates ()

  Performs local surrogate-based minimization by minimizing local, global, or hierarchical surrogates over a series of trust regions.
- bool tr_bounds (const RealVector &global_lower_bnds, const RealVector &global_upper_bnds, RealVector &tr_lower_bnds, RealVector &tr_upper_bnds)

  compute current trust region bounds
- void find_center_truth (const Iterator &dace_iterator, Model &truth_model)

  retrieve responseCenterTruth if possible, evaluate it if not
- void find_center_approx ()

  retrieve responseCenter_approx if possible, evaluate it if not
- void hard_convergence_check (const Response &response_truth, const RealVector &c_vars, const RealVector &lower_bnds, const RealVector &upper_bnds)

  check for hard convergence (norm of projected gradient of merit function near zero)
- void tr_ratio_check (const RealVector &c_vars_star, const RealVector &tr_lower_bounds, const RealVector &tr_upper_bounds)

  compute trust region ratio (for SBLM iterate acceptance and trust region resizing) and check for soft convergence (diminishing returns)
- void update_penalty (const RealVector &fns_center_truth, const RealVector &fns_star_truth)

  initialize and update the penaltyParameter
- void relax_constraints (const RealVector &lower_bnds, const RealVector &upper_bnds)

  relax constraints by updating bounds when current iterate is infeasible

Static Private Member Functions

- static void approx_subprob_objective_eval (const Variables &surrogate_vars, const Variables &recast_vars, const Response &surrogate_response, Response &recast_response)

  static function used to define the approximate subproblem objective.
- static void approx_subprob_constraint_eval (const Variables &surrogate_vars, const Variables &recast_vars, const Response &surrogate_response, Response &recast_response)

  static function used to define the approximate subproblem constraints.
- static void hom_objective_eval (int &mode, int &n, double *tau_and_x, double &f, double *grad_f, int &)

  static function used by NPSOL as the objective function in the homotopy constraint relaxation formulation.
- static void hom_constraint_eval (int &mode, int &ncnln, int &n, int &nrowj, int *needc, double *tau_and_x, double *c, double *cjac, int &nstate)

  static function used by NPSOL as the constraint function in the homotopy constraint relaxation formulation.

Private Attributes

- Real origTrustRegionFactor

  original user specification for trustRegionFactor
- Real trustRegionFactor

  the trust region factor is used to compute the total size of the trust region – it is a percentage, e.g. for trustRegionFactor = 0.1, the actual size of the trust region will be 10% of the global bounds (upper bound - lower bound for each design variable).
- Real minTrustRegionFactor

  a soft convergence control: stop SBLM when the trust region factor is reduced below the value of minTrustRegionFactor
- Real trRatioContractValue

  trust region ratio min value: contract tr if ratio below this value
- Real trRatioExpandValue

  trust region ratio sufficient value: expand tr if ratio above this value
- Real gammaContract

  trust region contraction factor
- Real gammaExpand

  trust region expansion factor
- short approxSubProbObj

  type of approximate subproblem objective: ORIGINAL_OBJ, LAGRANGIAN_OBJ, or AUGMENTED_LAGRANGIAN_OBJ
- short approxSubProbCon

  type of approximate subproblem constraints: NO_CON, LINEARIZED_CON, or ORIGINAL_CON
- Model approxSubProbModel

  the approximate sub-problem formulation solved on each approximate minimization cycle: may be a shallow copy of iteratedModel, or may involve a RecastModel recursion applied to iteratedModel
- bool recastSubProb

  flag to indicate when approxSubProbModel involves a RecastModel recursion
- short trConstraintRelax

  type of trust region constraint relaxation for infeasible starting points: NO_RELAX or HOMOTOPY

- short meritFnType

  type of merit function used in trust region ratio logic: PENALTY_MERIT, ADAPTIVE_PENALTY_MERIT, LAGRANGIAN_MERIT, or AUGMENTED_LAGRANGIAN_MERIT
- short acceptLogic

  type of iterate acceptance test logic: FILTER or TR_RATIO
- int penaltyIterOffset

  iteration offset used to update the scaling of the penalty parameter for adaptive_penalty merit functions
- short convergenceFlag

  code indicating satisfaction of hard or soft convergence conditions
- unsigned short softConvCount

  number of consecutive candidate point rejections. If the count reaches softConvLimit, stop SBLM.
- unsigned short softConvLimit

  the limit on consecutive candidate point rejections. If exceeded by softConvCount, stop SBLM.
- bool truthGradientFlag

  flags the use/availability of truth gradients within the SBLM process
- bool approxGradientFlag

  flags the use/availability of surrogate gradients within the SBLM process
- bool truthHessianFlag

  flags the use/availability of truth Hessians within the SBLM process
- bool approxHessianFlag

  flags the use/availability of surrogate Hessians within the SBLM process
- short correctionType

  flags the use of surrogate correction techniques at the center of each trust region
- bool globalApproxFlag

  flags the use of a global data fit surrogate (rsm, ann, mars, kriging)
- bool multiptApproxFlag

  flags the use of a multipoint data fit surrogate (TANA)
- bool localApproxFlag

  flags the use of a local data fit surrogate (Taylor series)
- bool hierarchApproxFlag

  flags the use of a model hierarchy/multifidelity surrogate
- bool newCenterFlag

  flags the acceptance of a candidate point and the existence of a new trust region center
- bool daceCenterPtFlag

  flags the availability of the center point in the DACE evaluations for global approximations (CCD, Box-Behnken)
- bool multiLayerBypassFlag

  flags the simultaneous presence of two conditions: (1) additional layerings w/i actual_model (e.g., surrogateModel = layered/nested/layered -> actual_model = nested/layered), and (2) a user-specification to bypass all layerings within actual_model for the evaluation of truth data (responseCenterTruth and responseStarTruth).
- bool useDerivsFlag

  flag for the "use_derivatives" specification for which derivatives are to be evaluated at each DACE point in global surrogate builds.
- RealVector nonlinIneqLowerBndsSlack

  individual violations of nonlinear inequality constraint lower bounds

- RealVector nonlinIneqUpperBndsSlack

  individual violations of nonlinear inequality constraint upper bounds
- RealVector nonlinEqTargetsSlack

  individual violations of nonlinear equality constraint targets
- Real tau

  constraint relaxation parameter
- Real alpha

  constraint relaxation parameter backoff parameter (multiplier)
- Variables varsCenter

  variables at the trust region center
- Response responseCenterApprox

  approx response at trust region center
- Response responseStarApprox

  approx response at SBLM cycle minimum
- IntResponsePair responseCenterTruth

  truth response at trust region center
- IntResponsePair responseStarTruth

  truth response at SBLM cycle minimum

Static Private Attributes

- static SurrBasedLocalMinimizer * sblmInstance

  pointer to SBLM instance used in static member functions

Additional Inherited Members

13.162.1 Detailed Description

Class for provably-convergent local surrogate-based optimization and nonlinear least squares.

This minimizer uses a SurrogateModel to perform minimization based on local, global, or hierarchical surrogates. It achieves provable convergence through the use of a sequence of trust regions and the application of surrogate corrections at the trust region centers.

13.162.2 Member Function Documentation

void minimize_surrogates ( ) `[private], [virtual]`

Performs local surrogate-based minimization by minimizing local, global, or hierarchical surrogates over a series of trust regions.

Trust region-based strategy to perform surrogate-based optimization in subregions (trust regions) of the parameter space. The minimizer operates on approximations in lieu of the more expensive simulation-based response functions. The size of the trust region is varied according to the goodness of the agreement between the approximations and the true response functions.

Implements SurrBasedMinimizer.

References Dakota::abort_handler(), Iterator::active_set(), Response::active_set(), Model::active_variables(), OutputManager::add_datapoint(), DiscrepancyCorrection::apply(), SurrBasedLocalMinimizer::approxGradientFlag, SurrBasedLocalMinimizer::approxHessianFlag, SurrBasedMinimizer::approxSubProbMinimizer, SurrBasedLocalMinimizer::approxSubProbModel, Iterator::bestResponseArray, Iterator::bestVariablesArray, Model::build_approximation(),

Model::component_parallel_mode(), DiscrepancyCorrection::compute(), Model::compute_response(), Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Variables::continuous_variables(), Model::continuous_variables(), SurrBasedLocalMinimizer::convergenceFlag, Variables::copy(), Dakota::copy_data(), SurrBasedLocalMinimizer::correctionType, Model::current_response(), Model::current_variables(), SurrBasedLocalMinimizer::daceCenterPtFlag, Model::discrepancy_correction(), Model::evaluation_id(), SurrBasedLocalMinimizer::find_center_approx(), SurrBasedLocalMinimizer::find_center_truth(), SurrBasedLocalMinimizer::globalApproxFlag, SurrBasedLocalMinimizer::hard_convergence_check(), Model::interface_id(), Iterator::is_null(), Iterator::iteratedModel, SurrBasedLocalMinimizer::localApproxFlag, Iterator::maxIterations, Iterator::methodPCIter, SurrBasedLocalMinimizer::minTrustRegionFactor, SurrBasedMinimizer::miPLIndex, SurrBasedLocalMinimizer::multiLayerBypassFlag, SurrBasedLocalMinimizer::multiptApproxFlag, SurrBasedLocalMinimizer::newCenterFlag, Model::nonlinear_eq_constraint_targets(), Model::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Minimizer::numContinuousVars, SurrBasedMinimizer::origNonlinEqTargets, SurrBasedMinimizer::origNonlinIneqLowerBnds, SurrBasedMinimizer::origNonlinIneqUpperBnds, ParallelLibrary::output_manager(), Iterator::parallelLib, SurrBasedLocalMinimizer::recastSubProb, SurrBasedLocalMinimizer::relax_constraints(), ActiveSet::request_values(), SurrBasedLocalMinimizer::reset(), Iterator::response_results(), SurrBasedLocalMinimizer::responseCenterApprox, SurrBasedLocalMinimizer::responseCenterTruth, SurrBasedLocalMinimizer::responseStarApprox, SurrBasedLocalMinimizer::responseStarTruth, Iterator::run(), Iterator::sampling_scheme(), SurrBasedMinimizer::sbIterNum, SurrBasedLocalMinimizer::sblmInstance, SurrBasedLocalMinimizer::softConvCount, SurrBasedLocalMinimizer::softConvLimit, Model::subordinate_iterator(), Model::surrogate_model(), Model::surrogate_response_mode(), SurrBasedLocalMinimizer::tr_bounds(), SurrBasedLocalMinimizer::tr_ratio_check(), SurrBasedLocalMinimizer::trConstraintRelax, SurrBasedLocalMinimizer::trustRegionFactor, Model::truth_model(), SurrBasedLocalMinimizer::truthGradientFlag, SurrBasedLocalMinimizer::truthHessianFlag, Response::update(), SurrBasedLocalMinimizer::useDerivsFlag, Iterator::variables_results(), and SurrBasedLocalMinimizer::varsCenter.

void hard_convergence_check (const Response & *response_truth,* const RealVector & *c_vars,* const RealVector & *lower_bnds,* const RealVector & *upper_bnds*) `[private]`

check for hard convergence (norm of projected gradient of merit function near zero)

The hard convergence check computes the gradient of the merit function at the trust region center, performs a projection for active bound constraints (removing any gradient component directed into an active bound), and signals convergence if the 2-norm of this projected gradient is less than convergenceTol.

References SurrBasedLocalMinimizer::acceptLogic, SurrBasedLocalMinimizer::approxSubProbObj, SurrBasedMinimizer::constraint_violation(), Minimizer::constraintTol, SurrBasedLocalMinimizer::convergenceFlag, Iterator::convergenceTol, Response::function_gradients(), Response::function_values(), Iterator::iteratedModel, SurrBasedMinimizer::lagrangian_gradient(), SurrBasedLocalMinimizer::meritFnType, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, SurrBasedMinimizer::origNonlinEqTargets, SurrBasedMinimizer::origNonlinIneqLowerBnds, SurrBasedMinimizer::origNonlinIneqUpperBnds, Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), SurrBasedMinimizer::sbIterNum, SurrBasedLocalMinimizer::truthGradientFlag, SurrBasedMinimizer::update_augmented_lagrange_multipliers(), SurrBasedMinimizer::update_filter(), and SurrBasedMinimizer::update_lagrange_multipliers().

Referenced by SurrBasedLocalMinimizer::minimize_surrogates().

void tr_ratio_check (const RealVector & *c_vars_star,* const RealVector & *tr_lower_bnds,* const RealVector & *tr_upper_bnds*) `[private]`

compute trust region ratio (for SBLM iterate acceptance and trust region resizing) and check for soft convergence (diminishing returns)

Assess acceptance of SBLM iterate (trust region ratio or filter) and compute soft convergence metrics (number of consecutive failures, min trust region size, etc.) to assess whether the convergence rate has decreased to a point where the process should be terminated (diminishing returns).

References SurrBasedLocalMinimizer::acceptLogic, SurrBasedLocalMinimizer::approxSubProbObj, SurrBasedMinimizer::augmented_lagrangian_merit(), SurrBasedMinimizer::constraint_violation(), Minimizer::constraintTol, Iterator::convergenceTol, SurrBasedMinimizer::etaSequence, Response::function_values(), SurrBasedLocalMinimizer::gammaContract, SurrBasedLocalMinimizer::gammaExpand, SurrBasedLocalMinimizer::globalApproxFlag, Iterator::iteratedModel, SurrBasedMinimizer::lagrangian_merit(), SurrBasedLocalMinimizer::meritFnType, SurrBasedLocalMinimizer::newCenterFlag, Minimizer::numContinuousVars, SurrBasedMinimizer::origNonlinEqTargets, SurrBasedMinimizer::origNonlinIneqLowerBnds, SurrBasedMinimizer::origNonlinIneqUpperBnds, SurrBasedMinimizer::penalty_merit(), Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), SurrBasedLocalMinimizer::responseCenterApprox, SurrBasedLocalMinimizer::responseCenterTruth, SurrBasedLocalMinimizer::responseStarApprox, SurrBasedLocalMinimizer::responseStarTruth, SurrBasedLocalMinimizer::softConvCount, SurrBasedLocalMinimizer::trRatioContractValue, SurrBasedLocalMinimizer::trRatioExpandValue, SurrBasedLocalMinimizer::trustRegionFactor, SurrBasedMinimizer::update_augmented_lagrange_multipliers(), SurrBasedMinimizer::update_filter(), and SurrBasedLocalMinimizer::update_penalty().

Referenced by SurrBasedLocalMinimizer::minimize_surrogates().

void update_penalty (const RealVector & *fns_center_truth,* const RealVector & *fns_star_truth*) `[private]`

initialize and update the penaltyParameter

Scaling of the penalty value is important to avoid rejecting SBLM iterates which must increase the objective to achieve a reduction in constraint violation. In the basic penalty case, the penalty is ramped exponentially based on the iteration counter. In the adaptive case, the ratio of relative change between center and star points for the objective and constraint violation values is used to rescale penalty values.

References SurrBasedMinimizer::alphaEta, SurrBasedLocalMinimizer::approxSubProbObj, SurrBasedMinimizer::constraint_violation(), Minimizer::constraintTol, SurrBasedMinimizer::eta, SurrBasedMinimizer::etaSequence, Iterator::iteratedModel, SurrBasedLocalMinimizer::meritFnType, Minimizer::objective(), SurrBasedLocalMinimizer::penaltyIterOffset, SurrBasedMinimizer::penaltyParameter, Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), and SurrBasedMinimizer::sbIterNum.

Referenced by SurrBasedLocalMinimizer::tr_ratio_check().

void approx_subprob_objective_eval (const Variables & *surrogate_vars,* const Variables & *recast_vars,* const Response & *surrogate_response,* Response & *recast_response*) `[static], [private]`

static function used to define the approximate subproblem objective.

Objective functions evaluator for solution of approximate subproblem using a RecastModel.

References Response::active_set_request_vector(), SurrBasedLocalMinimizer::approxSubProbCon, SurrBasedLocalMinimizer::approxSubProbModel, SurrBasedLocalMinimizer::approxSubProbObj, SurrBasedMinimizer::augmented_lagrangian_gradient(), SurrBasedMinimizer::augmented_lagrangian_merit(), Response::function_gradient(), Response::function_gradient_view(), Response::function_gradients(), Response::function_value(), Response::function_values(), Iterator::iteratedModel, SurrBasedMinimizer::lagrangian_gradient(), SurrBasedMinimizer::lagrangian_merit(), Model::nonlinear_eq_constraint_targets(), Model::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Minimizer::numUserPrimaryFns, Minimizer::objective(), Minimizer::objective_gradient(), SurrBasedMinimizer::origNonlinEqTargets, SurrBasedMinimizer::origNonlinIneqLowerBnds, SurrBasedMinimizer::origNonlinIneqUpperBnds, Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), and SurrBasedLocalMinimizer::sblmInstance.

Referenced by SurrBasedLocalMinimizer::SurrBasedLocalMinimizer().

void approx_subprob_constraint_eval (const Variables & *surrogate_vars,* const Variables & *recast_vars,* const Response & *surrogate_response,* Response & *recast_response*) [static], [private]

static function used to define the approximate subproblem constraints.

Constraint functions evaluator for solution of approximate subproblem using a RecastModel.

References Response::active_set_derivative_vector(), Response::active_set_request_vector(), SurrBasedLocalMinimizer::approxSubProbCon, SurrBasedLocalMinimizer::approxSubProbObj, Variables::continuous_variables(), Response::function_gradient(), Response::function_gradient_view(), Response::function_gradients(), Response::function_value(), Response::function_values(), Minimizer::numUserPrimaryFns, SurrBasedLocalMinimizer::responseCenterApprox, SurrBasedLocalMinimizer::sblmInstance, and SurrBasedLocalMinimizer::varsCenter.

Referenced by SurrBasedLocalMinimizer::SurrBasedLocalMinimizer().

void hom_objective_eval (int & *mode,* int & *n,* double * *tau_and_x,* double & *f,* double * *grad_f,* int &) [static], [private]

static function used by NPSOL as the objective function in the homotopy constraint relaxation formulation.

NPSOL objective functions evaluator for solution of homotopy constraint relaxation parameter optimization. This constrained optimization problem performs the update of the tau parameter in the homotopy heuristic approach used to relax the constraints in the original problem .

Referenced by SurrBasedLocalMinimizer::relax_constraints().

void hom_constraint_eval (int & *mode,* int & *ncnln,* int & *n,* int & *nrowj,* int * *needc,* double * *tau_and_x,* double * *c,* double * *cjac,* int & *nstate*) [static], [private]

static function used by NPSOL as the constraint function in the homotopy constraint relaxation formulation.

NPSOL constraint functions evaluator for solution of homotopy constraint relaxation parameter optimization. This constrained optimization problem performs the update of the tau parameter in the homotopy heuristic approach used to relax the constraints in the original problem.

References Response::active_set(), SurrBasedLocalMinimizer::approxSubProbModel, Model::compute_response(), Model::continuous_variables(), Model::current_response(), Response::function_gradients(), Response::function_values(), SurrBasedLocalMinimizer::nonlinEqTargetsSlack, SurrBasedLocalMinimizer::nonlinIneqLowerBndsSlack, SurrBasedLocalMinimizer::nonlinIneqUpperBndsSlack, Model::num_functions(), Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, ActiveSet::request_vector(), SurrBasedLocalMinimizer::sblmInstance, and SurrBasedLocalMinimizer::tau.

Referenced by SurrBasedLocalMinimizer::relax_constraints().

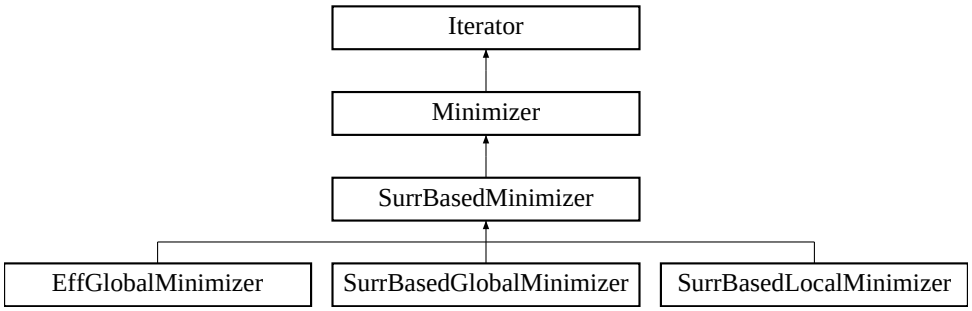
The documentation for this class was generated from the following files:

- SurrBasedLocalMinimizer.hpp
- SurrBasedLocalMinimizer.cpp

13.163 SurrBasedMinimizer Class Reference

Base class for local/global surrogate-based optimization/least squares.

Inheritance diagram for SurrBasedMinimizer:



Protected Member Functions

- SurrBasedMinimizer (ProblemDescDB &problem_db, Model &model)

  constructor
- ~SurrBasedMinimizer ()

  destructor
- void derived_init_communicators (ParLevLIter pl_iter)

  derived class contributions to initializing the communicators associated with this Iterator instance
- void derived_set_communicators (ParLevLIter pl_iter)

  derived class contributions to setting the communicators associated with this Iterator instance
- void derived_free_communicators (ParLevLIter pl_iter)

  derived class contributions to freeing the communicators associated with this Iterator instance
- void initialize_graphics (int iterator_server_id=1)

  initialize graphics customized for surrogate-based iteration
- void core_run ()

  core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post
- void print_results (std::ostream &s)
- virtual void minimize_surrogates ()=0

  Used for computing the optimal solution using a surrogate-based approach. Redefines the Iterator::core_run() virtual function.
- void update_lagrange_multipliers (const RealVector &fn_vals, const RealMatrix &fn_grads)

  initialize and update Lagrange multipliers for basic Lagrangian
- void update_augmented_lagrange_multipliers (const RealVector &fn_vals)

  initialize and update the Lagrange multipliers for augmented Lagrangian
- bool update_filter (const RealVector &fn_vals)

  update a filter from a set of function values
- Real lagrangian_merit (const RealVector &fn_vals, const BoolDeque &sense, const RealVector &primary_wts, const RealVector &nln_ineq_l_bnds, const RealVector &nln_ineq_u_bnds, const RealVector &nln_eq_tgts)

  compute a Lagrangian function from a set of function values
- void lagrangian_gradient (const RealVector &fn_vals, const RealMatrix &fn_grads, const BoolDeque &sense, const RealVector &primary_wts, const RealVector &nln_ineq_l_bnds, const RealVector &nln_ineq_u_bnds, const RealVector &nln_eq_tgts, RealVector &lag_grad)

  compute the gradient of the Lagrangian function

- Real augmented_lagrangian_merit (const RealVector &fn_vals, const BoolDeque &sense, const RealVector &primary_wts, const RealVector &nln_ineq_l_bnds, const RealVector &nln_ineq_u_bnds, const RealVector &nln_eq_tgts)

  compute an augmented Lagrangian function from a set of function values
- void augmented_lagrangian_gradient (const RealVector &fn_vals, const RealMatrix &fn_grads, const BoolDeque &sense, const RealVector &primary_wts, const RealVector &nln_ineq_l_bnds, const RealVector &nln_ineq_u_bnds, const RealVector &nln_eq_tgts, RealVector &alag_grad)

  compute the gradient of the augmented Lagrangian function
- Real penalty_merit (const RealVector &fn_vals, const BoolDeque &sense, const RealVector &primary_wts)

  compute a penalty function from a set of function values
- void penalty_gradient (const RealVector &fn_vals, const RealMatrix &fn_grads, const BoolDeque &sense, const RealVector &primary_wts, RealVector &pen_grad)

  compute the gradient of the penalty function
- Real constraint_violation (const RealVector &fn_vals, const Real &constraint_tol)

  compute the constraint violation from a set of function values

Protected Attributes

- Iterator approxSubProbMinimizer

  the minimizer used on the surrogate model to solve the approximate subproblem on each surrogate-based iteration
- int sbIterNum

  surrogate-based minimization iteration number
- RealVectorList sbFilter

  Set of response function vectors defining a filter (objective vs. constraint violation) for iterate selection/rejection.
- RealVector lagrangeMult

  Lagrange multipliers for basic Lagrangian calculations.
- RealVector augLagrangeMult

  Lagrange multipliers for augmented Lagrangian calculations.
- Real penaltyParameter

  the penalization factor for violated constraints used in quadratic penalty calculations; increased in update_penalty()
- RealVector origNonlinIneqLowerBnds

  original nonlinear inequality constraint lower bounds (no relaxation)
- RealVector origNonlinIneqUpperBnds

  original nonlinear inequality constraint upper bounds (no relaxation)
- RealVector origNonlinEqTargets

  original nonlinear equality constraint targets (no relaxation)
- Real eta

  constant used in etaSequence updates
- Real alphaEta

  power for etaSequence updates when updating penalty
- Real betaEta

  power for etaSequence updates when updating multipliers
- Real etaSequence

decreasing sequence of allowable constraint violation used in augmented Lagrangian updates (refer to Conn, Gould, and Toint, section 14.4)

- size_t miPLIndex

  index for the active ParallelLevel within ParallelConfiguration::miPLIters

Additional Inherited Members

13.163.1 Detailed Description

Base class for local/global surrogate-based optimization/least squares.

These minimizers use a SurrogateModel to perform optimization based either on local trust region methods or global updating methods.

13.163.2 Member Function Documentation

void core_run ( ) [inline], [protected], [virtual]

core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post

Virtual run function for the iterator class hierarchy. All derived classes need to redefine it.

Reimplemented from Iterator.

References SurrBasedMinimizer::minimize_surrogates().

void print_results (std::ostream & *s*) [protected], [virtual]

Redefines default iterator results printing to include optimization results (objective functions and constraints).

Reimplemented from Iterator.

References Dakota::abort_handler(), Iterator::activeSet, Minimizer::archive_allocate_best(), Minimizer::archive_best(), Iterator::bestResponseArray, Iterator::bestVariablesArray, Dakota::data_pairs, Model::interface_id(), Iterator::iteratedModel, Dakota::lookup_by_val(), Iterator::methodName, Minimizer::numFunctions, Minimizer::numUserPrimaryFns, Minimizer::optimizationFlag, ActiveSet::request_values(), and Model::truth_model().

void update_lagrange_multipliers (const RealVector & *fn_vals*, const RealMatrix & *fn_grads*) [protected]

initialize and update Lagrange multipliers for basic Lagrangian

For the Rockafellar augmented Lagrangian, simple Lagrange multiplier updates are available which do not require the active constraint gradients. For the basic Lagrangian, Lagrange multipliers are estimated through solution of a nonnegative linear least squares problem.

References Dakota::abort_handler(), Minimizer::bigRealBoundSize, Minimizer::constraintTol, Iterator::iteratedModel, SurrBasedMinimizer::lagrangeMult, Minimizer::numContinuousVars, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, Minimizer::numUserPrimaryFns, Minimizer::objective_gradient(), SurrBasedMinimizer::origNonlinIneqLowerBnds, SurrBasedMinimizer::origNonlinIneqUpperBnds, Model::primary_response_fn_sense(), and Model::primary_response_fn_weights().

Referenced by SurrBasedLocalMinimizer::hard_convergence_check().

void update_augmented_lagrange_multipliers (const RealVector & *fn_vals*) [protected]

initialize and update the Lagrange multipliers for augmented Lagrangian

For the Rockafellar augmented Lagrangian, simple Lagrange multiplier updates are available which do not require the active constraint gradients. For the basic Lagrangian, Lagrange multipliers are estimated through solution of a nonnegative linear least squares problem.

References SurrBasedMinimizer::augLagrangeMult, SurrBasedMinimizer::betaEta, Minimizer::bigRealBoundSize, SurrBasedMinimizer::etaSequence, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, Minimizer::numUserPrimaryFns, SurrBasedMinimizer::origNonlinEqTargets, SurrBasedMinimizer::origNonlinIneqLowerBnds, SurrBasedMinimizer::origNonlinIneqUpperBnds, and SurrBasedMinimizer::penaltyParameter.

Referenced by SurrBasedLocalMinimizer::hard_convergence_check(), EffGlobalMinimizer::minimize_surrogates_on_model(), and SurrBasedLocalMinimizer::tr_ratio_check().

bool update_filter (const RealVector & *fn_vals*) [protected]

update a filter from a set of function values

Update the sbFilter with fn_vals if new iterate is non-dominated.

References SurrBasedMinimizer::constraint_violation(), Iterator::iteratedModel, Minimizer::numNonlinearConstraints, Minimizer::objective(), Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), and SurrBasedMinimizer::sbFilter.

Referenced by SurrBasedLocalMinimizer::hard_convergence_check(), and SurrBasedLocalMinimizer::tr_ratio_check().

Real lagrangian_merit (const RealVector & *fn_vals,* const BoolDeque & *sense,* const RealVector & *primary_wts,* const RealVector & *nln_ineq_l_bnds,* const RealVector & *nln_ineq_u_bnds,* const RealVector & *nln_eq_tgts*) [protected]

compute a Lagrangian function from a set of function values

The Lagrangian function computation sums the objective function and the Lagrange multipler terms for inequality/equality constraints. This implementation follows the convention in Vanderplaats with g<=0 and h=0. The bounds/targets passed in may reflect the original constraints or the relaxed constraints.

References Minimizer::bigRealBoundSize, Minimizer::constraintTol, SurrBasedMinimizer::lagrangeMult, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, Minimizer::numUserPrimaryFns, and Minimizer::objective().

Referenced by SurrBasedLocalMinimizer::approx_subprob_objective_eval(), and SurrBasedLocalMinimizer::tr_ratio_check().

Real augmented_lagrangian_merit (const RealVector & *fn_vals,* const BoolDeque & *sense,* const RealVector & *primary_wts,* const RealVector & *nln_ineq_l_bnds,* const RealVector & *nln_ineq_u_bnds,* const RealVector & *nln_eq_tgts*) [protected]

compute an augmented Lagrangian function from a set of function values

The Rockafellar augmented Lagrangian function sums the objective function, Lagrange multipler terms for inequality/equality constraints, and quadratic penalty terms for inequality/equality constraints. This implementation follows the convention in Vanderplaats with g<=0 and h=0. The bounds/targets passed in may reflect the original constraints or the relaxed constraints.

References SurrBasedMinimizer::augLagrangeMult, Minimizer::bigRealBoundSize, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, Minimizer::numUserPrimaryFns, Minimizer::objective(), and SurrBasedMinimizer::penaltyParameter.

Referenced by SurrBasedLocalMinimizer::approx_subprob_objective_eval(), EffGlobalMinimizer::get_best_sample(), EffGlobalMinimizer::minimize_surrogates_on_model(), and SurrBasedLocalMinimizer::tr_ratio_check().

Real penalty_merit (const RealVector & *fn_vals,* const BoolDeque & *sense,* const RealVector & *primary_wts*) [protected]

compute a penalty function from a set of function values

The penalty function computation applies a quadratic penalty to any constraint violations and adds this to the objective function(s) p = f + r_p cv.

References SurrBasedMinimizer::constraint_violation(), Minimizer::constraintTol, Minimizer::objective(), and SurrBasedMinimizer::penaltyParameter.

Referenced by SurrBasedLocalMinimizer::tr_ratio_check().

Real constraint_violation (const RealVector & *fn_vals,* const Real & *constraint_tol*) `[protected]`

compute the constraint violation from a set of function values

Compute the quadratic constraint violation defined as cv = g+^T g+

- h+^T h+. This implementation supports equality constraints and 2-sided inequalities. The constraint_tol allows for a small constraint infeasibility (used for penalty methods, but not Lagrangian methods).

References Minimizer::bigRealBoundSize, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, Minimizer::numUserPrimaryFns, SurrBasedMinimizer::origNonlinEqTargets, SurrBasedMinimizer::origNonlinIneqLowerBnds, and SurrBasedMinimizer::origNonlinIneqUpperBnds.

Referenced by SurrBasedLocalMinimizer::hard_convergence_check(), EffGlobalMinimizer::minimize_surrogates_on_model(), SurrBasedMinimizer::penalty_merit(), SurrBasedLocalMinimizer::relax_constraints(), SurrBasedLocalMinimizer::tr_ratio_check(), SurrBasedMinimizer::update_filter(), and SurrBasedLocalMinimizer::update_penalty().

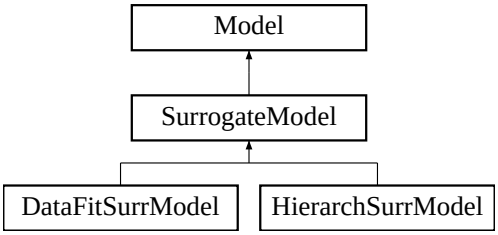
The documentation for this class was generated from the following files:

- SurrBasedMinimizer.hpp
- SurrBasedMinimizer.cpp

13.164 SurrogateModel Class Reference

Base class for surrogate models (DataFitSurrModel and HierarchSurrModel).

Inheritance diagram for SurrogateModel:



Protected Member Functions

- SurrogateModel (ProblemDescDB &problem_db)

  constructor
- SurrogateModel (ProblemDescDB &problem_db, ParallelLibrary &parallel_lib, const SharedVariablesData &svd, const ActiveSet &set, short output_level)

  alternate constructor
- ~SurrogateModel ()

  destructor
- Model & subordinate_model ()

return truth_model()

- short surrogate_response_mode () const

  return responseMode

- size_t mi_parallel_level_index () const

  return miPLIndex

- DiscrepancyCorrection & discrepancy_correction ()

  return deltaCorr

- void check_submodel_compatibility (const Model &sub_model)

  verify compatibility between SurrogateModel attributes and attributes of the submodel (DataFitSurrModel::actualModel or HierarchSurrModel::highFidelityModel)

- bool force_rebuild ()

  evaluate whether a rebuild of the approximation should be forced based on changes in the inactive data

- void asv_mapping (const ShortArray &orig_asv, ShortArray &actual_asv, ShortArray &approx_asv, bool build_flag)

  distributes the incoming orig_asv among actual_asv and approx_asv

- void asv_mapping (const ShortArray &actual_asv, const ShortArray &approx_asv, ShortArray &combined_asv)

  reconstitutes a combined_asv from actual_asv and approx_asv

- void response_mapping (const Response &actual_response, const Response &approx_response, Response &combined_response)

  overlays actual_response and approx_response to update combined_response

Protected Attributes

- IntSet surrogateFnIndices

  for mixed response sets, this array specifies the response function subset that is approximated

- IntResponseMap surrResponseMap

  map of surrogate responses used in derived_synchronize() and derived_synchronize_nowait() functions

- IntVariablesMap rawVarsMap

  map of raw continuous variables used by apply_correction(). Model::varsList cannot be used for this purpose since it does not contain lower level variables sets from finite differencing.

- IntIntMap truthIdMap

  map from actualModel/highFidelityModel evaluation ids to DataFitSurrModel.hppierarchSurrModel ids

- IntIntMap surrIdMap

  map from approxInterface/lowFidelityModel evaluation ids to DataFitSurrModel.hppierarchSurrModel ids

- IntResponseMap cachedApproxRespMap

  map of approximate responses retrieved in derived_synchronize_nowait() that could not be returned since corresponding truth model response portions were still pending.

- short responseMode

  an enumeration that controls the response calculation mode in {DataFit,Hierarch}SurrModel approximate response computations

- size_t approxBuilds

  number of calls to build_approximation()

- size_t miPLIndex

the index of the active metaiterator-iterator parallelism level (corresponding to ParallelConfiguration::miPLIters) used at runtime

- RealVector referenceCLBnds

  stores a reference copy of active continuous lower bounds when the approximation is built; used to detect when a rebuild is required.
- RealVector referenceCUBnds

  stores a reference copy of active continuous upper bounds when the approximation is built; used to detect when a rebuild is required.
- IntVector referenceDILBnds

  stores a reference copy of active discrete int lower bounds when the approximation is built; used to detect when a rebuild is required.
- IntVector referenceDIUBnds

  stores a reference copy of active discrete int upper bounds when the approximation is built; used to detect when a rebuild is required.
- RealVector referenceDRLBnds

  stores a reference copy of active discrete real lower bounds when the approximation is built; used to detect when a rebuild is required.
- RealVector referenceDRUBnds

  stores a reference copy of active discrete real upper bounds when the approximation is built; used to detect when a rebuild is required.
- RealVector referenceICVars

  stores a reference copy of the inactive continuous variables when the approximation is built using a Distinct view; used to detect when a rebuild is required.
- IntVector referenceIDIVars

  stores a reference copy of the inactive discrete int variables when the approximation is built using a Distinct view; used to detect when a rebuild is required.
- RealVector referenceIDRVars

  stores a reference copy of the inactive discrete real variables when the approximation is built using a Distinct view; used to detect when a rebuild is required.
- DiscrepancyCorrection deltaCorr

  manages construction and application of correction functions that are applied to a surrogate model (DataFitSurr or HierarchSurr) in order to reproduce high fidelity data.

Private Attributes

- Variables truthModelVars

  copy of the truth model variables object used to simplify conversion among differing variable views in force_rebuild()
- Constraints truthModelCons

  copy of the truth model constraints object used to simplify conversion among differing variable views in force_rebuild()

Additional Inherited Members

13.164.1 Detailed Description

Base class for surrogate models (DataFitSurrModel and HierarchSurrModel).

The SurrogateModel class provides common functions to derived classes for computing and applying corrections to approximations.

13.164.2 Member Function Documentation

bool force_rebuild ( ) [protected], [virtual]

evaluate whether a rebuild of the approximation should be forced based on changes in the inactive data

This function forces a rebuild of the approximation according to the sub-model variables view, the approximation type, and whether the active approximation bounds or inactive variable values have changed since the last approximation build.

Reimplemented from Model.

References Constraints::all_continuous_lower_bounds(), Constraints::all_continuous_upper_bounds(), Variables::all_continuous_variables(), Constraints::all_discrete_int_lower_bounds(), Constraints::all_discrete_int_upper_bounds(), Variables::all_discrete_int_variables(), Constraints::all_discrete_real_lower_bounds(), Constraints::all_discrete_real_upper_bounds(), Variables::all_discrete_real_variables(), Constraints::continuous_lower_bounds(), Model::continuous_lower_bounds(), Constraints::continuous_upper_bounds(), Model::continuous_upper_bounds(), Variables::continuous_variables(), Constraints::copy(), Variables::copy(), Model::current_variables(), Model::currentVariables, Constraints::discrete_int_lower_bounds(), Model::discrete_int_lower_bounds(), Constraints::discrete_int_upper_bounds(), Model::discrete_int_upper_bounds(), Variables::discrete_int_variables(), Constraints::discrete_real_lower_bounds(), Model::discrete_real_lower_bounds(), Constraints::discrete_real_upper_bounds(), Model::discrete_real_upper_bounds(), Variables::discrete_real_variables(), Variables::inactive_continuous_variables(), Variables::inactive_discrete_int_variables(), Variables::inactive_discrete_real_variables(), Constraints::is_null(), Variables::is_null(), Model::is_null(), Model::model_type(), SurrogateModel::referenceCLBnds, SurrogateModel::referenceCUBnds, SurrogateModel::referenceDILBnds, SurrogateModel::referenceDIUBnds, SurrogateModel::referenceDRLBnds, SurrogateModel::referenceDRUBnds, SurrogateModel::referenceICVars, SurrogateModel::referenceIDIVars, SurrogateModel::referenceIDRVars, Dakota::strbegins(), Model::subordinate_model(), Model::surrogateType, Model::truth_model(), SurrogateModel::truthModelCons, SurrogateModel::truthModelVars, Model::user_defined_constraints(), Model::userDefinedConstraints, and Variables::view().

Referenced by HierarchSurrModel::derived_asynch_compute_response(), DataFitSurrModel::derived_asynch_compute_response(), HierarchSurrModel::derived_compute_response(), and DataFitSurrModel::derived_compute_response().

13.164.3 Member Data Documentation

short responseMode [protected]

an enumeration that controls the response calculation mode in {DataFit,Hierarch}SurrModel approximate response computations

SurrBasedLocalMinimizer toggles this mode since compute_correction() does not back out old corrections.

Referenced by HierarchSurrModel::derived_asynch_compute_response(), DataFitSurrModel::derived_asynch_compute_response(), HierarchSurrModel::derived_compute_response(), DataFitSurrModel::derived_compute_response(), HierarchSurrModel::derived_set_communicators(), HierarchSurrModel::derived_synchronize(), DataFitSurrModel::derived_synchronize(), HierarchSurrModel::derived_synchronize_nowait(), DataFitSurrModel::derived_synchronize_nowait(), SurrogateModel::surrogate_response_mode(), HierarchSurrModel::surrogate_response_mode(), and DataFitSurrModel::surrogate_response_mode().

size_t approxBuilds [protected]

number of calls to build_approximation()

used as a flag to automatically build the approximation if one of the derived compute_response functions is called prior to build_approximation().

Referenced by DataFitSurrModel::append_approximation(), DataFitSurrModel::approximation_coefficients(), HierarchSurrModel::build_approximation(), DataFitSurrModel::build_approximation(), HierarchSurrModel::derived-

_asynch_compute_response(), DataFitSurrModel::derived_asynch_compute_response(), HierarchSurrModel::derived_compute_response(), DataFitSurrModel::derived_compute_response(), DataFitSurrModel::pop_approximation(), and DataFitSurrModel::update_approximation().

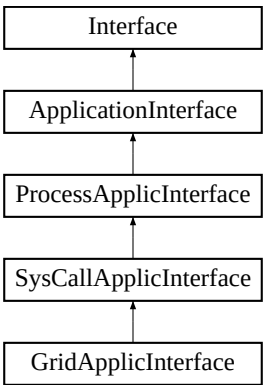
The documentation for this class was generated from the following files:

- SurrogateModel.hpp
- SurrogateModel.cpp

13.165 SysCallApplicInterface Class Reference

Derived application interface class which spawns simulation codes using system calls.

Inheritance diagram for SysCallApplicInterface:



Public Member Functions

- SysCallApplicInterface (const ProblemDescDB &problem_db)

  constructor
- ~SysCallApplicInterface ()

  destructor

Protected Member Functions

- void wait_local_evaluations (PRPQueue &prp_queue)
- void test_local_evaluations (PRPQueue &prp_queue)
- int synchronous_local_analysis (int analysis_id)
- void init_communicators_checks (int max_eval_concurrency)
- void set_communicators_checks (int max_eval_concurrency)
- void map_bookkeeping (pid_t pid, int fn_eval_id)

  bookkeeping of process and evaluation ids for asynchronous maps
- pid_t create_evaluation_process (bool block_flag)

  Spawn the evaluation by managing the input filter, analysis drivers, and output filter. Called from derived_map() & derived_map_asynch().

Private Member Functions

- bool system_call_file_test (const bfs::path &root_file)

  detect completion of a function evaluation through existence of the necessary results file(s)
- void spawn_evaluation_to_shell (bool block_flag)

  spawn a complete function evaluation
- void spawn_input_filter_to_shell (bool block_flag)

  spawn the input filter portion of a function evaluation
- void spawn_analysis_to_shell (int analysis_id, bool block_flag)

  spawn a single analysis as part of a function evaluation
- void spawn_output_filter_to_shell (bool block_flag)

  spawn the output filter portion of a function evaluation

Private Attributes

- IntSet sysCallSet

  set of function evaluation id's for active asynchronous system call evaluations
- IntShortMap failCountMap

  map linking function evaluation id's to number of response read failures

Additional Inherited Members

13.165.1 Detailed Description

Derived application interface class which spawns simulation codes using system calls.

system() is part of the C API and can be used on both Windows and Unix systems.

13.165.2 Member Function Documentation

void wait_local_evaluations (PRPQueue & *prp_queue*) `[inline], [protected], [virtual]`

Check for completion of active asynch jobs (tracked with sysCallSet). Wait for at least one completion and complete all jobs that have returned. This satisifies a "fairness" principle, in the sense that a completed job will *always* be processed (whereas accepting only a single completion could always accept the same completion - the case of very inexpensive fn. evals. - and starve some servers).

Reimplemented from ApplicationInterface.

References ApplicationInterface::completionSet, and SysCallApplicInterface::test_local_evaluations().

void test_local_evaluations (PRPQueue & *prp_queue*) `[protected], [virtual]`

Check for completion of active asynch jobs (tracked with sysCallSet). Make one pass through sysCallSet & complete all jobs that have returned.

Reimplemented from ApplicationInterface.

References Dakota::abort_handler(), Response::active_set(), ApplicationInterface::completionSet, SysCallApplicInterface::failCountMap, ProcessApplicInterface::fileNameMap, Interface::final_eval_id_tag(), Dakota::lookup_by_eval_id(), ApplicationInterface::manage_failure(), ProcessApplicInterface::read_results_files(), SysCallApplicInterface::sysCallSet, and SysCallApplicInterface::system_call_file_test().

Referenced by SysCallApplicInterface::wait_local_evaluations().

int synchronous_local_analysis (int *analysis_id*) [inline], [protected], [virtual]

This code provides the derived function used by ApplicationInterface::serve_analyses_synch().

Reimplemented from ApplicationInterface.

References SysCallApplicInterface::spawn_analysis_to_shell().

void init_communicators_checks (int *max_eval_concurrency*) [inline], [protected], [virtual]

No derived interface plug-ins, so perform construct-time checks. However, process init issues as warnings since some contexts (e.g., HierarchSurrModel) initialize more configurations than will be used.

Reimplemented from ApplicationInterface.

References ApplicationInterface::check_multiprocessor_analysis().

void set_communicators_checks (int *max_eval_concurrency*) [inline], [protected], [virtual]

Process run-time issues as hard errors.

Reimplemented from ApplicationInterface.

References Dakota::abort_handler(), and ApplicationInterface::check_multiprocessor_analysis().

void spawn_evaluation_to_shell (bool *block_flag*) [private]

spawn a complete function evaluation

Put the SysCallApplicInterface to the shell. This function is used when all portions of the function evaluation (i.e., all analysis drivers) are executed on the local processor.

References CommandShell::asynch_flag(), ProcessApplicInterface::commandLineArgs, ProcessApplicInterface::curWorkdir, Dakota::flush(), ProcessApplicInterface::iFilterName, ProcessApplicInterface::multipleParamsFiles, ProcessApplicInterface::oFilterName, ProcessApplicInterface::paramsFileName, ProcessApplicInterface::prepare_process_environment(), ProcessApplicInterface::programNames, ProcessApplicInterface::reset_process_environment(), ProcessApplicInterface::resultsFileName, CommandShell::suppress_output_flag(), ApplicationInterface::suppressOutput, and ProcessApplicInterface::useWorkdir.

Referenced by SysCallApplicInterface::create_evaluation_process().

void spawn_input_filter_to_shell (bool *block_flag*) [private]

spawn the input filter portion of a function evaluation

Put the input filter to the shell. This function is used when multiple analysis drivers are spread between processors. No need to check for a Null input filter, as this is checked externally. Use of nonblocking shells is supported in this fn, although its use is currently prevented externally.

References CommandShell::asynch_flag(), ProcessApplicInterface::commandLineArgs, Dakota::flush(), ProcessApplicInterface::iFilterName, ProcessApplicInterface::paramsFileName, ProcessApplicInterface::prepare_process_environment(), ProcessApplicInterface::reset_process_environment(), ProcessApplicInterface::resultsFileName, CommandShell::suppress_output_flag(), and ApplicationInterface::suppressOutput.

Referenced by SysCallApplicInterface::create_evaluation_process().

void spawn_analysis_to_shell (int *analysis_id*, bool *block_flag*) [private]

spawn a single analysis as part of a function evaluation

Put a single analysis to the shell. This function is used when multiple analysis drivers are spread between processors. Use of nonblocking shells is supported in this fn, although its use is currently prevented externally.

References CommandShell::asynch_flag(), ProcessApplicInterface::commandLineArgs, Dakota::flush(), ProcessApplicInterface::multipleParamsFiles, ProcessApplicInterface::paramsFileName, ProcessApplicInterface::prepare_process_environment(), ProcessApplicInterface::programNames, ProcessApplicInterface::reset_process_environment(), ProcessApplicInterface::resultsFileName, CommandShell::suppress_output_flag(), and ApplicationInterface::suppressOutput.

Referenced by SysCallApplicInterface::create_evaluation_process(), SysCallApplicInterface::synchronous_local_analysis(), and GridApplicInterface::synchronous_local_analysis().

void spawn_output_filter_to_shell (bool *block_flag*) `[private]`

spawn the output filter portion of a function evaluation

Put the output filter to the shell. This function is used when multiple analysis drivers are spread between processors. No need to check for a Null output filter, as this is checked externally. Use of nonblocking shells is supported in this fn, although its use is currently prevented externally.

References CommandShell::asynch_flag(), ProcessApplicInterface::commandLineArgs, Dakota::flush(), ProcessApplicInterface::oFilterName, ProcessApplicInterface::paramsFileName, ProcessApplicInterface::prepare_process_environment(), ProcessApplicInterface::reset_process_environment(), ProcessApplicInterface::resultsFileName, CommandShell::suppress_output_flag(), and ApplicationInterface::suppressOutput.

Referenced by SysCallApplicInterface::create_evaluation_process().

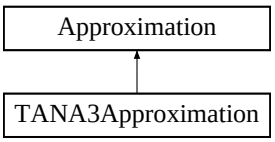
The documentation for this class was generated from the following files:

- SysCallApplicInterface.hpp
- SysCallApplicInterface.cpp

13.166 TANA3Approximation Class Reference

Derived approximation class for TANA-3 two-point exponential approximation (a multipoint approximation).

Inheritance diagram for TANA3Approximation:



Public Member Functions

- TANA3Approximation ()

  default constructor
- TANA3Approximation (ProblemDescDB &problem_db, const SharedApproxData &shared_data)

  standard constructor
- TANA3Approximation (const SharedApproxData &shared_data)

  alternate constructor
- ~TANA3Approximation ()

  destructor

Protected Member Functions

- int min_coefficients () const

  return the minimum number of samples (unknowns) required to build the derived class approximation type in numVars dimensions
- int num_constraints () const

  return the number of constraints to be enforced via an anchor point
- void build ()

  builds the approximation from scratch
- Real value (const Variables &vars)

  retrieve the approximate function value for a given parameter vector
- const RealVector & gradient (const Variables &vars)

  retrieve the approximate function gradient for a given parameter vector
- void clear_current ()

Private Member Functions

- void find_scaled_coefficients ()

  compute TANA coefficients based on scaled inputs
- void offset (const RealVector &x, RealVector &s)

  based on minX, apply offset scaling to x to define s

Private Attributes

- RealVector pExp

  vector of exponent values
- RealVector minX

  vector of minimum parameter values used in scaling
- RealVector scX1

  vector of scaled x1 values
- RealVector scX2

  vector of scaled x2 values
- Real H

  the scalar Hessian value in the TANA-3 approximation

Additional Inherited Members

13.166.1 Detailed Description

Derived approximation class for TANA-3 two-point exponential approximation (a multipoint approximation).

The TANA3Approximation class provides a multipoint approximation based on matching value and gradient data from two points (typically the current and previous iterates) in parameter space. It forms an exponential approximation in terms of intervening variables.

13.166.2 Member Function Documentation

void build ( ) [protected], [virtual]

builds the approximation from scratch

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented from Approximation.

References Dakota::abort_handler(), Approximation::approxData, Approximation::build(), TANA3Approximation::find_scaled_coefficients(), TANA3Approximation::minX, SharedApproxData::numVars, TANA3Approximation::pExp, and Approximation::sharedDataRep.

void clear_current ( ) [inline], [protected], [virtual]

Redefine default implementation to support history mechanism.

Reimplemented from Approximation.

References Approximation::approxData.

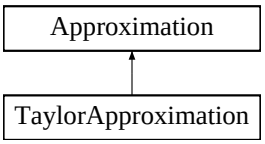
The documentation for this class was generated from the following files:

- TANA3Approximation.hpp
- TANA3Approximation.cpp

13.167 TaylorApproximation Class Reference

Derived approximation class for first- or second-order Taylor series (a local approximation).

Inheritance diagram for TaylorApproximation:



Public Member Functions

- TaylorApproximation ()

  default constructor

- TaylorApproximation (ProblemDescDB &problem_db, const SharedApproxData &shared_data)

  standard constructor

- TaylorApproximation (const SharedApproxData &shared_data)

  alternate constructor

- ~TaylorApproximation ()

  destructor

Protected Member Functions

- int min_coefficients () const

  return the minimum number of samples (unknowns) required to build the derived class approximation type in numVars dimensions

- void build ()

  builds the approximation from scratch

- Real value (const Variables &vars)

  retrieve the approximate function value for a given parameter vector

- const RealVector & gradient (const Variables &vars)

  retrieve the approximate function gradient for a given parameter vector

- const RealSymMatrix & hessian (const Variables &vars)

  retrieve the approximate function Hessian for a given parameter vector

Additional Inherited Members

13.167.1 Detailed Description

Derived approximation class for first- or second-order Taylor series (a local approximation).

The TaylorApproximation class provides a local approximation based on data from a single point in parameter space. It uses a zeroth-, first- or second-order Taylor series expansion: f(x) = f(x_c) for zeroth-order, plus grad(x_c)' (x - x_c) for first- and second-order, and plus (x - x_c)' Hess(x_c) (x - x_c) / 2 for second-order.

13.167.2 Member Function Documentation

void build ( ) [protected], [virtual]

builds the approximation from scratch

This is the common base class portion of the virtual fn and is insufficient on its own; derived implementations should explicitly invoke (or reimplement) this base class contribution.

Reimplemented from Approximation.

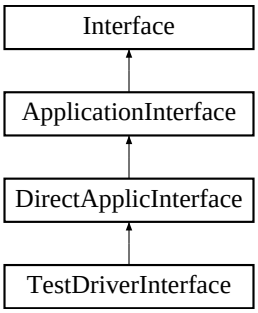
References Dakota::abort_handler(), Approximation::approxData, Approximation::build(), SharedApproxData::buildDataOrder, SharedApproxData::numVars, and Approximation::sharedDataRep.

The documentation for this class was generated from the following files:

- TaylorApproximation.hpp
- TaylorApproximation.cpp

13.168 TestDriverInterface Class Reference

Inheritance diagram for TestDriverInterface:



Public Member Functions

- TestDriverInterface (const ProblemDescDB &problem_db)

  constructor
- ~TestDriverInterface ()

  destructor

Protected Member Functions

- virtual int derived_map_ac (const Dakota::String &ac_name)

  execute an analysis code portion of a direct evaluation invocation

Private Member Functions

- int cantilever ()

  scaled cantilever test function for optimization
- int mod_cantilever ()

  unscaled cantilever test function for UQ
- int cyl_head ()

  the cylinder head constrained optimization test fn
- int multimodal ()

  multimodal UQ test function
- int log_ratio ()

  the log_ratio UQ test function
- int short_column ()

  the short_column UQ/OUU test function
- int lf_short_column ()

  a low fidelity short_column test function
- int mf_short_column ()

  alternate short_column formulations for < multifidelity or model form studies
- int alternate_short_column_forms (int form)

  helper fn for alternate forms
- int side_impact_cost ()

  the side_impact_cost UQ/OUU test function

- int side_impact_perf ()

  the side_impact_perf UQ/OUU test function
- int rosenbrock ()

  the Rosenbrock optimization and least squares test fn
- int generalized_rosenbrock ()

  n-dimensional Rosenbrock (Schittkowski)
- int extended_rosenbrock ()

  n-dimensional Rosenbrock (Nocedal/Wright)
- int lf_rosenbrock ()

  a low fidelity version of the Rosenbrock function
- int mf_rosenbrock ()

  alternate Rosenbrock formulations for < multifidelity or model form studies
- int gerstner ()

  the isotropic/anisotropic Gerstner test function family
- int scalable_gerstner ()

  scalable versions of the Gerstner test family
- void get_genz_coefficients (int num_dims, Real factor, int c_type, RealVector &c, RealVector &w)

  define coefficients needs for genz model
- int genz ()

  scalable test functions from the Genz test suite
- int steel_column_cost ()

  the steel_column_cost UQ/OUU test function
- int steel_column_perf ()

  the steel_column_perf UQ/OUU test function
- int sobol_rational ()

  Sobol SA rational test function.
- int sobol_g_function ()

  Sobol SA discontinuous test function.
- int sobol_ishigami ()

  Sobol SA transcendental test function.
- int text_book ()

  the text_book constrained optimization test function
- int text_book1 ()

  portion of text_book() evaluating the objective fn
- int text_book2 ()

  portion of text_book() evaluating constraint 1
- int text_book3 ()

  portion of text_book() evaluating constraint 2
- int text_book_ouu ()

  the text_book_ouu OUU test function
- int scalable_text_book ()

  scalable version of the text_book test function
- int scalable_monomials ()

simple monomials for UQ exactness testing

- void herbie1D (size_t der_mode, Real xc_loc, std::vector< Real > &w_and_ders)

  1D components of herbie function
- void smooth_herbie1D (size_t der_mode, Real xc_loc, std::vector< Real > &w_and_ders)

  1D components of smooth_herbie function
- void shubert1D (size_t der_mode, Real xc_loc, std::vector< Real > &w_and_ders)

  1D components of shubert function
- int herbie ()

  returns the N-D herbie function
- int smooth_herbie ()

  returns the N-D smooth herbie function
- int shubert ()

  returns the N-D shubert function
- void separable_combine (Real mult_scale_factor, std::vector< Real > &w, std::vector< Real > &d1w, std::vector< Real > &d2w)

  utility to combine components of separable fns
- int salinas ()

  direct interface to the SALINAS structural dynamics code
- int mc_api_run ()

  direct interface to ModelCenter via API, HKIM 4/3/03

Additional Inherited Members

13.168.1 Detailed Description

Specialization of DirectApplicInterface to embed algebraic test function drivers directly in Dakota

13.168.2 Member Function Documentation

int derived_map_ac (const Dakota::String & *ac_name*) `[protected], [virtual]`

execute an analysis code portion of a direct evaluation invocation

Derived map to evaluate a particular built-in test analysis function

Reimplemented from DirectApplicInterface.

References Dakota::abort_handler(), ApplicationInterface::analysisServerId, TestDriverInterface::cantilever(), TestDriverInterface::cyl_head(), DirectApplicInterface::driverTypeMap, TestDriverInterface::extended_rosenbrock(), TestDriverInterface::generalized_rosenbrock(), TestDriverInterface::genz(), TestDriverInterface::gerstner(), TestDriverInterface::herbie(), TestDriverInterface::lf_rosenbrock(), TestDriverInterface::lf_short_column(), TestDriverInterface::log_ratio(), TestDriverInterface::mc_api_run(), TestDriverInterface::mf_rosenbrock(), TestDriverInterface::mf_short_column(), TestDriverInterface::mod_cantilever(), TestDriverInterface::multimodal(), TestDriverInterface::rosenbrock(), TestDriverInterface::salinas(), TestDriverInterface::scalable_gerstner(), TestDriverInterface::scalable_monomials(), TestDriverInterface::scalable_text_book(), TestDriverInterface::short_column(), TestDriverInterface::shubert(), TestDriverInterface::side_impact_cost(), TestDriverInterface::side_impact_perf(), TestDriverInterface::smooth_herbie(), TestDriverInterface::sobol_g_function(), TestDriverInterface::sobol_ishigami(), TestDriverInterface::sobol_rational(), TestDriverInterface::steel_column_cost(), TestDriverInterface::steel_column_perf(), TestDriverInterface::text_book(), TestDriverInterface::text_book1(), TestDriverInterface::text_book2(), TestDriverInterface::text_book3(), and TestDriverInterface::text_book_ouu().

void herbie1D (size_t *der_mode,* Real *xc_loc,* std::vector< Real > & *w_and_ders*) [private]

1D components of herbie function

1D Herbie function and its derivatives (apart from a multiplicative factor)

Referenced by TestDriverInterface::herbie().

void smooth_herbie1D (size_t *der_mode,* Real *xc_loc,* std::vector< Real > & *w_and_ders*) [private]

1D components of smooth_herbie function

1D Smoothed Herbie= 1DHerbie minus the high frequency sine term, and its derivatives (apart from a multiplicative factor)

Referenced by TestDriverInterface::smooth_herbie().

void shubert1D (size_t *der_mode,* Real *xc_loc,* std::vector< Real > & *w_and_ders*) [private]

1D components of shubert function

1D Shubert function and its derivatives (apart from a multiplicative factor)

Referenced by TestDriverInterface::shubert().

int herbie ( ) [private]

returns the N-D herbie function

N-D Herbie function and its derivatives.

References DirectApplicInterface::directFnASV, DirectApplicInterface::directFnDVV, TestDriverInterface::herbie1D(), DirectApplicInterface::numDerivVars, DirectApplicInterface::numVars, TestDriverInterface::separable_combine(), and DirectApplicInterface::xC.

Referenced by TestDriverInterface::derived_map_ac().

int smooth_herbie ( ) [private]

returns the N-D smooth herbie function

N-D Smoothed Herbie function and its derivatives.

References DirectApplicInterface::directFnASV, DirectApplicInterface::directFnDVV, DirectApplicInterface::numDerivVars, DirectApplicInterface::numVars, TestDriverInterface::separable_combine(), TestDriverInterface::smooth_herbie1D(), and DirectApplicInterface::xC.

Referenced by TestDriverInterface::derived_map_ac().

void separable_combine (Real *mult_scale_factor,* std::vector< Real > & *w,* std::vector< Real > & *d1w,* std::vector< Real > & *d2w*) [private]

utility to combine components of separable fns

this function combines N 1D functions and their derivatives to compute a N-D separable function and its derivatives, logic is general enough to support different 1D functions in different dimensions (can mix and match)

References DirectApplicInterface::directFnASV, DirectApplicInterface::directFnDVV, DirectApplicInterface::fnGrads, DirectApplicInterface::fnHessians, DirectApplicInterface::fnVals, DirectApplicInterface::numDerivVars, and DirectApplicInterface::numVars.

Referenced by TestDriverInterface::herbie(), TestDriverInterface::shubert(), and TestDriverInterface::smooth_herbie().

int mc_api_run ( ) [private]

direct interface to ModelCenter via API, HKIM 4/3/03

The ModelCenter interface doesn't have any specific construct vs. run time functions. For now, we manage it along with the integrated test drivers

References Dakota::abort_handler(), DirectApplicInterface::analysisComponents, DirectApplicInterface::analysisDriverIndex, Dakota::dc_ptr_int, DirectApplicInterface::directFnASV, Interface::fnLabels, DirectApplicInterface::fnVals, Dakota::mc_ptr_int, ApplicationInterface::multiProcAnalysisFlag, DirectApplicInterface::numACV, DirectApplicInterface::numADIV, DirectApplicInterface::numADRV, DirectApplicInterface::numFns, DirectApplicInterface::xC, DirectApplicInterface::xCLabels, DirectApplicInterface::xDI, DirectApplicInterface::xDILabels, DirectApplicInterface::xDR, and DirectApplicInterface::xDRLabels.

Referenced by TestDriverInterface::derived_map_ac().

The documentation for this class was generated from the following files:

- TestDriverInterface.hpp
- TestDriverInterface.cpp

13.169 TrackerHTTP Class Reference

TrackerHTTP: a usage tracking module that uses HTTP/HTTPS via the curl library.

Public Member Functions

- TrackerHTTP ()

  default constructor is allowed, but doesn't generate output
- TrackerHTTP (int world_rank=0)

  standard constructor with ProblemDescDB, rank
- ~TrackerHTTP ()

  destructor to free handles
- void post_start (ProblemDescDB &problem_db)

  post the start of an analysis and archive start time
- void post_finish (unsigned runtime=0)

  post the completion of an analysis including elapsed time

Private Member Functions

- void initialize (int world_rank=0)

  shared initialization functions across constructors
- void url_add_field (std::string &url, const char *keyword, const std::string &value, bool delimit=true) const

  append keyword/value pair to url in GET style (with &keyword=value); set delimit = false to omit the &
- void build_default_data (std::string &url, std::time_t &rawtime, const std::string &mode) const

  construct URL with shared information for start/finish
- void send_data_using_get (const std::string &urltopost) const

  transmit data to the web server using GET
- void send_data_using_post (const std::string &datatopost) const

  POST separate location and query; datatopost="name=daniel&project=curl".

- void populate_method_list (ProblemDescDB &problem_db)

  extract list of methods from problem database
- std::string get_uid () const

  get the real user ID
- std::string get_username () const

  get the username as reported by the environment
- std::string get_hostname () const

  get the system hostname
- std::string get_os () const

  get the operating system
- std::string get_datetime (const std::time_t &rawtime) const

  get the date and time as a string YYYYMMDDHHMMSS

Private Attributes

- CURL * curlPtr

  pointer to the curl handler instance
- FILE * devNull

  pointer to /dev/null
- std::string trackerLocation

  base URL for the tracker
- std::string proxyLocation

  if empty, proxy may still be specified via environment variables (unlike default CURL behavior)
- long timeoutSeconds

  seconds until the request will timeout (may have issues with signals)
- std::string methodList

  list of active methods
- std::string dakotaVersion

  DAKOTA version.
- std::time_t startTime

  cached starting time in raw seconds
- short outputLevel

  verbosity control

13.169.1 Detailed Description

TrackerHTTP: a usage tracking module that uses HTTP/HTTPS via the curl library.

13.169.2 Member Function Documentation

void send_data_using_get (const std::string & *urltopost*) const `[private]`

transmit data to the web server using GET

whole url including location&fields

References TrackerHTTP::curlPtr, and TrackerHTTP::outputLevel.

void send_data_using_post (const std::string & *datatopost*) const [private]

POST separate location and query; datatopost="name=daniel&project=curl".

separate location and query; datatopost="name=daniel&project=curl"

References TrackerHTTP::curlPtr, TrackerHTTP::outputLevel, and TrackerHTTP::trackerLocation.

Referenced by TrackerHTTP::post_finish(), and TrackerHTTP::post_start().

The documentation for this class was generated from the following files:

- TrackerHTTP.hpp
- TrackerHTTP.cpp

13.170 UsageTracker Class Reference

Lightweight class to manage conditionally active Curl-based HTTP tracker via PIMPL.

Public Member Functions

- UsageTracker ()

  default construction: no output
- UsageTracker (int world_rank)

  standard constructor; will output on rank 0
- void post_start (ProblemDescDB &problem_db)

  post the start of an analysis and archive start time
- void post_finish (unsigned runtime=0)

  post the completion of an analysis including elapsed time

Private Member Functions

- UsageTracker (const UsageTracker &)

  copy construction is disallowed

Private Attributes

- boost::shared_ptr< TrackerHTTP > pTrackerHTTP

  posts usage data to Web server; using shared_ptr due to potentially incomplete type and requirements for checked-_delete in debug builds (scoped_ptr would suffice)

13.170.1 Detailed Description

Lightweight class to manage conditionally active Curl-based HTTP tracker via PIMPL.

All conditional compilation in managed in the cpp file; all operations are no-op in this wrapper if not enabling tracking...

13.170.2 Constructor & Destructor Documentation

UsageTracker (int *world_rank*)

standard constructor; will output on rank 0

standard constructor; will output on rank 0 and only initializes if tracking compiled in and not disable by environment

References UsageTracker::pTrackerHTTP.

The documentation for this class was generated from the following files:

- UsageTracker.hpp
- UsageTracker.cpp

13.171 Var_icheck Struct Reference

structure for verifying bounds and initial point for string-valued vars

Public Attributes

- const char ∗ **name**
- size_t DataVariablesRep::∗ **n**
- void(∗ **vgen**)(DataVariablesRep ∗, size_t)
- IntVector DataVariablesRep::∗ **L**
- IntVector DataVariablesRep::∗ **U**
- IntVector DataVariablesRep::∗ **V**
- StringArray DataVariablesRep::∗ **Lbl**

13.171.1 Detailed Description

structure for verifying bounds and initial point for string-valued vars

structure for verifying bounds and initial point for integer-valued vars

The documentation for this struct was generated from the following file:

- NIDRProblemDescDB.cpp

13.172 Var_rcheck Struct Reference

structure for verifying bounds and initial point for real-valued vars

Public Attributes

- const char ∗ **name**
- size_t DataVariablesRep::∗ **n**
- void(∗ **vgen**)(DataVariablesRep ∗, size_t)
- RealVector DataVariablesRep::∗ **L**
- RealVector DataVariablesRep::∗ **U**
- RealVector DataVariablesRep::∗ **V**
- StringArray DataVariablesRep::∗ **Lbl**

13.172.1 Detailed Description

structure for verifying bounds and initial point for real-valued vars

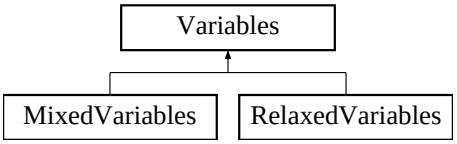
The documentation for this struct was generated from the following file:

- NIDRProblemDescDB.cpp

13.173 Variables Class Reference

Base class for the variables class hierarchy.

Inheritance diagram for Variables:



Public Member Functions

- Variables ()

  default constructor
- Variables (const ProblemDescDB &problem_db)

  standard constructor
- Variables (const SharedVariablesData &svd)

  alternate constructor for instantiations on the fly
- Variables (const Variables &vars)

  copy constructor
- virtual ~Variables ()

  destructor
- Variables operator= (const Variables &vars)

  assignment operator
- virtual void read (std::istream &s)

  read a variables object from an std::istream
- virtual void write (std::ostream &s) const

  write a variables object to an std::ostream
- virtual void write_aprepro (std::ostream &s) const

  write a variables object to an std::ostream in aprepro format
- virtual void read_annotated (std::istream &s)

  read a variables object in annotated format from an istream
- virtual void write_annotated (std::ostream &s) const

  write a variables object in annotated format to an std::ostream
- virtual void read_tabular (std::istream &s, bool active_only=false)

  read a variables object in tabular format from an istream
- virtual void write_tabular (std::ostream &s, bool active_only=false) const

write a variables object in tabular format to an std::ostream

- void write_tabular_labels (std::ostream &s, bool active_only=false) const

  write the labels in input spec order to a std::ostream
- virtual void read (MPIUnpackBuffer &s)

  read a variables object from a packed MPI buffer
- virtual void write (MPIPackBuffer &s) const

  write a variables object to a packed MPI buffer
- size_t tv () const

  total number of vars
- size_t cv () const

  number of active continuous vars
- size_t cv_start () const

  start index of active continuous vars
- size_t div () const

  number of active discrete int vars
- size_t div_start () const

  start index of active discrete int vars
- size_t dsv () const

  number of active discrete string vars
- size_t dsv_start () const

  start index of active discrete string vars
- size_t drv () const

  number of active discrete real vars
- size_t drv_start () const

  start index of active discrete real vars
- size_t icv () const

  number of inactive continuous vars
- size_t icv_start () const

  start index of inactive continuous vars
- size_t idiv () const

  number of inactive discrete int vars
- size_t idiv_start () const

  start index of inactive discrete int vars
- size_t idsv () const

  number of inactive discrete string vars
- size_t idsv_start () const

  start index of inactive discrete string vars
- size_t idrv () const

  number of inactive discrete real vars
- size_t idrv_start () const

  start index of inactive discrete real vars
- size_t acv () const

  total number of continuous vars

- size_t adiv () const

  total number of discrete integer vars
- size_t adsv () const

  total number of discrete string vars
- size_t adrv () const

  total number of discrete real vars
- const SharedVariablesData & shared_data () const

  return sharedVarsData
- SharedVariablesData & shared_data ()

  return sharedVarsData
- void shape ()

  shape a Variables object based on sharedVarsData
- void reshape ()

  reshape an existing Variables object based on updated sharedVarsData
- Real continuous_variable (size_t index) const

  return an active continuous variable
- const RealVector & continuous_variables () const

  return the active continuous variables (Note: returns a view by const reference, but initializing a RealVector from this reference invokes the Teuchos matrix copy constructor to create a Teuchos::Copy instance; to obtain a mutable view, use continuous_variables_view())
- void continuous_variable (Real c_var, size_t index)

  set an active continuous variable
- void continuous_variables (const RealVector &c_vars)

  set the active continuous variables
- int discrete_int_variable (size_t index) const

  return an active discrete integer variable
- const IntVector & discrete_int_variables () const

  return the active discrete integer variables (Note: returns a view by const reference, but initializing an IntVector from this reference invokes the Teuchos matrix copy constructor to create a Teuchos::Copy instance; to obtain a mutable view, use discrete_int_variables_view())
- void discrete_int_variable (int di_var, size_t index)

  set an active discrete integer variable
- void discrete_int_variables (const IntVector &di_vars)

  set the active discrete integer variables
- const String & discrete_string_variable (size_t index) const

  return an active discrete string variable
- StringMultiArrayConstView discrete_string_variables () const

  return the active discrete string variables (Note: returns a view by const reference, but initializing a StringArray from this reference invokes the Teuchos matrix copy constructor to create a Teuchos::Copy instance; to obtain a mutable view, use discrete_string_variables_view())
- void discrete_string_variable (const String &ds_var, size_t index)

  set an active discrete string variable
- void discrete_string_variables (StringMultiArrayConstView ds_vars)

  set the active discrete string variables

- Real discrete_real_variable (size_t index) const

  return an active discrete real variable
- const RealVector & discrete_real_variables () const

  return the active discrete real variables (Note: returns a view by const reference, but initializing a RealVector from this reference invokes the Teuchos matrix copy constructor to create a Teuchos::Copy instance; to obtain a mutable view, use discrete_real_variables_view())
- void discrete_real_variable (Real dr_var, size_t index)

  set an active discrete real variable
- void discrete_real_variables (const RealVector &dr_vars)

  set the active discrete real variables
- void active_variables (const Variables &vars)

  copy the active cv/div/dsv/drv variables from vars
- RealVector & continuous_variables_view ()

  return a mutable view of the active continuous variables
- IntVector & discrete_int_variables_view ()

  return a mutable view of the active discrete integer variables
- StringMultiArrayView discrete_string_variables_view ()

  return a mutable view of the active discrete string variables
- RealVector & discrete_real_variables_view ()

  return a mutable view of the active discrete real variables
- StringMultiArrayConstView continuous_variable_labels () const

  return the active continuous variable labels
- void continuous_variable_labels (StringMultiArrayConstView cv_labels)

  set the active continuous variable labels
- void continuous_variable_label (const String &cv_label, size_t index)

  set an active continuous variable label
- StringMultiArrayConstView discrete_int_variable_labels () const

  return the active discrete integer variable labels
- void discrete_int_variable_labels (StringMultiArrayConstView div_labels)

  set the active discrete integer variable labels
- void discrete_int_variable_label (const String &div_label, size_t index)

  set an active discrete integer variable label
- StringMultiArrayConstView discrete_string_variable_labels () const

  return the active discrete string variable labels
- void discrete_string_variable_labels (StringMultiArrayConstView dsv_labels)

  set the active discrete string variable labels
- void discrete_string_variable_label (const String &dsv_label, size_t index)

  set an active discrete string variable label
- StringMultiArrayConstView discrete_real_variable_labels () const

  return the active discrete real variable labels
- void discrete_real_variable_labels (StringMultiArrayConstView drv_labels)

  set the active discrete real variable labels
- void discrete_real_variable_label (const String &drv_label, size_t index)

set an active discrete real variable label

- UShortMultiArrayConstView continuous_variable_types () const

  return the active continuous variable types
- void continuous_variable_types (UShortMultiArrayConstView cv_types)

  set the active continuous variable types
- void continuous_variable_type (unsigned short cv_type, size_t index)

  set an active continuous variable type
- UShortMultiArrayConstView discrete_int_variable_types () const

  return the active discrete integer variable types
- void discrete_int_variable_types (UShortMultiArrayConstView div_types)

  set the active discrete integer variable types
- void discrete_int_variable_type (unsigned short div_type, size_t index)

  set an active discrete integer variable type
- UShortMultiArrayConstView discrete_string_variable_types () const

  return the active discrete string variable types
- void discrete_string_variable_types (UShortMultiArrayConstView dsv_types)

  set the active discrete string variable types
- void discrete_string_variable_type (unsigned short dsv_type, size_t index)

  set an active discrete string variable type
- UShortMultiArrayConstView discrete_real_variable_types () const

  return the active discrete real variable types
- void discrete_real_variable_types (UShortMultiArrayConstView drv_types)

  set the active discrete real variable types
- void discrete_real_variable_type (unsigned short drv_type, size_t index)

  set an active discrete real variable type
- SizetMultiArrayConstView continuous_variable_ids () const

  return the active continuous variable position identifiers
- void continuous_variable_ids (SizetMultiArrayConstView cv_ids)

  set the active continuous variable position identifiers
- void continuous_variable_id (size_t cv_id, size_t index)

  set an active continuous variable position identifier
- const RealVector & inactive_continuous_variables () const

  return the inactive continuous variables
- void inactive_continuous_variables (const RealVector &ic_vars)

  set the inactive continuous variables
- const IntVector & inactive_discrete_int_variables () const

  return the inactive discrete variables
- void inactive_discrete_int_variables (const IntVector &idi_vars)

  set the inactive discrete variables
- StringMultiArrayConstView inactive_discrete_string_variables () const

  return the inactive discrete variables
- void inactive_discrete_string_variables (StringMultiArrayConstView ids_vars)

  set the inactive discrete variables

- const RealVector & inactive_discrete_real_variables () const

  return the inactive discrete variables
- void inactive_discrete_real_variables (const RealVector &idr_vars)

  set the inactive discrete variables
- StringMultiArrayConstView inactive_continuous_variable_labels () const

  return the inactive continuous variable labels
- void inactive_continuous_variable_labels (StringMultiArrayConstView ic_vars)

  set the inactive continuous variable labels
- StringMultiArrayConstView inactive_discrete_int_variable_labels () const

  return the inactive discrete variable labels
- void inactive_discrete_int_variable_labels (StringMultiArrayConstView idi_vars)

  set the inactive discrete variable labels
- StringMultiArrayConstView inactive_discrete_string_variable_labels () const

  return the inactive discrete variable labels
- void inactive_discrete_string_variable_labels (StringMultiArrayConstView ids_vars)

  set the inactive discrete variable labels
- StringMultiArrayConstView inactive_discrete_real_variable_labels () const

  return the inactive discrete variable labels
- void inactive_discrete_real_variable_labels (StringMultiArrayConstView idr_vars)

  set the inactive discrete variable labels
- UShortMultiArrayConstView inactive_continuous_variable_types () const

  return the inactive continuous variable types
- UShortMultiArrayConstView inactive_discrete_int_variable_types () const

  return the inactive discrete integer variable types
- UShortMultiArrayConstView inactive_discrete_string_variable_types () const

  return the inactive discrete string variable types
- UShortMultiArrayConstView inactive_discrete_real_variable_types () const

  return the inactive discrete real variable types
- SizetMultiArrayConstView inactive_continuous_variable_ids () const

  return the inactive continuous variable position identifiers
- const RealVector & all_continuous_variables () const

  returns a single array with all continuous variables
- void all_continuous_variables (const RealVector &ac_vars)

  sets all continuous variables using a single array
- void all_continuous_variable (Real ac_var, size_t index)

  set a variable within the all continuous array
- const IntVector & all_discrete_int_variables () const

  returns a single array with all discrete variables
- void all_discrete_int_variables (const IntVector &adi_vars)

  sets all discrete variables using a single array
- void all_discrete_int_variable (int adi_var, size_t index)

  set a variable within the all discrete array
- StringMultiArrayConstView all_discrete_string_variables () const

returns a single array with all discrete variables

- void all_discrete_string_variables (StringMultiArrayConstView ads_vars)

  sets all discrete variables using a single array
- void all_discrete_string_variable (const String &ads_var, size_t index)

  set a variable within the all discrete array
- const RealVector & all_discrete_real_variables () const

  returns a single array with all discrete variables
- void all_discrete_real_variables (const RealVector &adr_vars)

  sets all discrete variables using a single array
- void all_discrete_real_variable (Real adr_var, size_t index)

  set a variable within the all discrete array
- StringMultiArrayView all_continuous_variable_labels () const

  returns a single array with all continuous variable labels
- void all_continuous_variable_labels (StringMultiArrayConstView acv_labels)

  sets all continuous variable labels using a single array
- void all_continuous_variable_label (const String &acv_label, size_t index)

  set a label within the all continuous label array
- StringMultiArrayView all_discrete_int_variable_labels () const

  returns a single array with all discrete variable labels
- void all_discrete_int_variable_labels (StringMultiArrayConstView adiv_labels)

  sets all discrete variable labels using a single array
- void all_discrete_int_variable_label (const String &adiv_label, size_t index)

  set a label within the all discrete label array
- StringMultiArrayView all_discrete_string_variable_labels () const

  returns a single array with all discrete variable labels
- void all_discrete_string_variable_labels (StringMultiArrayConstView adsv_labels)

  sets all discrete variable labels using a single array
- void all_discrete_string_variable_label (const String &adsv_label, size_t index)

  set a label within the all discrete label array
- StringMultiArrayView all_discrete_real_variable_labels () const

  returns a single array with all discrete variable labels
- void all_discrete_real_variable_labels (StringMultiArrayConstView adrv_labels)

  sets all discrete variable labels using a single array
- void all_discrete_real_variable_label (const String &adrv_label, size_t index)

  set a label within the all discrete label array
- UShortMultiArrayConstView all_continuous_variable_types () const

  return all continuous variable types
- UShortMultiArrayConstView all_discrete_int_variable_types () const

  return all discrete variable types
- UShortMultiArrayConstView all_discrete_string_variable_types () const

  return all discrete variable types
- UShortMultiArrayConstView all_discrete_real_variable_types () const

  return all discrete variable types

- SizetMultiArrayConstView all_continuous_variable_ids () const

  return all continuous variable position identifiers
- Variables copy (bool deep_svd=false) const

  for use when a deep copy is needed (the representation is not shared)
- const std::pair< short, short > & view () const

  returns variablesView
- std::pair< short, short > get_view (const ProblemDescDB &problem_db) const

  defines variablesView from problem_db attributes
- void inactive_view (short view2)

  sets the inactive view based on higher level (nested) context
- const String & variables_id () const

  returns the variables identifier string
- const SizetArray & variables_components_totals () const

  returns the number of variables for each of the constitutive components
- bool is_null () const

  function to check variablesRep (does this envelope contain a letter)

Protected Member Functions

- Variables (BaseConstructor, const ProblemDescDB &problem_db, const std::pair< short, short > &view)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
- Variables (BaseConstructor, const SharedVariablesData &svd)

  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
- void build_views ()

  construct active/inactive views of all variables arrays
- void build_active_views ()

  construct active views of all variables arrays
- void build_inactive_views ()

  construct inactive views of all variables arrays

Protected Attributes

- SharedVariablesData sharedVarsData

  reference-counted instance of shared variables data: id's, labels, counts
- RealVector allContinuousVars

  array combining all of the continuous variables
- IntVector allDiscreteIntVars

  array combining all of the discrete integer variables
- StringMultiArray allDiscreteStringVars

  array combining all of the discrete string variables
- RealVector allDiscreteRealVars

  array combining all of the discrete real variables

- RealVector continuousVars

  the active continuous variables array view
- IntVector discreteIntVars

  the active discrete integer variables array view
- RealVector discreteRealVars

  the active discrete real variables array view
- RealVector inactiveContinuousVars

  the inactive continuous variables array view
- IntVector inactiveDiscreteIntVars

  the inactive discrete integer variables array view
- RealVector inactiveDiscreteRealVars

  the inactive discrete real variables array view

Private Member Functions

- Variables ∗ get_variables (const ProblemDescDB &problem_db)

  Used by the standard envelope constructor to instantiate the correct letter class.
- Variables ∗ get_variables (const SharedVariablesData &svd) const

  Used by the alternate envelope constructors, by read functions, and by copy() to instantiate a new letter class.
- short method_map (short view_spec, bool relaxed) const

  infer domain from method selection
- short method_domain (const ProblemDescDB &problem_db) const

  infer domain from method selection
- short method_view (const ProblemDescDB &problem_db) const

  infer view from method selection
- short response_view (const ProblemDescDB &problem_db) const

  infer view from type of response data set
- void check_view_compatibility ()

  perform sanity checks on view.first and view.second after update
- template<class Archive >
  void load (Archive &ar, const unsigned int version)

  read a Variables object from an archive
- template<class Archive >
  void save (Archive &ar, const unsigned int version) const

  write a Variables object to an archive
- BOOST_SERIALIZATION_SPLIT_MEMBER () Variables ∗variablesRep

  pointer to the letter (initialized only for the envelope)

Private Attributes

- int referenceCount

  number of objects sharing variablesRep

Friends

- class boost::serialization::access

  for serializing private data members
- bool operator== (const Variables &vars1, const Variables &vars2)

  strict equality operator (for boost hash-based lookups)
- bool operator!= (const Variables &vars1, const Variables &vars2)

  strict inequality operator
- bool nearby (const Variables &vars1, const Variables &vars2, Real rel_tol)

  tolerance-based equality operator
- std::size_t hash_value (const Variables &vars)

  hash_value

13.173.1 Detailed Description

Base class for the variables class hierarchy.

The Variables class is the base class for the class hierarchy providing design, uncertain, and state variables for continuous and discrete domains within a Model. Using the fundamental arrays from the input specification, different derived classes define different views of the data. For memory efficiency and enhanced polymorphism, the variables hierarchy employs the "letter/envelope idiom" (see Coplien "Advanced C++", p. 133), for which the base class (Variables) serves as the envelope and one of the derived classes (selected in Variables::get_variables()) serves as the letter.

13.173.2 Member Function Documentation

StringMultiArrayView discrete_string_variables_view ( ) `[inline]`

return a mutable view of the active discrete string variables

same as discrete_string_variables(), except mutable view

References Variables::allDiscreteStringVars, SharedVariablesData::dsv(), SharedVariablesData::dsv_start(), and Variables::sharedVarsData.

void build_views ( ) `[inline], [protected]`

construct active/inactive views of all variables arrays

= EMPTY)

= EMPTY)

References Variables::build_active_views(), Variables::build_inactive_views(), Variables::sharedVarsData, Variables::view(), and SharedVariablesData::view().

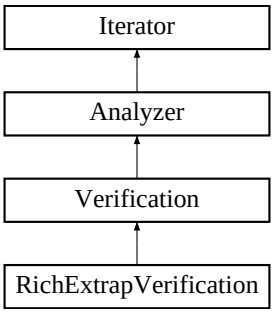
The documentation for this class was generated from the following file:

- DakotaVariables.hpp

13.174 Verification Class Reference

Base class for managing common aspects of verification studies.

Inheritance diagram for Verification:



Protected Member Functions

- Verification (ProblemDescDB &problem_db, Model &model)

  constructor
- Verification (unsigned short method_name, Model &model)

  alternate constructor for instantiations "on the fly"
- ~Verification ()

  destructor
- void core_run ()

  core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post
- void print_results (std::ostream &s)

  print the final iterator results
- virtual void perform_verification ()=0

  Redefines the core_run() virtual function for the Verification branch.

Additional Inherited Members

13.174.1 Detailed Description

Base class for managing common aspects of verification studies.

The Verification base class manages common data and functions, such as those involving ...

13.174.2 Member Function Documentation

void core_run ( ) [inline], [protected], [virtual]

core portion of run; implemented by all derived classes and may include pre/post steps in lieu of separate pre/post

Virtual run function for the iterator class hierarchy. All derived classes need to redefine it.

Reimplemented from Iterator.

References Verification::perform_verification().

void print_results (std::ostream & *s*) [protected], [virtual]

print the final iterator results

This virtual function provides additional iterator-specific final results outputs beyond the function evaluation summary printed in finalize_run().

Reimplemented from Analyzer.

Reimplemented in RichExtrapVerification.
References Analyzer::print_results().
Referenced by RichExtrapVerification::print_results().
The documentation for this class was generated from the following files:

- DakotaVerification.hpp
- DakotaVerification.cpp

13.175 VLint Struct Reference

structure for validating integer uncertain variable labels, bounds, values

Public Attributes

- int **n**
- VarLabel Var_Info::∗ **VL**
- Var_uinfo ∗ **vui**
- StringArray DataVariablesRep::∗ **Labels**
- IntVector DataVariablesRep::∗ **LowerBnds**
- IntVector DataVariablesRep::∗ **UpperBnds**
- IntVector DataVariablesRep::∗ **UncVars**

13.175.1 Detailed Description

structure for validating integer uncertain variable labels, bounds, values
The documentation for this struct was generated from the following file:

- NIDRProblemDescDB.cpp

13.176 VLreal Struct Reference

structure for validating real uncertain variable labels, bounds, values

Public Attributes

- int **n**
- VarLabel Var_Info::∗ **VL**
- Var_uinfo ∗ **vui**
- StringArray DataVariablesRep::∗ **Labels**
- RealVector DataVariablesRep::∗ **LowerBnds**
- RealVector DataVariablesRep::∗ **UpperBnds**
- RealVector DataVariablesRep::∗ **UncVars**

13.176.1 Detailed Description

structure for validating real uncertain variable labels, bounds, values
The documentation for this struct was generated from the following file:

- NIDRProblemDescDB.cpp

13.177 VLstr Struct Reference

structure for validating string uncertain variable labels, bounds, values

Public Attributes

- int **n**
- VarLabel Var_Info::∗ **VL**
- Var_uinfo ∗ **vui**
- StringArray DataVariablesRep::∗ **Labels**
- StringArray DataVariablesRep::∗ **LowerBnds**
- StringArray DataVariablesRep::∗ **UpperBnds**
- StringArray DataVariablesRep::∗ **UncVars**

13.177.1 Detailed Description

structure for validating string uncertain variable labels, bounds, values

The documentation for this struct was generated from the following file:

- NIDRProblemDescDB.cpp

13.178 WorkdirHelper Class Reference

Static Public Member Functions

- static void initialize ()

  initialize (at runtime) cached values for paths and environment
- static const std::string & startup_pwd ()

  Query for dakota's startup $PWD.
- static void change_directory (const bfs::path &new_dir)

  change current directory
- static void prepend_preferred_env_path (const std::string &extra_path)

  Prepend cached preferredEnvPath with extra_path and update $PATH environment variable.
- static void set_environment (const std::string &env_name, const std::string &env_val, bool overwrite_flag=true)

  Set an environment variable.
- static bfs::path which (const std::string &driver_name)

  Returns the bfs::path for the analysis driver, supporting typical windows filename extensions, or empty if not found.
- static bfs::path rel_to_abs (const bfs::path &subdir_path)

  get a valid absolute bfs::path to a subdirectory relative to rundir
- static StringArray tokenize_driver (const String &user_an_driver)

  tokenize a white-space separated analysis driver, respecting escapes and nested quotes
- static bool resolve_driver_path (String &an_driver)

  parse off the first whitespace-separated entry in the user's analysis_driver, and convert it to an absolute path if it begins with ./ or ../, replacing the passed string if needed. Returns true if the first token was modified.
- static void split_wildcard (const std::string &path_with_wc, bfs::path &search_dir, bfs::path &wild_card)

  given a string with an optional path and a wildcard, e.g., /tmp/D∗.?pp, parse it into the search path /tmp (default .) and the wildcard D∗.?pp. Return wild_card as path to reduce wstring conversions

- static bfs::path concat_path (const bfs::path &p_in, const String &tag)

  concatenate a string onto the end of a path
- static bfs::path system_tmp_file (const std::string &prefix)

  generate a valid temporary file name <prefix>_%%%%%%%%
- static bfs::path system_tmp_path ()

  get the system tmp path, e.g., /tmp or C:
- static bool create_directory (const bfs::path &dir_path, short mkdir_option)

  Create a directory, with options for remove or error.
- static void recursive_remove (const bfs::path &rm_path, short fileop_option)

  Remove a path (file, directory, or symlink) without regard to its type. Only error if existed and there's an error in the remove.
- static void rename (const bfs::path &old_path, const bfs::path &new_path, short fileop_option)

  Rename a file, catching any errors and optionally warning/erroring.
- static void link_items (const StringArray &source_itemss, const bfs::path &dest_dir, bool overwrite)

  top-level link a list of source_paths (files, directories, symlinks), potentially including wildcards, from destination_dir, which must exist
- static void copy_items (const StringArray &source_items, const bfs::path &dest_dir, bool overwrite)

  copy a list of source_paths (files, directories, symlinks), potentially including wildcards into destination_dir, which must exist
- static void prepend_path_items (const StringArray &source_items)

  prepend any directories (including wildcards) found in source_items to the preferred environment path; this will update cached preferred path and PATH
- static bool check_equivalent_dest (const StringArray &source_items, const bfs::path &dest_dir)

  check whether any of the passed source items are filesystem equivalent to the destination path, return true if any one is equivalent to dest
- static bool find_driver (const StringArray &source_items, const bfs::path &search_driver)

  check whether the any of the passed source items (possibly including wildcards to be expanded) matches the passed search driver
- static bool link (const bfs::path &src_path, const bfs::path &dest_dir, bool overwrite)

  create link from dest_dir/src_path.filename() to a single path (file, dir, link) in source directory
- static bool recursive_copy (const bfs::path &src_path, const bfs::path &dest_dir, bool overwrite)

  Recrusive copy of src_path into dest_dir, with optional top-level overwrite (remove/recreate) of dest_dir/src_path.filename()
- static bool prepend_path_item (const bfs::path &src_path, const bfs::path &dest_dir, bool overwrite)

  prepend the preferred env path with source path if it's a directory; this will update cached preferred path and manipulate PATH
- static bool check_equivalent (const bfs::path &src_path, const bfs::path &dest_dir, bool overwrite)

  return true if the src and dest are filesystem equivalent
- static bool find_file (const bfs::path &src_path, const bfs::path &search_file, bool overwrite)

  return true if the src_path is a regular file and has same filename as search_file
- static bool file_op_items (const file_op_function &file_op, const StringArray &source_paths, const bfs::path &dest_dir, bool overwrite)

  recursively perform file_op (copy, path adjust, etc.) on a list of source_paths (files, directories, symlinks), which potentially include wildcards, w.r.t. destination_dir
- static void set_preferred_path ()

set/reset PATH to dakPreferredEnvPath

- static void set_preferred_path (const boost::filesystem::path &extra_path)

  set PATH to absolute(extra_path):dakPreferredEnvPath, without changing cached preferred PATH

- static void reset ()

  Resets the working directory "state" to its initial state when DAKOTA was launched.

Private Member Functions

- WorkdirHelper ()

  default constructor

- WorkdirHelper (const WorkdirHelper &)

  copy constructor

- ~WorkdirHelper ()

  destructor

- WorkdirHelper & operator= (const WorkdirHelper &)

  assignment operator

Static Private Member Functions

- static bfs::path po_which (const std::string &driver_name)

  Returns the bfs::path for the analysis driver - POSIX-style implementation, returns empty if not found.

- static std::string init_startup_path ()

  Initializes class member, startupPATH.

- static std::string init_preferred_env_path ()

  Initializes class member, dakPreferredEnvPath.

- static std::vector< std::string > tokenize_env_path (const std::string &path)

  Tokenizes $PATH environment variable into a "list" of directories.

Static Private Attributes

- static std::string startupPWD = "."

  Value of $PWD var upon entry to dakota main()

- static std::string startupPATH = "."

  Value of $PATH (PATH% on windows) var upon entry to dakota main(), omitting any leading PATH= or Path=.

- static std::string dakPreferredEnvPath = "."

  Dakota preferred search PATH/Path = ".:startupPWD:startupPATH", omitting any leading PATH= or Path=.

13.178.1 Detailed Description

Utility class for cross-platform management of environment and paths. Including directory and file operations. On initialization, this class does not manipulate the present working directory, nor the PATH environment variable, but stores context to manipulate them later.

13.178.2 Member Function Documentation

void initialize ( ) [static]

initialize (at runtime) cached values for paths and environment

Initialize defers calls to Boost filesystem utilities until runtime (required on some operating systems.

References WorkdirHelper::dakPreferredEnvPath, WorkdirHelper::init_preferred_env_path(), WorkdirHelper::init_startup_path(), WorkdirHelper::startupPATH, and WorkdirHelper::startupPWD.

Referenced by Environment::Environment().

void prepend_preferred_env_path (const std::string & *extra_path*) [static]

Prepend cached preferredEnvPath with extra_path and update $PATH environment variable.

Overwrites $PATH with an additional directory prepended, typically for the purpose of ensuring templatedir is in the $PATH; updates cached preferred PATH and environment PATH, so exercise caution with repeated calls.

References WorkdirHelper::dakPreferredEnvPath, WorkdirHelper::set_environment(), and WorkdirHelper::startupPWD.

Referenced by WorkdirHelper::prepend_path_item().

bfs::path which (const std::string & *driver_name*) [static]

Returns the bfs::path for the analysis driver, supporting typical windows filename extensions, or empty if not found.

Uses string representing $PATH to locate an analysis driver on the host computer. Returns the path to the driver (as a string)

This version is a wrapper over the "plain ol' which" implementation, allowing an array of windows, 3-letter extensions to be checked.

References Dakota::get_pathext(), and WorkdirHelper::po_which().

Referenced by NIDRProblemDescDB::check_driver().

void split_wildcard (const std::string & *path_with_wc,* bfs::path & *search_dir,* bfs::path & *wild_card*) [static]

given a string with an optional path and a wildcard, e.g., /tmp/D∗.?pp, parse it into the search path /tmp (default .) and the wildcard D∗.?pp. Return wild_card as path to reduce wstring conversions

Input: path_with_wc; Output: search_dir, wild_card

Referenced by WorkdirHelper::file_op_items().

bool create_directory (const bfs::path & *dir_path,* short *mkdir_option*) [static]

Create a directory, with options for remove or error.

mkdir_option is DIR_CLEAN (remove and recreate), DIR_PERSIST (leave existing), or DIR_ERROR (don't allow existing) returns whether a new directory was created.

References Dakota::abort_handler(), and WorkdirHelper::recursive_remove().

Referenced by ProcessApplicInterface::define_filenames().

void link_items (const StringArray & *source_items,* const bfs::path & *dest_dir,* bool *overwrite*) [static]

top-level link a list of source_paths (files, directories, symlinks), potentially including wildcards, from destination_dir, which must exist

Iterate source items (paths or wildcards), linking each of them from the destination. If overwrite, remove and replace any existing destination target, otherwise, allow to persist

References WorkdirHelper::file_op_items(), and WorkdirHelper::link().

Referenced by ProcessApplicInterface::define_filenames().

void copy_items (const StringArray & *source_items,* const bfs::path & *dest_dir,* bool *overwrite*) `[static]`

copy a list of source_paths (files, directories, symlinks), potentially including wildcards into destination_dir, which must exist

Iterate source items (paths or wildcards), copying each of them into the destination. If overwrite, remove and replace any existing destination target, otherwise, allow to persist

References WorkdirHelper::file_op_items(), and WorkdirHelper::recursive_copy().

Referenced by ProcessApplicInterface::define_filenames().

bool link (const bfs::path & *src_path,* const bfs::path & *dest_dir,* bool *overwrite*) `[static]`

create link from dest_dir/src_path.filename() to a single path (file, dir, link) in source directory

Assumes source file exists since it was iterated in the calling context. If overwrite, any existing file in dest_dir will be removed prior to creating the new link.

References Dakota::abort_handler().

Referenced by WorkdirHelper::link_items().

bool recursive_copy (const bfs::path & *src_path,* const bfs::path & *dest_dir,* bool *overwrite*) `[static]`

Recrusive copy of src_path into dest_dir, with optional top-level overwrite (remove/recreate) of dest_dir/src_path.-filename()

note dest_dir is the containing folder for the src_path contents to be placed in for consistency with other convenience functions (may need to reconsider)

References Dakota::abort_handler().

Referenced by WorkdirHelper::copy_items().

bool prepend_path_item (const bfs::path & *src_path,* const bfs::path & *dest_dir,* bool *overwrite*) `[static]`

prepend the preferred env path with source path if it's a directory; this will update cached preferred path and manipulate PATH

prepend the env path with source path if it's a directory or directory symlink

References Dakota::abort_handler(), and WorkdirHelper::prepend_preferred_env_path().

Referenced by WorkdirHelper::prepend_path_items().

bool file_op_items (const file_op_function & *file_op,* const StringArray & *source_items,* const bfs::path & *dest_dir,* bool *overwrite*) `[static]`

recursively perform file_op (copy, path adjust, etc.) on a list of source_paths (files, directories, symlinks), which potentially include wildcards, w.r.t. destination_dir

Iterator implementation for copy, link, etc file operation. Iterate source items (paths or wildcards), performing file_op on each w.r.t. destination. If overwrite, remove and replace any existing destination target (at top-level), otherwise, allow to persist. Return code true indicates abnormal behavior.

References WorkdirHelper::split_wildcard(), and Dakota::strcontains().

Referenced by WorkdirHelper::check_equivalent_dest(), WorkdirHelper::copy_items(), WorkdirHelper::find_-driver(), WorkdirHelper::link_items(), and WorkdirHelper::prepend_path_items().

void set_preferred_path (const boost::filesystem::path & *extra_path*) [static]

set PATH to absolute(extra_path):dakPreferredEnvPath, without changing cached preferred PATH

If needed, convert the passed item to an absolute path (while could make sense to prepend a relative path, no current use cases) and prepend when setting environment. Does not update cached preferred path.

References WorkdirHelper::dakPreferredEnvPath, WorkdirHelper::rel_to_abs(), and WorkdirHelper::set_environment().

bfs::path po_which (const std::string & *driver_name*) [static], [private]

Returns the bfs::path for the analysis driver - POSIX-style implementation, returns empty if not found.

For absolute driver_name, validates that is regular file. For relative, uses string representing $PATH (preferred path) to locate an analysis driver on the host computer. Returns the path to the driver, or empty if not found.

This is the "plain ol' which" impl that worked well, historically, on POSIX.

References Dakota::contains(), WorkdirHelper::dakPreferredEnvPath, and WorkdirHelper::tokenize_env_path().

Referenced by WorkdirHelper::which().

std::string init_startup_path ( ) [static], [private]

Initializes class member, startupPATH.

Gets the $PATH (PATH% on windows) and returns the std::string value

References Dakota::abort_handler().

Referenced by WorkdirHelper::initialize().

std::string init_preferred_env_path ( ) [static], [private]

Initializes class member, dakPreferredEnvPath.

Prepends '.' and the startupPWD to the initial startup $PATH string so that analysis driver detection is more robust

References WorkdirHelper::startupPATH, and WorkdirHelper::startupPWD.

Referenced by WorkdirHelper::initialize().

std::vector< std::string > tokenize_env_path (const std::string & *env_path*) [static], [private]

Tokenizes $PATH environment variable into a "list" of directories.

Creates a a vector of directories (as an aid to search) by breaking up the $PATH environment variable (passed in as a string argument)

Referenced by WorkdirHelper::po_which().

The documentation for this class was generated from the following files:

- WorkdirHelper.hpp
- WorkdirHelper.cpp

Chapter 14

File Documentation

14.1 dakota_dll_api.cpp File Reference

This file contains a DakotaRunner class, which launches DAKOTA.

Namespaces

- Dakota

  The primary namespace for DAKOTA.

Constant Groups

- Dakota

  The primary namespace for DAKOTA.

Functions

- void DAKOTA_DLL_FN dakota_create (int ∗dakota_ptr_int, const char ∗logname)

  create and configure a new DakotaRunner, adding it to list of instances
- int DAKOTA_DLL_FN dakota_readInput (int id, const char ∗dakotaInput)

  command DakotaRunner instance id to read from file dakotaInput
- void DAKOTA_DLL_FN dakota_get_variable_info (int id, char ∗∗∗pVarNames, int ∗pNumVarNames, char ∗∗∗pRespNames, int ∗pNumRespNames)

  return the variable and response names
- int DAKOTA_DLL_FN dakota_start (int id)

  command DakotaRunner instance id to start (plugin interface and run strategy)
- void DAKOTA_DLL_FN dakota_destroy (int id)

  delete Dakota runner instance id and remove from active list
- void DAKOTA_DLL_FN dakota_stop (int ∗id)

  command DakotaRunner instance id to stop execution
- const char ∗DAKOTA_DLL_FN dakota_getStatus (int id)

  return current results output as a string
- int get_mc_ptr_int ()

get the DAKOTA pointer to ModelCenter

- void set_mc_ptr_int (int ptr_int)

  set the DAKOTA pointer to ModelCenter
- int get_dc_ptr_int ()

  get the DAKOTA pointer to ModelCenter current design point
- void set_dc_ptr_int (int ptr_int)

  set the DAKOTA pointer to ModelCenter current design point

14.1.1 Detailed Description

This file contains a DakotaRunner class, which launches DAKOTA.

14.1.2 Function Documentation

void DAKOTA_DLL_FN dakota_stop (int ∗ *id*)

command DakotaRunner instance id to stop execution

TODO: trick application to quit through the syscall interface or throw exception.

14.2 dakota_dll_api.h File Reference

API for DLL interactions.

Functions

- void DAKOTA_DLL_FN dakota_create (int ∗dakota_ptr_int, const char ∗logname)

  create and configure a new DakotaRunner, adding it to list of instances
- int DAKOTA_DLL_FN dakota_readInput (int id, const char ∗dakotaInput)

  command DakotaRunner instance id to read from file dakotaInput
- int DAKOTA_DLL_FN dakota_start (int id)

  command DakotaRunner instance id to start (plugin interface and run strategy)
- void DAKOTA_DLL_FN dakota_destroy (int id)

  delete Dakota runner instance id and remove from active list
- void DAKOTA_DLL_FN dakota_stop (int ∗id)

  command DakotaRunner instance id to stop execution
- const char ∗DAKOTA_DLL_FN dakota_getStatus (int id)

  return current results output as a string
- int DAKOTA_DLL_FN get_mc_ptr_int ()

  get the DAKOTA pointer to ModelCenter
- void DAKOTA_DLL_FN set_mc_ptr_int (int ptr_int)

  set the DAKOTA pointer to ModelCenter
- int DAKOTA_DLL_FN get_dc_ptr_int ()

  get the DAKOTA pointer to ModelCenter current design point
- void DAKOTA_DLL_FN set_dc_ptr_int (int ptr_int)

  set the DAKOTA pointer to ModelCenter current design point

- void DAKOTA_DLL_FN dakota_get_variable_info (int id, char ∗∗∗pVarNames, int ∗pNumVarNames, char ∗∗∗pRespNames, int ∗pNumRespNames)

  return the variable and response names

14.2.1 Detailed Description

API for DLL interactions.

14.2.2 Function Documentation

void DAKOTA_DLL_FN dakota_stop (int ∗ *id*)

command DakotaRunner instance id to stop execution

TODO: trick application to quit through the syscall interface or throw exception.

14.3 dakota_tabular_io.hpp File Reference

Utility functions for reading and writing tabular data files Emerging utilities for tabular file I/O. For now, just extraction of capability from separate contexts to faciliate rework. These augment (and leverage) those in data_-util.h.

Namespaces

- Dakota

  The primary namespace for DAKOTA.

Constant Groups

- Dakota

  The primary namespace for DAKOTA.

Functions

- void **open_file** (std::ifstream &data_file, const std::string &input_filename, const std::string &context_-message)

  open the file specified by name for reading, using passed input stream, presenting context-specific error on failure
- void **open_file** (std::ofstream &data_file, const std::string &output_filename, const std::string &context_-message)

  open the file specified by name for writing, using passed output stream, presenting context-specific error on failure
- void **write_header_tabular** (std::ostream &tabular_ostream, const Variables &vars, const Response &re-sponse, const std::string &counter_label)

  Output the header row (labels) for a tabular data file, with variables in input spec order. Conditionally include interface ID. Primary uses: environment tabular data, pre-run output, surrogate approx evals.
- void **write_leading_columns** (std::ostream &tabular_ostream, size_t eval_id, const String &iface_id)

  Write the leading column with eval ID and conditionally, the interface ID.
- void **write_data_tabular** (std::ostream &tabular_ostream, const Variables &vars, const String &iface, const Response &response, size_t counter, bool annotated)

Output a row of tabular data from variables and response object used by graphics to append to tabular file during iteration. All active/inactive variables written in input spec order. Conditionally include interface ID. Primary uses: environment tabular data, pre-run output, surrogate approx evals.

- void **write_data_tabular** (const std::string &output_filename, const std::string &context_message, const RealVectorArray &output_coeffs, const UShort2DArray &output_indices)

  PCE export: write freeform format file with whitespace-separated data where each row has num_fns reals from coeffs, followed by num_vars unsigned shorts from indices.
- bool **exists_extra_data** (std::istream &tabular_file)

  Check if an input stream contains unexpected additional data.
- void **read_header_tabular** (std::istream &input_stream, bool annotated)

  read and discard header line from the stream
- size_t **read_leading_columns** (std::istream &input_stream, bool annotated)

  read leading columns [int eval_id [String iface_id]]
- void **read_data_tabular** (const std::string &input_filename, const std::string &context_message, RealVector &input_data, size_t num_entries, bool annotated)

  read possibly header-annotated whitespace-separated data into a vector of length num_entries; if annotated then it's a column vector for now
- void **read_data_tabular** (const std::string &input_filename, const std::string &context_message, Variables vars, size_t num_fns, RealArray &input_vector, bool annotated, bool active_only)

  read possibly header-annotated whitespace-separated data of Variables, followed by num_fns, into a dynamic vector with minimal error checking
- void **read_data_tabular** (const std::string &input_filename, const std::string &context_message, RealVectorArray &input_coeffs, UShort2DArray &input_indices, bool annotated, size_t num_vars, size_t num_fns)

  PCE import: read possibly header-annotated whitespace-separated data of unknown length where each row has num_fns reals followed by num_vars unsigned shorts; append data to arrays passed by reference.
- void **read_data_tabular** (const std::string &input_filename, const std::string &context_message, Variables vars, Response resp, VariablesList &input_vars, ResponseList &input_resp, bool annotated, bool verbose=false, bool active_only=false)

  read whitespace-separated data with optional row and column headers into lists of Variables and Responses until out of data; continuous variables only
- void **read_data_tabular** (const std::string &input_filename, const std::string &context_message, RealMatrix &input_matrix, size_t num_rows, size_t num_cols, bool annotated, bool verbose=false)

  read whitespace-separated data with optional row and column headers into a single matrix
- size_t **read_data_tabular** (const std::string &input_filename, const std::string &context_message, RealVectorArray &cva, IntVectorArray &diva, StringMulti2DArray &dsva, RealVectorArray &drva, bool annotated, bool active_only, Variables vars)

  read specified input data file into arrays with sizes specified by the passed vc_totals array; used in ParamStudy

14.3.1 Detailed Description

Utility functions for reading and writing tabular data files Emerging utilities for tabular file I/O. For now, just extraction of capability from separate contexts to faciliate rework. These augment (and leverage) those in data_util.h. Design/capability goals: Ability to read / write data with row/col headers or in free-form Detect premature end of file, report if extra data More consistent and reliable checks for file open errors Require right number of cols in header mode; only total data checking in free-form (likely) Allow comment character for header rows or even in data? variables vs. variables/responses for both read and write Should we support CSV? delimiter = ',' ; other? Verify treatment of trailing newline without reading a zero Allow reading into the transpose of the data structure

14.4 dll_tester.cpp File Reference

Test the DLL with a DAKOTA input file.

Functions

- int main (int argc, char *argv[ ])

  The main program for exercising the DLL API with a simple command-line.

14.4.1 Detailed Description

Test the DLL with a DAKOTA input file.

14.5 JEGAOptimizer.cpp File Reference

Contains the implementation of the JEGAOptimizer class.

Classes

- class JEGAOptimizer::Evaluator

  An evaluator specialization that knows how to interact with Dakota.
- class JEGAOptimizer::EvaluatorCreator

  A specialization of the JEGA::FrontEnd::EvaluatorCreator that creates a new instance of a Evaluator.
- class JEGAOptimizer::Driver

  A subclass of the JEGA front end driver that exposes the individual protected methods to execute the algorithm.

Namespaces

- Dakota

  The primary namespace for DAKOTA.

Constant Groups

- Dakota

  The primary namespace for DAKOTA.

Functions

- template<typename T >
  string asstring (const T &val)

  Creates a string from the argument val using an ostringstream.

14.5.1 Detailed Description

Contains the implementation of the JEGAOptimizer class.

14.6 JEGAOptimizer.hpp File Reference

Contains the definition of the JEGAOptimizer class.

Classes

- class JEGAOptimizer

  A version of Dakota::Optimizer for instantiation of John Eddy's Genetic Algorithms (JEGA).

Namespaces

- Dakota

  The primary namespace for DAKOTA.

Constant Groups

- Dakota

  The primary namespace for DAKOTA.

14.6.1 Detailed Description

Contains the definition of the JEGAOptimizer class.

14.7 library_mode.cpp File Reference

file containing a mock simulator main for testing Dakota in library mode

Classes

- struct callback_data

Functions

- void fpinit_ASL ()
- void run_dakota_parse (const char *dakota_input_file)

  Run a Dakota LibraryEnvironment, mode 1: parsing an input file.
- void run_dakota_data ()

  Run a Dakota LibraryEnvironment, mode 2: from C++ API inserted data.
- void run_dakota_mixed (const char *dakota_input_file, bool mpirun_flag)

  Run a Dakota LibraryEnvironment, from string or input file input, supplemented with additional C++ API adjustments.
- void serial_interface_plugin (Dakota::LibraryEnvironment &env)

  Convenience function with simplest example of interface plugin: plugin a serial DirectApplicInterface that can be constructed independent of Dakota's configuration details.
- void parallel_interface_plugin (Dakota::LibraryEnvironment &env)

  Convenience function to plug a library client's interface into the appropriate model, demonstrating use of Dakota parallel configuration in constructing the plugin Interface on the right MPI_Comm.

- static void callback_function (Dakota::ProblemDescDB *db, void *ptr)

  Example: user-provided post-parse callback (Dakota::DbCallbackFunction)

- int main (int argc, char *argv[ ])

  A mock simulator main for testing Dakota in library mode.

Variables

- static const char serial_input [ ]

  Default Dakota input string for serial case (rosenbrock):

- static const char parallel_input [ ]

  Default Dakota input string for parallel case (text_book)

14.7.1 Detailed Description

file containing a mock simulator main for testing Dakota in library mode

14.7.2 Function Documentation

void fpinit_ASL ( )

Floating-point initialization from AMPL: switch to 53-bit rounding if appropriate, to eliminate some cross-platform differences.

Referenced by main().

void run_dakota_parse (const char * *dakota_input_file*)

Run a Dakota LibraryEnvironment, mode 1: parsing an input file.

Simplest library case: this function parses from an input file to define the ProblemDescDB data.

References Environment::execute(), ProgramOptions::input_file(), Environment::mpi_manager(), MPIManager::mpirun_flag(), parallel_interface_plugin(), serial_interface_plugin(), and MPIManager::world_rank().

Referenced by main().

void run_dakota_data ( )

Run a Dakota LibraryEnvironment, mode 2: from C++ API inserted data.

Rather than parsing from an input file, this function populates Data class objects directly using a minimal specification and relies on constructor defaults and post-processing in post_process() to fill in the rest.

References DataInterface::data_rep(), DataResponses::data_rep(), DataVariables::data_rep(), DataMethod::data_rep(), LibraryEnvironment::done_modifying_db(), Environment::execute(), Environment::exit_mode(), DataResponsesRep::gradientType, DataResponsesRep::hessianType, LibraryEnvironment::insert_nodes(), DataMethodRep::methodName, Environment::mpi_manager(), MPIManager::mpirun_flag(), ParallelLibrary::mpirun_flag(), DataVariablesRep::numContinuousDesVars, DataResponsesRep::numNonlinearIneqConstraints, DataResponsesRep::numObjectiveFunctions, parallel_interface_plugin(), Environment::parallel_library(), serial_interface_plugin(), and ParallelLibrary::world_rank().

Referenced by main().

void run_dakota_mixed (const char * *dakota_input_file,* bool *mpirun_flag*)

Run a Dakota LibraryEnvironment, from string or input file input, supplemented with additional C++ API adjustments.

Function to encapsulate the Dakota object instantiations for mode 3: mixed parsing and direct updating.

This function showcases multiple features. For parsing, either an input file (dakota_input_file != NULL) or a default input string (dakota_input_file == NULL) are shown. This parsed input is then mixed with input from three sources: (1) input from a user-supplied callback function, (2) updates to the DB prior to Environment instantiation, (3) updates directly to Iterators/Models following Environment instantiation.

References callback_function(), LibraryEnvironment::done_modifying_db(), ProgramOptions::echo_input(), Environment::execute(), LibraryEnvironment::filtered_model_list(), ProblemDescDB::get_sa(), ProgramOptions::input_file(), ProgramOptions::input_string(), Environment::mpi_manager(), MPIManager::mpirun_flag(), parallel_input, parallel_interface_plugin(), Environment::parallel_library(), Environment::problem_description_db(), ProblemDescDB::resolve_top_method(), callback_data::rosen_cdv_upper_bd, serial_input, serial_interface_plugin(), ProblemDescDB::set(), and ParallelLibrary::world_rank().

Referenced by main().

void serial_interface_plugin (Dakota::LibraryEnvironment & *env*)

Convenience function with simplest example of interface plugin: plugin a serial DirectApplicInterface that can be constructed independent of Dakota's configuration details.

Demonstration of simple plugin where client code doesn't require access to detailed Dakota data (such as Model-based parallel configuration information) to construct the DirectApplicInterface. This example plugs-in a derived serial direct application interface instance ("plugin_rosenbrock").

References Dakota::abort_handler(), LibraryEnvironment::plugin_interface(), and Environment::problem_description_db().

Referenced by run_dakota_data(), run_dakota_mixed(), and run_dakota_parse().

void parallel_interface_plugin (Dakota::LibraryEnvironment & *env*)

Convenience function to plug a library client's interface into the appropriate model, demonstrating use of Dakota parallel configuration in constructing the plugin Interface on the right MPI_Comm.

From a filtered list of Model candidates, plug-in a derived direct application interface instance ("plugin_textbook" for parallel). This approach provides more complete access to the Model, e.g., for access to analysis communicators.

References Dakota::abort_handler(), Interface::assign_rep(), LibraryEnvironment::filtered_model_list(), ProblemDescDB::get_db_model_node(), Environment::problem_description_db(), and ProblemDescDB::set_db_model_nodes().

Referenced by run_dakota_data(), run_dakota_mixed(), and run_dakota_parse().

static void callback_function (Dakota::ProblemDescDB * *db,* void * *ptr*) `[static]`

Example: user-provided post-parse callback (Dakota::DbCallbackFunction)

Example of user-provided callback function (an instance of Dakota::DbCallbackFunction) to override input provided by parsed Dakota input file or input string data.

References Dakota::contains(), ProblemDescDB::get_sa(), ProblemDescDB::get_ushort(), ProblemDescDB::resolve_top_method(), callback_data::rosen_cdv_upper_bd, and ProblemDescDB::set().

Referenced by run_dakota_mixed().

int main (int *argc*, char * *argv[ ]*)

A mock simulator main for testing Dakota in library mode.

Overall Usage: dakota_library_mode [-mixed] [dakota.in]

Uses alternative instantiation syntax as described in the library mode documentation within the Developers Manual. Tests several problem specification modes:

(1) run_dakota_parse: reads all problem specification data from a Dakota input file. Usage: dakota_library_-mode dakota.in

(2) run_dakota_data: creates all problem specification from direct Data instance instantiations in the C++ code. Usage: dakota_library_mode

(3) run_dakota_mixed: a mixture of input parsing and direct data updates, where the data updates occur: (a) via the DB during Environment instantiation, and (b) via Iterators/Models following Environment instantiation. Usage: dakota_library_mode -mixed (input from default string) dakota_library_mode -mixed dakota.in (input from specified file)

Serial cases use a plugin rosenbrock model, while parallel cases use textbook.

References MPIManager::detect_parallel_launch(), fpinit_ASL(), Dakota::mpi_debug_hold(), run_dakota_data(), run_dakota_mixed(), and run_dakota_parse().

14.7.3 Variable Documentation

const char serial_input[ ] [static]

Initial value:

```
=
  "     method,"
  "             optpp_q_newton"
  "               max_iterations = 50"
  "               convergence_tolerance = 1e-4"
  "     variables,"
  "             continuous_design = 2"
  "               descriptors 'x1' 'x2'"
  "     interface,"
  "             direct"
  "               analysis_driver = 'plugin_rosenbrock'"
  "     responses,"
  "             num_objective_functions = 1"
  "             analytic_gradients"
  "             no_hessians"
```

Default Dakota input string for serial case (rosenbrock):

Referenced by run_dakota_mixed().

const char parallel_input[ ] [static]

Initial value:

```
=
  "     method,"
  "             optpp_q_newton"
  "               max_iterations = 50"
  "               convergence_tolerance = 1e-4"
  "     variables,"
  "             continuous_design = 2"
  "               descriptors 'x1' 'x2'"
  "     interface,"
  "             direct"
  "               analysis_driver = 'plugin_text_book'"
  "     responses,"
  "             num_objective_functions = 1"
  "             num_nonlinear_inequality_constraints = 2"
```

```
"            analytic_gradients"
"            no_hessians"
```

Default Dakota input string for parallel case (text_book)
Referenced by run_dakota_mixed().

14.8 library_split.cpp File Reference

file containing a mock simulator main for testing DAKOTA in library mode on a split communicator

Functions

- void manage_mpi (MPI_Comm &my_comm, int &color)

  Split MPI_COMM_WORLD, returning the comm and color.
- void gen_dakota_input (const int &color, std::string &input)

  Return the appropriate DAKOTA input based on color (1 or 2)
- void run_dakota (const MPI_Comm &comm, const std::string &input, const int &color)

  Launch DAKOTA on passed communicator, tagging output/error with color.
- void collect_results ()

  Wait for and collect results from DAKOTA runs.
- int main (int argc, char *argv[ ])

  Driver routine for testing library mode with partitioned MPI_Comm. This test fixture requires MPI and can be run on 3–8 processors.

14.8.1 Detailed Description

file containing a mock simulator main for testing DAKOTA in library mode on a split communicator

14.9 main.cpp File Reference

file containing the main program for DAKOTA

Functions

- void fpinit_ASL ()
- int main (int argc, char *argv[ ])

  The main DAKOTA program.

14.9.1 Detailed Description

file containing the main program for DAKOTA

14.9.2 Function Documentation

void fpinit_ASL ( )

Floating-point initialization from AMPL: switch to 53-bit rounding if appropriate, to eliminate some cross-platform differences.

int main (int *argc,* char * *argv[ ]*)

The main DAKOTA program.

Manage command line inputs, input files, restart file(s), output streams, and top level parallel iterator communicators. Instantiate the ExecutableEnvironment and invoke its execute() virtual function.

References Environment::check(), ExecutableEnvironment::execute(), fpinit_ASL(), Dakota::mpi_debug_hold(), and Dakota::register_signal_handlers().

14.10 restart_util.cpp File Reference

file containing the DAKOTA restart utility main program

Namespaces

- Dakota

  The primary namespace for DAKOTA.

Constant Groups

- Dakota

  The primary namespace for DAKOTA.

Functions

- void print_restart (int argc, char **argv, String print_dest)

  print a restart file
- void print_restart_pdb (int argc, char **argv, String print_dest)

  print a restart file (PDB format)
- void print_restart_tabular (int argc, char **argv, String print_dest)

  print a restart file (tabular format)
- void read_neutral (int argc, char **argv)

  read a restart file (neutral file format)
- void repair_restart (int argc, char **argv, String identifier_type)

  repair a restart file by removing corrupted evaluations
- void concatenate_restart (int argc, char **argv)

  concatenate multiple restart files
- int main (int argc, char *argv[ ])

  The main program for the DAKOTA restart utility.

14.10.1 Detailed Description

file containing the DAKOTA restart utility main program

14.10.2 Function Documentation

int main (int *argc*, char * *argv[ ]*)

The main program for the DAKOTA restart utility.

Parse command line inputs and invoke the appropriate utility function (print_restart(), print_restart_tabular(), read_neutral(), repair_restart(), or concatenate_restart()).

References Dakota::concatenate_restart(), Dakota::print_restart(), Dakota::print_restart_pdb(), Dakota::print_restart_tabular(), Dakota::read_neutral(), and Dakota::repair_restart().

Index

~Approximation
Dakota::Approximation, 334
~Constraints
Dakota::Constraints, 377
~DataFitSurrModel
Dakota::DataFitSurrModel, 387
~EffGlobalMinimizer
Dakota::EffGlobalMinimizer, 450
~Environment
Dakota::Environment, 460
~Interface
Dakota::Interface, 498
~Iterator
Dakota::Iterator, 508
~Model
Dakota::Model, 571
~NonDAdaptiveSampling
Dakota::NonDAdaptiveSampling, 634
~ProblemDescDB
Dakota::ProblemDescDB, 773
~SharedApproxData
Dakota::SharedApproxData, 839
_initPts
Dakota::JEGAOptimizer, 525
_model
Dakota::JEGAOptimizer::Evaluator, 468

A
Dakota::CONMINOptimizer, 368
APPSEvalMgr, 342
Dakota::APPSEvalMgr, 344
APPSOptimizer, 344
abort_handler_t
Dakota, 146
abort_mode
Dakota, 151
accepts_multiple_points
Dakota::JEGAOptimizer, 524
ActiveSet, 307
activeVarsCompsTotals
Dakota::SharedVariablesDataRep, 854
actualModel
Dakota::DataFitSurrModel, 392
add_anchor_to_surfdata
Dakota::SurfpackApproximation, 875
add_datapoint
Dakota::Graphics, 485
Dakota::OutputManager, 735
algorithm_space_model
Dakota::Analyzer, 314
Dakota::EffGlobalMinimizer, 451
Dakota::Minimizer, 542
Dakota::NonDBayesCalibration, 636
Dakota::NonDExpansion, 649
Dakota::NonDGlobalInterval, 655
Dakota::NonDReliability, 707
allContinuousIds
Dakota::SharedVariablesDataRep, 854
Analyzer, 309
anisotropic_order_to_dimension_preference
Dakota::NonDIntegration, 669
append_approximation
Dakota::ApproximationInterface, 340
Dakota::DataFitSurrModel, 389, 390
ApplicationInterface, 316
approx_subprob_constraint_eval
Dakota::SurrBasedLocalMinimizer, 882
approx_subprob_objective_eval
Dakota::SurrBasedLocalMinimizer, 882
approxBuilds
Dakota::SurrogateModel, 891
Approximation, 330
Dakota::Approximation, 334
ApproximationInterface, 337
array_insert
Dakota::ResultsDBAny, 819
assess_reconstruction
Dakota::EfficientSubspaceMethod, 454
assign_rep
Dakota::Interface, 499
Dakota::Iterator, 511

Dakota::Model, 576
asstring
Dakota, 148
asynchronous_local_analyses
Dakota::ProcessHandleApplicInterface, 783
asynchronous_local_evaluations
Dakota::ApplicationInterface, 327
asynchronous_local_evaluations_nowait
Dakota::ApplicationInterface, 328
augmented_lagrangian_merit
Dakota::SurrBasedMinimizer, 887
autotag_files
Dakota::ProcessApplicInterface, 778

B
Dakota::CONMINOptimizer, 367
BaseConstructor, 347
brent_minimize
Dakota::NonlinearCGOptimizer, 721
build
Dakota::Approximation, 335
Dakota::PecosApproximation, 764
Dakota::SurfpackApproximation, 874
Dakota::TANA3Approximation, 897
Dakota::TaylorApproximation, 898
build_approximation
Dakota::ApproximationInterface, 341
Dakota::DataFitSurrModel, 388
build_global
Dakota::DataFitSurrModel, 391
build_local_multipoint
Dakota::DataFitSurrModel, 392
build_views
Dakota::Constraints, 378
Dakota::Variables, 916
buildDataOrder
Dakota::SharedApproxData, 840

C
Dakota::CONMINOptimizer, 367
CAUVLbl
Dakota, 300
CEUVLbl
Dakota, 301
COLINApplication, 348
COLINOptimizer, 351
Dakota::COLINOptimizer, 353
CONMINOptimizer, 362
CT
Dakota::CONMINOptimizer, 367
callback_data, 347
callback_function
library_mode.cpp, 932
ccv_index_map
Dakota::NestedModel, 597
cdiv_index_map
Dakota::NestedModel, 597
cdrv_index_map
Dakota::NestedModel, 597
cdsv_index_map
Dakota::NestedModel, 597
check_and_broadcast
Dakota::ProblemDescDB, 774
check_driver
Dakota::NIDRProblemDescDB, 603
check_input
Dakota::ProblemDescDB, 774
check_variables
Dakota::NonDIntegration, 669
check_wait
Dakota::ProcessHandleApplicInterface, 782
clear_all
Dakota::Approximation, 336
clear_current
Dakota::Approximation, 336
Dakota::TANA3Approximation, 897
Clone
Dakota::JEGAOptimizer::Evaluator, 467
colin_cache_lookup
Dakota::COLINOptimizer, 354
colin_request_to_dakota_request
Dakota::COLINApplication, 350
CollabHybridMetaIterator, 355
collect_evaluation_impl
Dakota::COLINApplication, 350
CommandLineHandler, 356
CommandShell, 358
compute
Dakota::DiscrepancyCorrection, 443
compute_correlations
Dakota::SensAnalysisGlobal, 830
compute_covariance_metric
Dakota::NonDExpansion, 649
Dakota::NonDStochCollocation, 718
compute_final_statistics_metric
Dakota::NonDExpansion, 649
Dakota::NonDStochCollocation, 719
compute_statistics
Dakota::NonDExpansion, 650

concatenate_restart
Dakota, 151
ConcurrentMetaIterator, 359
conminInfo
Dakota::CONMINOptimizer, 365
ConsoleRedirector, 369
constraint0_evaluator
Dakota::SNLLOptimizer, 868
constraint1_evaluator
Dakota::SNLLOptimizer, 868
constraint1_evaluator_gn
Dakota::SNLLLeastSq, 862
constraint2_evaluator
Dakota::SNLLOptimizer, 868
constraint2_evaluator_gn
Dakota::SNLLLeastSq, 862
constraint_violation
Dakota::COLINOptimizer, 354
Dakota::SurrBasedMinimizer, 888
constraintMappingIndices
Dakota::CONMINOptimizer, 366
Dakota::DOTOptimizer, 446
constraintMappingMultipliers
Dakota::CONMINOptimizer, 366
Dakota::DOTOptimizer, 446
constraintMappingOffsets
Dakota::CONMINOptimizer, 366
Dakota::DOTOptimizer, 447
constraintValues
Dakota::CONMINOptimizer, 366
Dakota::DOTOptimizer, 446
Constraints, 370
Dakota::Constraints, 376, 377
contains
Dakota, 151
converge_order
Dakota::RichExtrapVerification, 827
converge_qoi
Dakota::RichExtrapVerification, 827
copy
Dakota::Constraints, 377
Dakota::SharedVariablesData, 850
copy_data
Dakota::DDACEDesignCompExp, 435
copy_items
Dakota::WorkdirHelper, 923
core_run
Dakota::Iterator, 509
Dakota::LeastSq, 527
Dakota::NonD, 623
Dakota::Optimizer, 730
Dakota::PStudyDACE, 790
Dakota::SurrBasedMinimizer, 886
Dakota::Verification, 917
create_command_arguments
Dakota::ProcessHandleApplicInterface, 783
create_directory
Dakota::WorkdirHelper, 922
create_evaluation_process
Dakota::ProcessHandleApplicInterface, 782
create_plots_2d
Dakota::Graphics, 485
create_tabular_datastream
Dakota::OutputManager, 735
CreateEvaluator
Dakota::JEGAOptimizer::EvaluatorCreator, 468
cv_index_map
Dakota::NestedModel, 596

DAUIVLbl
Dakota, 300
DAURVLbl
Dakota, 301
DAUSVLbl
Dakota, 300
DDACEDesignCompExp, 432
Dakota::DDACEDesignCompExp, 434
DEUIVLbl
Dakota, 301
DEURVLbl
Dakota, 301
DEUSVLbl
Dakota, 301
DF
Dakota::CONMINOptimizer, 368
DOTOptimizer, 443
DakFuncs0
Dakota, 151
Dakota, 71
abort_handler_t, 146
abort_mode, 151
asstring, 148
CAUVLbl, 300
CEUVLbl, 301
concatenate_restart, 151
contains, 151
DAUIVLbl, 300
DAURVLbl, 301
DAUSVLbl, 300

DEUIVLbl, 301
DEURVLbl, 301
DEUSVLbl, 301
DakFuncs0, 151
DesignAndStateLabelsCheck, 302
DiscSetLbl, 301
FIELD_NAMES, 152
flush, 146
get_pathext, 151
getRmax, 148
getdist, 147
id_vars_exact_compare, 149
kw_1, 152, 230
kw_10, 154, 232
kw_100, 175, 249
kw_101, 175, 249
kw_102, 175, 249
kw_103, 175, 249
kw_104, 175, 249
kw_105, 175, 250
kw_106, 176, 250
kw_107, 176, 250
kw_108, 176, 250
kw_109, 176, 250
kw_11, 154, 232
kw_110, 176, 250
kw_111, 177, 251
kw_112, 177, 251
kw_113, 177, 251
kw_114, 177, 251
kw_115, 177, 251
kw_116, 178, 252
kw_117, 178, 252
kw_118, 178, 252
kw_119, 178, 252
kw_12, 154, 232
kw_120, 178, 252
kw_121, 178, 252
kw_122, 179, 253
kw_123, 179, 253
kw_124, 179, 253
kw_125, 179, 253
kw_126, 179, 253
kw_127, 180, 253
kw_128, 180, 254
kw_129, 180, 254
kw_13, 154, 232
kw_130, 180, 254
kw_131, 180, 254
kw_132, 181, 254
kw_133, 181, 255
kw_134, 181, 255
kw_135, 181, 255
kw_136, 182, 255
kw_137, 182, 255
kw_138, 182, 255
kw_139, 182, 256
kw_14, 154, 232
kw_140, 182, 256
kw_141, 182, 256
kw_142, 183, 256
kw_143, 183, 256
kw_144, 183, 256
kw_145, 183, 257
kw_146, 183, 257
kw_147, 183, 257
kw_148, 184, 257
kw_149, 184, 257
kw_15, 155, 233
kw_150, 184, 257
kw_151, 184, 258
kw_152, 184, 258
kw_153, 185, 258
kw_154, 185, 258
kw_155, 185, 258
kw_156, 186, 258
kw_157, 186, 259
kw_158, 186, 259
kw_159, 186, 259
kw_16, 155, 233
kw_160, 186, 259
kw_161, 187, 259
kw_162, 187, 260
kw_163, 187, 260
kw_164, 187, 260
kw_165, 187, 260
kw_166, 187, 260
kw_167, 188, 261
kw_168, 188, 261
kw_169, 188, 261
kw_17, 155, 233
kw_170, 189, 261
kw_171, 189, 261
kw_172, 189, 262
kw_173, 189, 262
kw_174, 190, 262
kw_175, 190, 262
kw_176, 190, 262

kw_177, 190, 262
kw_178, 190, 263
kw_179, 191, 263
kw_18, 156, 233
kw_180, 191, 263
kw_181, 191, 263
kw_182, 191, 263
kw_183, 191, 264
kw_184, 192, 264
kw_185, 192, 264
kw_186, 192, 264
kw_187, 192, 264
kw_188, 192, 265
kw_189, 192, 265
kw_19, 156, 233
kw_190, 193, 265
kw_191, 193, 265
kw_192, 193, 265
kw_193, 193, 265
kw_194, 193, 266
kw_195, 194, 266
kw_196, 194, 266
kw_197, 194, 266
kw_198, 194, 266
kw_199, 194, 266
kw_2, 152, 230
kw_20, 156, 234
kw_200, 195, 267
kw_201, 195, 267
kw_202, 195, 267
kw_203, 195, 267
kw_204, 195, 267
kw_205, 267
kw_206, 196, 268
kw_207, 196, 268
kw_208, 196, 268
kw_209, 196, 268
kw_21, 156, 234
kw_210, 197, 269
kw_211, 197, 269
kw_212, 197, 269
kw_213, 197, 269
kw_214, 197, 269
kw_215, 197, 270
kw_216, 198, 270
kw_217, 198, 270
kw_218, 198, 270
kw_219, 198, 270
kw_22, 156, 234
kw_220, 199, 271
kw_221, 199, 271
kw_222, 200, 271
kw_223, 200, 271
kw_224, 200, 272
kw_225, 200, 272
kw_226, 201, 272
kw_227, 201, 272
kw_228, 201, 272
kw_229, 201, 272
kw_23, 157, 234
kw_230, 202, 273
kw_231, 202, 273
kw_232, 202, 273
kw_233, 202, 273
kw_234, 203, 274
kw_235, 203, 274
kw_236, 203, 274
kw_237, 203, 274
kw_238, 203, 274
kw_239, 204, 274
kw_24, 157, 234
kw_240, 205, 275
kw_241, 205, 275
kw_242, 205, 275
kw_243, 205, 275
kw_244, 205, 275
kw_245, 206, 276
kw_246, 206, 276
kw_247, 206, 276
kw_248, 276
kw_249, 207, 276
kw_25, 157, 234
kw_250, 207, 276
kw_251, 207, 277
kw_252, 207, 277
kw_253, 207, 277
kw_254, 208, 277
kw_255, 208, 277
kw_256, 208, 278
kw_257, 208, 278
kw_258, 208, 278
kw_259, 208, 278
kw_26, 157, 235
kw_260, 209, 278
kw_261, 209, 279
kw_262, 209, 279
kw_263, 209, 279
kw_264, 209

kw_265, 210, 279
kw_266, 210, 279
kw_267, 210, 280
kw_268, 210, 280
kw_269, 210, 280
kw_27, 157, 235
kw_270, 211, 280
kw_271, 211, 280
kw_272, 211, 280
kw_273, 212, 281
kw_274, 212, 281
kw_275, 212, 281
kw_276, 212, 281
kw_277, 212, 281
kw_278, 213, 281
kw_279, 213, 282
kw_28, 157, 235
kw_280, 213, 282
kw_281, 213, 282
kw_282, 214, 282
kw_283, 214, 282
kw_284, 214, 282
kw_285, 215, 283
kw_286, 215, 283
kw_287, 215, 283
kw_288, 215, 283
kw_289, 216, 284
kw_29, 158, 235
kw_290, 216, 284
kw_291, 216, 284
kw_292, 216, 284
kw_293, 216, 284
kw_294, 217, 284
kw_295, 217, 285
kw_296, 217, 285
kw_297, 218, 285
kw_298, 218, 285
kw_299, 218, 285
kw_3, 152, 230
kw_30, 158, 235
kw_300, 218, 286
kw_301, 218, 286
kw_302, 219, 286
kw_303, 219, 287
kw_304, 220, 287
kw_305, 220, 287
kw_306, 220, 287
kw_307, 220, 287
kw_308, 221, 287
kw_309, 221, 288
kw_31, 158, 235
kw_310, 221, 288
kw_311, 221, 288
kw_312, 222, 288
kw_313, 222, 289
kw_314, 222, 289
kw_315, 222, 289
kw_316, 223, 289
kw_317, 223, 290
kw_318, 223, 290
kw_319, 223, 290
kw_32, 158, 236
kw_320, 224, 290
kw_321, 224, 291
kw_322, 224, 291
kw_323, 224, 291
kw_324, 225, 291
kw_325, 225, 292
kw_326, 225, 292
kw_327, 225, 292
kw_328, 226, 292
kw_329, 226, 293
kw_33, 158, 236
kw_330, 226, 293
kw_331, 226, 293
kw_332, 227, 293
kw_333, 227, 294
kw_334, 227, 294
kw_335, 227, 294
kw_336, 227, 294
kw_337, 228, 294
kw_338, 228, 295
kw_339, 228, 295
kw_34, 159, 236
kw_340, 229, 295
kw_341, 229, 295
kw_342, 229, 295
kw_343, 229, 296
kw_344, 296
kw_345, 230, 296
kw_346, 296
kw_347, 297
kw_348, 297
kw_349, 297
kw_35, 159, 236
kw_350, 297
kw_351, 297
kw_352, 298

kw_353, 298
kw_354, 298
kw_355, 298
kw_356, 298
kw_357, 299
kw_358, 299
kw_359, 299
kw_36, 159, 236
kw_360, 299
kw_362, 300
kw_37, 160, 237
kw_38, 160, 237
kw_39, 160, 237
kw_4, 152, 231
kw_40, 161, 237
kw_41, 161, 238
kw_42, 161, 238
kw_43, 161, 238
kw_44, 161, 238
kw_45, 161, 238
kw_46, 162, 239
kw_47, 162, 239
kw_48, 162, 239
kw_49, 162, 239
kw_5, 153, 231
kw_50, 162, 239
kw_51, 162, 239
kw_52, 163, 240
kw_53, 163, 240
kw_54, 163, 240
kw_55, 163, 240
kw_56, 163, 240
kw_57, 163, 240
kw_58, 164, 241
kw_59, 164, 241
kw_6, 153, 231
kw_60, 164, 241
kw_61, 164, 241
kw_62, 165, 241
kw_63, 165, 241
kw_64, 165, 242
kw_65, 166, 242
kw_66, 166, 242
kw_67, 166, 242
kw_68, 166, 242
kw_69, 166, 243
kw_7, 153, 231
kw_70, 167, 243
kw_71, 167, 243
kw_72, 167, 243
kw_73, 168, 243
kw_74, 168, 243
kw_75, 168, 244
kw_76, 168, 244
kw_77, 169, 244
kw_78, 169, 244
kw_79, 170, 244
kw_8, 153, 231
kw_80, 170, 245
kw_81, 170, 245
kw_82, 171, 245
kw_83, 171, 245
kw_84, 171, 245
kw_85, 171, 246
kw_86, 172, 246
kw_87, 172, 246
kw_88, 172, 246
kw_89, 172, 246
kw_9, 153, 232
kw_90, 173, 247
kw_91, 173, 247
kw_92, 173, 247
kw_93, 173, 247
kw_94, 173, 247
kw_95, 174, 248
kw_96, 174, 248
kw_97, 174, 248
kw_98, 174, 248
kw_99, 174, 248
lookup_by_val, 149
mindist, 148
mindistindx, 148
mpi_debug_hold, 146
NUMBER_OF_FIELDS, 152
operator==, 147, 149
PRPMultiIndexCache, 145
PRPMultiIndexQueue, 145
perform_analysis, 148
print_restart, 150
print_restart_pdb, 150
print_restart_tabular, 150
read_neutral, 150
register_signal_handlers, 146
repair_restart, 150
SUBMETHOD_COLLABORATIVE, 146
SCI_FIELD_NAMES, 305
set_compare, 149
start_dakota_heartbeat, 148

start_grid_computing, 148
stop_grid_computing, 148
VLUncertainInt, 302
VLUncertainReal, 302
VLUncertainStr, 303
var_mp_cbound, 305
var_mp_check_cau, 303
var_mp_check_ceu, 304
var_mp_check_cv, 303
var_mp_check_daui, 304
var_mp_check_daur, 304
var_mp_check_daus, 304
var_mp_check_deui, 304
var_mp_check_deur, 305
var_mp_check_deus, 304
var_mp_check_dset, 303
var_mp_drange, 305
write_ordered, 147
Dakota::GetLongOpt
MandatoryValue, 482
OptionalValue, 482
Valueless, 482
Dakota::APPSEvalMgr
APPSEvalMgr, 344
isReadyForWork, 344
recv, 344
submit, 344
Dakota::APPSOptimizer
find_optimum, 346
initialize_variables_and_constraints, 346
set_apps_parameters, 346
Dakota::ActiveSet
derivVarsVector, 309
requestVector, 309
Dakota::Analyzer
algorithm_space_model, 314
evaluate_parameter_sets, 314
initialize_run, 313
num_samples, 313
post_run, 313
pre_output, 314
print_results, 314
print_sobol_indices, 315
read_variables_responses, 315
sample_to_variables, 313
variables_to_sample, 315
variance_based_decomp, 315
Dakota::ApplicationInterface
asynchronous_local_evaluations, 327
asynchronous_local_evaluations_nowait, 328
duplication_detect, 325
init_communicators_checks, 325
init_serial, 323
map, 323
master_dynamic_schedule_analyses, 325
master_dynamic_schedule_evaluations, 326
master_dynamic_schedule_evaluations_nowait, 328
peer_dynamic_schedule_evaluations, 327
peer_dynamic_schedule_evaluations_nowait, 328
peer_static_schedule_evaluations, 326
serve_analyses_synch, 325
serve_evaluations, 324
serve_evaluations_asynch, 329
serve_evaluations_asynch_peer, 329
serve_evaluations_synch, 329
serve_evaluations_synch_peer, 329
set_communicators_checks, 325
stop_evaluation_servers, 324
synch, 323
synch_nowait, 324
synchronous_local_evaluations, 327
Dakota::Approximation
~Approximation, 334
Approximation, 334
build, 335
clear_all, 336
clear_current, 336
finalize, 336
get_approx, 336
operator=, 335
pop, 335
rebuild, 335
restore, 335
Dakota::ApproximationInterface
append_approximation, 340
build_approximation, 341
functionSurfaces, 342
pop_approximation, 341
read_challenge_points, 341
rebuild_approximation, 341
restore_approximation, 341
update_approximation, 339, 340
Dakota::COLINApplication
colin_request_to_dakota_request, 350
collect_evaluation_impl, 350
dakota_response_to_colin_response, 350
evaluation_available, 349
map_domain, 350

perform_evaluation_impl, 350
set_problem, 349
spawn_evaluation_impl, 349
Dakota::COLINOptimizer
COLINOptimizer, 353
colin_cache_lookup, 354
constraint_violation, 354
find_optimum, 353
post_run, 354
returns_multiple_points, 353
set_rng, 354
set_solver_parameters, 354
solver_setup, 353
Dakota::CONMINOptimizer
A, 368
B, 367
C, 367
CT, 367
conminInfo, 365
constraintMappingIndices, 366
constraintMappingMultipliers, 366
constraintMappingOffsets, 366
constraintValues, 366
DF, 368
G1, 367
G2, 367
IC, 368
ISC, 368
MS1, 368
N1, 366
N2, 366
N3, 366
N4, 367
N5, 367
printControl, 365
S, 367
SCAL, 368
Dakota::CommandLineHandler
output_helper, 357
Dakota::CommandShell
flush, 359
operator<<, 359
Dakota::ConcurrentMetaIterator
pre_run, 361
print_results, 362
Dakota::Constraints
~Constraints, 377
build_views, 378
Constraints, 376, 377
copy, 377
get_constraints, 379
manage_linear_constraints, 378
operator=, 377
reshape, 378
shape, 378
Dakota::DDACEDesignCompExp
copy_data, 435
DDACEDesignCompExp, 434
num_samples, 435
post_run, 435
pre_run, 435
resolve_samples_symbols, 435
Dakota::DOTOptimizer
constraintMappingIndices, 446
constraintMappingMultipliers, 446
constraintMappingOffsets, 447
constraintValues, 446
dotFDSinfo, 445
dotInfo, 445
dotMethod, 446
intCntlParmArray, 446
printControl, 446
realCntlParmArray, 446
Dakota::DataFitSurrModel
~DataFitSurrModel, 387
actualModel, 392
append_approximation, 389, 390
build_approximation, 388
build_global, 391
build_local_multipoint, 392
derived_asynch_compute_response, 387
derived_compute_response, 387
derived_init_communicators, 390
derived_synchronize, 387
derived_synchronize_nowait, 388
evaluation_id, 390
export_point, 391
import_points, 391
initialize_export, 391
manage_data_recastings, 391
update_actual_model, 392
update_approximation, 389
update_from_actual_model, 392
Dakota::DirectApplicInterface
derived_map_ac, 439
init_communicators_checks, 439
set_communicators_checks, 439
synchronous_local_analysis, 439

Dakota::DiscrepancyCorrection
compute, 443
Dakota::EffGlobalMinimizer
~EffGlobalMinimizer, 450
algorithm_space_model, 451
get_best_sample, 451
Dakota::EfficientSubspaceMethod
assess_reconstruction, 454
derived_init_communicators, 454
map_xi_to_x, 455
reduced_space_uq, 454
uncertain_vars_to_subspace, 454
Dakota::Environment
~Environment, 460
Environment, 459, 460
exit_mode, 460
get_environment, 461
operator=, 460
parse, 461
Dakota::ExperimentData
load_scalar, 470
Dakota::FSUDesignCompExp
enforce_input_rules, 476
FSUDesignCompExp, 475
num_samples, 475
post_run, 475
pre_run, 475
Dakota::GaussProcApproximation
GPmodel_apply, 480
GaussProcApproximation, 480
trendOrder, 481
Dakota::GetLongOpt
enroll, 483
GetLongOpt, 483
OptType, 482
parse, 483
retrieve, 483
usage, 484
Dakota::Graphics
add_datapoint, 485
create_plots_2d, 485
new_dataset, 485
Dakota::GridApplicInterface
synchronous_local_analysis, 487
Dakota::HierarchSurrModel
derived_asynch_compute_response, 490
derived_compute_response, 490
derived_synchronize, 490
derived_synchronize_nowait, 491
evaluation_id, 491
Dakota::Interface
~Interface, 498
assign_rep, 499
eval_tag_prefix, 499
get_interface, 500
Interface, 498
operator=, 499
rawResponseMap, 500
response_mapping, 499
Dakota::Iterator
~Iterator, 508
assign_rep, 511
core_run, 509
eval_tag_prefix, 511
finalize_run, 510
get_iterator, 512
initialize_graphics, 510
initialize_run, 509
Iterator, 507, 508
maxEvalConcurrency, 513
operator=, 509
post_run, 509
pre_run, 509
print_results, 510
probDescDB, 512
run, 511
uses_method, 510
Dakota::IteratorScheduler
free_iterator, 517
init_evaluation_concurrency, 518
init_iterator, 516, 517
init_iterator_parallelism, 518
IteratorScheduler, 516
master_dynamic_schedule_iterators, 519
run_iterator, 517
schedule_iterators, 518
serve_iterators, 519
set_iterator, 517
Dakota::JEGAOptimizer
_initPts, 525
accepts_multiple_points, 524
find_optimum, 524
GetBestMOSolutions, 523
GetBestSOSolutions, 523
GetBestSolutions, 523
initial_points, 524
JEGAOptimizer, 521
LoadAlgorithmConfig, 522

LoadDakotaResponses, 521
LoadProblemConfig, 522
LoadTheConstraints, 523
LoadTheDesignVariables, 522
LoadTheObjectiveFunctions, 522
LoadTheParameterDatabase, 522
returns_multiple_points, 524
ToDoubleMatrix, 523
Dakota::JEGAOptimizer::Driver
DestroyAlgorithm, 448
Driver, 447
ExtractAllData, 448
PerformIterations, 448
Dakota::JEGAOptimizer::Evaluator
_model, 468
Clone, 467
Description, 465
Evaluate, 466, 467
Evaluator, 464, 465
GetDescription, 467
GetName, 467
GetNumberLinearConstraints, 466
GetNumberNonLinearConstraints, 466
Name, 465
RecordResponses, 466
SeparateVariables, 465
Dakota::JEGAOptimizer::EvaluatorCreator
CreateEvaluator, 468
EvaluatorCreator, 468
Dakota::LeastSq
core_run, 527
finalize_run, 527
get_confidence_intervals, 528
initialize_run, 527
LeastSq, 527
post_run, 527
primary_resp_weighter, 528
print_results, 528
weight_model, 528
Dakota::LibraryEnvironment
filtered_interface_list, 530
filtered_model_list, 530
LibraryEnvironment, 530
Dakota::MatlabInterface
derived_map_ac, 533
matlab_engine_run, 533
Dakota::MetaIterator
post_run, 534
Dakota::Minimizer
algorithm_space_model, 542
data_transform_model, 542
finalize_run, 542
gnewton_set_recast, 542
initialize_run, 541
initialize_scaling, 545
lin_coeffs_modify_n2s, 546
Minimizer, 541
modify_n2s, 546
modify_s2n, 543
need_resp_trans_byvars, 543
objective, 543, 544
objective_gradient, 544
objective_hessian, 544
post_run, 541
primary_resp_differencer, 545
resize_best_resp_array, 545
resize_best_vars_array, 544
response_modify_n2s, 546
response_modify_s2n, 543
scale_model, 542
secondary_resp_copier, 543
secondary_resp_scaler, 546
variables_scaler, 545
Dakota::MixedVarConstraints
MixedVarConstraints, 547
Dakota::MixedVariables
MixedVariables, 549
read_tabular, 549
Dakota::Model
~Model, 571
assign_rep, 576
derivative_concurrency, 576
derived_interface, 573
estimate_derivatives, 577
estimate_message_lengths, 576
eval_tag_prefix, 574
evaluation_cache, 574
FDstep1, 577
FDstep2, 577
fdGradStepSize, 579
fdHessByFnStepSize, 579
fdHessByGradStepSize, 579
get_model, 577
init_communicators, 575
init_serial, 575
initialize_h, 577
interface_id, 574
local_eval_concurrency, 574

local_eval_synchronization, 573
manage_asv, 579
Model, 571
operator=, 572
probDescDB, 579
subordinate_iterator, 572
subordinate_model, 572
subordinate_models, 575
surrogate_model, 572
synchronize_derivatives, 578
truth_model, 573
update_from_subordinate_model, 573
update_quasi_hessians, 578
update_response, 578
Dakota::NCSUOptimizer
NCSUOptimizer, 587
objective_eval, 588
Dakota::NIDRProblemDescDB
check_driver, 603
derived_parse_inputs, 603
make_variable_defaults, 603
Dakota::NL2SOLLeastSq
minimize_residuals, 606
Dakota::NLSSOLLeastSq
NLSSOLLeastSq, 612
Dakota::NPSOLOptimizer
NPSOLOptimizer, 724
Dakota::NestedModel
ccv_index_map, 597
cdiv_index_map, 597
cdrv_index_map, 597
cdsv_index_map, 597
cv_index_map, 596
derived_asynch_compute_response, 594
derived_compute_response, 594
derived_init_communicators, 595
derived_master_overload, 595
derived_synchronize, 595
div_index_map, 596
drv_index_map, 597
dsv_index_map, 596
evaluation_id, 596
local_eval_concurrency, 595
local_eval_synchronization, 595
response_mapping, 598
subModel, 598
Dakota::NomadOptimizer
load_parameters, 615
NomadOptimizer, 615
Dakota::NomadOptimizer::Evaluator
eval_x, 462
Evaluator, 462
Dakota::NonD
core_run, 623
finalize_run, 623
initialize_final_statistics, 624
initialize_random_variable_parameters, 624
initialize_random_variable_types, 624
initialize_random_variables, 623
initialize_run, 623
print_distribution_map, 625
print_distribution_mappings, 624
set_u_to_x_mapping, 625
vars_u_to_x_mapping, 625
vars_x_to_u_mapping, 625
Dakota::NonDAdaptImpSampling
initialize, 629
NonDAdaptImpSampling, 628
Dakota::NonDAdaptiveSampling
~NonDAdaptiveSampling, 634
NonDAdaptiveSampling, 634
Dakota::NonDBayesCalibration
algorithm_space_model, 636
NonDBayesCalibration, 636
Dakota::NonDCalibration
NonDCalibration, 638
Dakota::NonDCubature
increment_grid_preference, 640
increment_reference, 640
NonDCubature, 640
num_samples, 640
sampling_reset, 640
Dakota::NonDDREAMBayesCalibration
likelihoodScale, 644
NonDDREAMBayesCalibration, 643
prior_density, 643
prior_sample, 643
problem_size, 643
problem_value, 643
quantify_uncertainty, 644
sample_likelihood, 644
Dakota::NonDExpansion
algorithm_space_model, 649
compute_covariance_metric, 649
compute_final_statistics_metric, 649
compute_statistics, 650
increment_specification_sequence, 649
useDerivs, 650

Dakota::NonDGPImpSampling
NonDGPImpSampling, 661
quantify_uncertainty, 661
Dakota::NonDGPMSABayesCalibration
NonDGPMSABayesCalibration, 664
quantify_uncertainty, 664
randomSeed, 665
Dakota::NonDGlobalInterval
algorithm_space_model, 655
Dakota::NonDIncremLHSSampling
NonDIncremLHSSampling, 666
quantify_uncertainty, 666
Dakota::NonDIntegration
anisotropic_order_to_dimension_preference, 669
check_variables, 669
dimension_preference_to_anisotropic_order, 669
NonDIntegration, 669
Dakota::NonDLHSSampling
NonDLHSSampling, 675, 676
quantify_uncertainty, 676
Dakota::NonDLocalReliability
dg_ds_eval, 690
dp2_dbeta_factor, 690
initial_taylor_series, 688
initialize_class_data, 688
initialize_level_data, 688
initialize_mpp_search_data, 689
PMA2_constraint_eval, 687
PMA_constraint_eval, 687
PMA_objective_eval, 687
probability, 690
RIA_constraint_eval, 686
RIA_objective_eval, 686
update_level_data, 689
update_mpp_search_data, 689
Dakota::NonDPOFDarts
NonDPOFDarts, 695
Dakota::NonDPolynomialChaos
increment_order, 698
increment_specification_sequence, 698
NonDPolynomialChaos, 698
Dakota::NonDQUESOBayesCalibration
NonDQUESOBayesCalibration, 704
quantify_uncertainty, 705
Dakota::NonDQuadrature
initialize_grid, 702
NonDQuadrature, 701, 702
num_samples, 702
sampling_reset, 702
Dakota::NonDReliability
algorithm_space_model, 707
Dakota::NonDSampling
get_parameter_sets, 711, 712
mode_counts, 713
NonDSampling, 711
num_samples, 711
sampling_reset, 711
variables_to_sample, 712
view_aleatory_uncertain_counts, 712
view_design_counts, 712
view_epistemic_uncertain_counts, 713
view_uncertain_counts, 713
Dakota::NonDSparseGrid
NonDSparseGrid, 716
num_samples, 716
sampling_reset, 716
Dakota::NonDStochCollocation
compute_covariance_metric, 718
compute_final_statistics_metric, 719
NonDStochCollocation, 718
Dakota::NonlinearCGOptimizer
brent_minimize, 721
Dakota::Optimizer
core_run, 730
finalize_run, 730
initialize_run, 730
local_objective_recast_retrieve, 732
objective_reduction, 731
post_run, 730
primary_resp_reducer, 731
print_results, 731
reduce_model, 731
Dakota::OutputManager
add_datapoint, 735
create_tabular_datastream, 735
OutputManager, 734
pop_output_tag, 735
Dakota::PSUADEDesignCompExp
enforce_input_rules, 793
num_samples, 793
PSUADEDesignCompExp, 792
post_run, 793
pre_run, 792
Dakota::PStudyDACE
core_run, 790
print_results, 790
volumetric_quality, 790
Dakota::ParallelConfiguration

mi_parallel_level, 738
mi_parallel_level_iterator, 738
Dakota::ParallelLibrary
increment_parallel_configuration, 750
init_communicators, 751
init_mpi_comm, 750
ParallelLibrary, 749
push_output_tag, 750
resolve_inputs, 751
terminate_modelcenter, 750
Dakota::ParamResponsePair
evalInterfaceIds, 754
ParamResponsePair, 754
read, 754
write, 754
Dakota::ParamStudy
distribute_list_of_points, 759
load_distribute_points, 759
post_run, 759
pre_run, 758
Dakota::PecosApproximation
build, 764
finalize, 765
pop, 764
rebuild, 764
restore, 765
Dakota::ProblemDescDB
~ProblemDescDB, 773
check_and_broadcast, 774
check_input, 774
get_db, 775
operator=, 774
parse_inputs, 774
post_process, 775
ProblemDescDB, 773
Dakota::ProcessApplicInterface
autotag_files, 778
file_cleanup, 778
prepare_process_environment, 779
reset_process_environment, 779
synchronous_local_analyses, 779
Dakota::ProcessHandleApplicInterface
asynchronous_local_analyses, 783
check_wait, 782
create_command_arguments, 783
create_evaluation_process, 782
init_communicators_checks, 782
ProcessHandleApplicInterface, 781
serve_analyses_asynch, 783
set_communicators_checks, 782
synchronous_local_analysis, 782
Dakota::ProgramOptions
split_filenames, 788
Dakota::PythonInterface
derived_map_ac, 795
python_convert_int, 795
Dakota::RecastModel
derived_compute_response, 802
eval_tag_prefix, 802
initialize, 802
RecastModel, 801
update_from_sub_model, 803
Dakota::RelaxedVarConstraints
RelaxedVarConstraints, 804
Dakota::RelaxedVariables
read_tabular, 806
RelaxedVariables, 806
Dakota::Response
load, 810
save, 810
Dakota::ResponseRep
functionGradients, 816
load, 816
read, 814, 815
read_annotated, 814
read_tabular, 814
reset, 815
reset_inactive, 816
reshape, 815
ResponseRep, 814
save, 816
update, 815
update_partial, 815
write_annotated, 814
write_tabular, 815
Dakota::ResultsDBAny
array_insert, 819
extract_data, 819
insert, 819
Dakota::ResultsEntry
ResultsEntry, 821
Dakota::RichExtrapVerification
converge_order, 827
converge_qoi, 827
estimate_order, 827
print_results, 826
Dakota::SNLLLeastSq
constraint1_evaluator_gn, 862

constraint2_evaluator_gn, 862
nlf2_evaluator_gn, 861
post_run, 861
Dakota::SNLLOptimizer
constraint0_evaluator, 868
constraint1_evaluator, 868
constraint2_evaluator, 868
nlf0_evaluator, 867
nlf1_evaluator, 867
nlf2_evaluator, 868
SNLLOptimizer, 866, 867
Dakota::SensAnalysisGlobal
compute_correlations, 830
Dakota::SeqHybridMetaIterator
extract_parameter_sets, 834
print_results, 833
run_sequential, 833
run_sequential_adaptive, 833
Dakota::SharedApproxData
~SharedApproxData, 839
buildDataOrder, 840
get_shared_data, 840
operator=, 839
SharedApproxData, 838, 839
Dakota::SharedSurfpackApproxData
SharedSurfpackApproxData, 844
Dakota::SharedVariablesData
copy, 850
Dakota::SharedVariablesDataRep
activeVarsCompsTotals, 854
allContinuousIds, 854
inactiveVarsCompsTotals, 854
SharedVariablesDataRep, 854
variablesCompsTotals, 854
Dakota::SingleModel
eval_tag_prefix, 857
Dakota::SurfpackApproximation
add_anchor_to_surfdata, 875
build, 874
hessian, 874
SurfpackApproximation, 874
surrogates_to_surf_data, 875
Dakota::SurrBasedLocalMinimizer
approx_subprob_constraint_eval, 882
approx_subprob_objective_eval, 882
hard_convergence_check, 881
hom_constraint_eval, 883
hom_objective_eval, 883
minimize_surrogates, 880
tr_ratio_check, 881
update_penalty, 882
Dakota::SurrBasedMinimizer
augmented_lagrangian_merit, 887
constraint_violation, 888
core_run, 886
lagrangian_merit, 887
penalty_merit, 887
print_results, 886
update_augmented_lagrange_multipliers, 886
update_filter, 887
update_lagrange_multipliers, 886
Dakota::SurrogateModel
approxBuilds, 891
force_rebuild, 891
responseMode, 891
Dakota::SysCallApplicInterface
init_communicators_checks, 894
set_communicators_checks, 894
spawn_analysis_to_shell, 894
spawn_evaluation_to_shell, 894
spawn_input_filter_to_shell, 894
spawn_output_filter_to_shell, 895
synchronous_local_analysis, 893
test_local_evaluations, 893
wait_local_evaluations, 893
Dakota::TANA3Approximation
build, 897
clear_current, 897
Dakota::TaylorApproximation
build, 898
Dakota::TestDriverInterface
derived_map_ac, 901
herbie, 902
herbie1D, 901
mc_api_run, 902
separable_combine, 902
shubert1D, 902
smooth_herbie, 902
smooth_herbie1D, 902
Dakota::TrackerHTTP
send_data_using_get, 904
send_data_using_post, 904
Dakota::UsageTracker
UsageTracker, 906
Dakota::Variables
build_views, 916
discrete_string_variables_view, 916
Dakota::Verification

core_run, 917
print_results, 917
Dakota::WorkdirHelper
copy_items, 923
create_directory, 922
file_op_items, 923
init_preferred_env_path, 924
init_startup_path, 924
initialize, 922
link, 923
link_items, 922
po_which, 924
prepend_path_item, 923
prepend_preferred_env_path, 922
recursive_copy, 923
set_preferred_path, 924
split_wildcard, 922
tokenize_env_path, 924
which, 922
dakota_dll_api.cpp, 925
dakota_stop, 926
dakota_dll_api.h, 926
dakota_stop, 927
dakota_response_to_colin_response
Dakota::COLINApplication, 350
dakota_stop
dakota_dll_api.cpp, 926
dakota_dll_api.h, 927
dakota_tabular_io.hpp, 927
data_transform_model
Dakota::Minimizer, 542
DataEnvironment, 379
DataEnvironmentRep, 380
DataFitSurrModel, 382
DataInterface, 393
DataMethod, 394
DataMethodRep, 395
DataModel, 408
DataModelRep, 409
DataResponses, 413
DataResponsesRep, 414
DataVariables, 418
DataVariablesRep, 419
derivVarsVector
Dakota::ActiveSet, 309
derivative_concurrency
Dakota::Model, 576
derived_asynch_compute_response
Dakota::DataFitSurrModel, 387
Dakota::HierarchSurrModel, 490
Dakota::NestedModel, 594
derived_compute_response
Dakota::DataFitSurrModel, 387
Dakota::HierarchSurrModel, 490
Dakota::NestedModel, 594
Dakota::RecastModel, 802
derived_init_communicators
Dakota::DataFitSurrModel, 390
Dakota::EfficientSubspaceMethod, 454
Dakota::NestedModel, 595
derived_interface
Dakota::Model, 573
derived_map_ac
Dakota::DirectApplicInterface, 439
Dakota::MatlabInterface, 533
Dakota::PythonInterface, 795
Dakota::TestDriverInterface, 901
derived_master_overload
Dakota::NestedModel, 595
derived_parse_inputs
Dakota::NIDRProblemDescDB, 603
derived_synchronize
Dakota::DataFitSurrModel, 387
Dakota::HierarchSurrModel, 490
Dakota::NestedModel, 595
derived_synchronize_nowait
Dakota::DataFitSurrModel, 388
Dakota::HierarchSurrModel, 491
Description
Dakota::JEGAOptimizer::Evaluator, 465
DesignAndStateLabelsCheck
Dakota, 302
DestroyAlgorithm
Dakota::JEGAOptimizer::Driver, 448
dg_ds_eval
Dakota::NonDLocalReliability, 690
dimension_preference_to_anisotropic_order
Dakota::NonDIntegration, 669
DirectApplicInterface, 436
DiscSetLbl
Dakota, 301
DiscrepancyCorrection, 440
discrete_string_variables_view
Dakota::Variables, 916
distribute_list_of_points
Dakota::ParamStudy, 759
div_index_map
Dakota::NestedModel, 596

dll_tester.cpp, 929
dotFDSinfo
Dakota::DOTOptimizer, 445
dotInfo
Dakota::DOTOptimizer, 445
dotMethod
Dakota::DOTOptimizer, 446
dp2_dbeta_factor
Dakota::NonDLocalReliability, 690
Driver
Dakota::JEGAOptimizer::Driver, 447
drv_index_map
Dakota::NestedModel, 597
dsv_index_map
Dakota::NestedModel, 596
duplication_detect
Dakota::ApplicationInterface, 325

EffGlobalMinimizer, 449
EfficientSubspaceMethod, 451
EmbedHybridMetaIterator, 455
enforce_input_rules
Dakota::FSUDesignCompExp, 476
Dakota::PSUADEDesignCompExp, 793
enroll
Dakota::GetLongOpt, 483
Environment, 457
Dakota::Environment, 459, 460
estimate_derivatives
Dakota::Model, 577
estimate_message_lengths
Dakota::Model, 576
estimate_order
Dakota::RichExtrapVerification, 827
eval_tag_prefix
Dakota::Interface, 499
Dakota::Iterator, 511
Dakota::Model, 574
Dakota::RecastModel, 802
Dakota::SingleModel, 857
eval_x
Dakota::NomadOptimizer::Evaluator, 462
evalInterfaceIds
Dakota::ParamResponsePair, 754
Evaluate
Dakota::JEGAOptimizer::Evaluator, 466, 467
evaluate_parameter_sets
Dakota::Analyzer, 314
evaluation_available
Dakota::COLINApplication, 349
evaluation_cache
Dakota::Model, 574
evaluation_id
Dakota::DataFitSurrModel, 390
Dakota::HierarchSurrModel, 491
Dakota::NestedModel, 596
Evaluator
Dakota::JEGAOptimizer::Evaluator, 464, 465
Dakota::NomadOptimizer::Evaluator, 462
EvaluatorCreator
Dakota::JEGAOptimizer::EvaluatorCreator, 468
ExecutableEnvironment, 469
exit_mode
Dakota::Environment, 460
ExperimentData, 470
export_point
Dakota::DataFitSurrModel, 391
extract_data
Dakota::ResultsDBAny, 819
extract_parameter_sets
Dakota::SeqHybridMetaIterator, 834
ExtractAllData
Dakota::JEGAOptimizer::Driver, 448

FDstep1
Dakota::Model, 577
FDstep2
Dakota::Model, 577
FIELD_NAMES
Dakota, 152
FSUDesignCompExp, 473
Dakota::FSUDesignCompExp, 475
fdGradStepSize
Dakota::Model, 579
fdHessByFnStepSize
Dakota::Model, 579
fdHessByGradStepSize
Dakota::Model, 579
file_cleanup
Dakota::ProcessApplicInterface, 778
file_op_items
Dakota::WorkdirHelper, 923
filtered_interface_list
Dakota::LibraryEnvironment, 530
filtered_model_list
Dakota::LibraryEnvironment, 530
finalize
Dakota::Approximation, 336
Dakota::PecosApproximation, 765
finalize_run

Dakota::Iterator, 510
Dakota::LeastSq, 527
Dakota::Minimizer, 542
Dakota::NonD, 623
Dakota::Optimizer, 730
find_optimum
Dakota::APPSOptimizer, 346
Dakota::COLINOptimizer, 353
Dakota::JEGAOptimizer, 524
flush
Dakota, 146
Dakota::CommandShell, 359
force_rebuild
Dakota::SurrogateModel, 891
ForkApplicInterface, 471
fpinit_ASL
library_mode.cpp, 931
main.cpp, 934
free_iterator
Dakota::IteratorScheduler, 517
functionGradients
Dakota::ResponseRep, 816
functionSurfaces
Dakota::ApproximationInterface, 342

G1
Dakota::CONMINOptimizer, 367
G2
Dakota::CONMINOptimizer, 367
GPmodel_apply
Dakota::GaussProcApproximation, 480
GaussProcApproximation, 476
Dakota::GaussProcApproximation, 480
get_approx
Dakota::Approximation, 336
get_best_sample
Dakota::EffGlobalMinimizer, 451
get_confidence_intervals
Dakota::LeastSq, 528
get_constraints
Dakota::Constraints, 379
get_db
Dakota::ProblemDescDB, 775
get_environment
Dakota::Environment, 461
get_interface
Dakota::Interface, 500
get_iterator
Dakota::Iterator, 512
get_model
Dakota::Model, 577
get_parameter_sets
Dakota::NonDSampling, 711, 712
get_pathext
Dakota, 151
get_shared_data
Dakota::SharedApproxData, 840
GetBestMOSolutions
Dakota::JEGAOptimizer, 523
GetBestSOSolutions
Dakota::JEGAOptimizer, 523
GetBestSolutions
Dakota::JEGAOptimizer, 523
GetDescription
Dakota::JEGAOptimizer::Evaluator, 467
GetLongOpt, 481
Dakota::GetLongOpt, 483
GetName
Dakota::JEGAOptimizer::Evaluator, 467
GetNumberLinearConstraints
Dakota::JEGAOptimizer::Evaluator, 466
GetNumberNonLinearConstraints
Dakota::JEGAOptimizer::Evaluator, 466
getRmax
Dakota, 148
getdist
Dakota, 147
gnewton_set_recast
Dakota::Minimizer, 542
Graphics, 484
GridApplicInterface, 486

hard_convergence_check
Dakota::SurrBasedLocalMinimizer, 881
herbie
Dakota::TestDriverInterface, 902
herbie1D
Dakota::TestDriverInterface, 901
hessian
Dakota::SurfpackApproximation, 874
HierarchSurrModel, 487
hom_constraint_eval
Dakota::SurrBasedLocalMinimizer, 883
hom_objective_eval
Dakota::SurrBasedLocalMinimizer, 883

IC
Dakota::CONMINOptimizer, 368
ISC
Dakota::CONMINOptimizer, 368

id_vars_exact_compare
Dakota, 149
import_points
Dakota::DataFitSurrModel, 391
inactiveVarsCompsTotals
Dakota::SharedVariablesDataRep, 854
increment_grid_preference
Dakota::NonDCubature, 640
increment_order
Dakota::NonDPolynomialChaos, 698
increment_parallel_configuration
Dakota::ParallelLibrary, 750
increment_reference
Dakota::NonDCubature, 640
increment_specification_sequence
Dakota::NonDExpansion, 649
Dakota::NonDPolynomialChaos, 698
init_communicators
Dakota::Model, 575
Dakota::ParallelLibrary, 751
init_communicators_checks
Dakota::ApplicationInterface, 325
Dakota::DirectApplicInterface, 439
Dakota::ProcessHandleApplicInterface, 782
Dakota::SysCallApplicInterface, 894
init_evaluation_concurrency
Dakota::IteratorScheduler, 518
init_iterator
Dakota::IteratorScheduler, 516, 517
init_iterator_parallelism
Dakota::IteratorScheduler, 518
init_mpi_comm
Dakota::ParallelLibrary, 750
init_preferred_env_path
Dakota::WorkdirHelper, 924
init_serial
Dakota::ApplicationInterface, 323
Dakota::Model, 575
init_startup_path
Dakota::WorkdirHelper, 924
initial_points
Dakota::JEGAOptimizer, 524
initial_taylor_series
Dakota::NonDLocalReliability, 688
initialize
Dakota::NonDAdaptImpSampling, 629
Dakota::RecastModel, 802
Dakota::WorkdirHelper, 922
initialize_class_data
Dakota::NonDLocalReliability, 688
initialize_export
Dakota::DataFitSurrModel, 391
initialize_final_statistics
Dakota::NonD, 624
initialize_graphics
Dakota::Iterator, 510
initialize_grid
Dakota::NonDQuadrature, 702
initialize_h
Dakota::Model, 577
initialize_level_data
Dakota::NonDLocalReliability, 688
initialize_mpp_search_data
Dakota::NonDLocalReliability, 689
initialize_random_variable_parameters
Dakota::NonD, 624
initialize_random_variable_types
Dakota::NonD, 624
initialize_random_variables
Dakota::NonD, 623
initialize_run
Dakota::Analyzer, 313
Dakota::Iterator, 509
Dakota::LeastSq, 527
Dakota::Minimizer, 541
Dakota::NonD, 623
Dakota::Optimizer, 730
initialize_scaling
Dakota::Minimizer, 545
initialize_variables_and_constraints
Dakota::APPSOptimizer, 346
insert
Dakota::ResultsDBAny, 819
intCntlParmArray
Dakota::DOTOptimizer, 446
Interface, 491
Dakota::Interface, 498
interface_id
Dakota::Model, 574
isReadyForWork
Dakota::APPSEvalMgr, 344
Iterator, 500
Dakota::Iterator, 507, 508
IteratorScheduler, 514
Dakota::IteratorScheduler, 516

JEGAOptimizer, 519
Dakota::JEGAOptimizer, 521
JEGAOptimizer.cpp, 929

JEGAOptimizer.hpp, 930
JEGAOptimizer::Driver, 447
JEGAOptimizer::Evaluator, 463
JEGAOptimizer::EvaluatorCreator, 468

kw_1
Dakota, 152, 230
kw_10
Dakota, 154, 232
kw_100
Dakota, 175, 249
kw_101
Dakota, 175, 249
kw_102
Dakota, 175, 249
kw_103
Dakota, 175, 249
kw_104
Dakota, 175, 249
kw_105
Dakota, 175, 250
kw_106
Dakota, 176, 250
kw_107
Dakota, 176, 250
kw_108
Dakota, 176, 250
kw_109
Dakota, 176, 250
kw_11
Dakota, 154, 232
kw_110
Dakota, 176, 250
kw_111
Dakota, 177, 251
kw_112
Dakota, 177, 251
kw_113
Dakota, 177, 251
kw_114
Dakota, 177, 251
kw_115
Dakota, 177, 251
kw_116
Dakota, 178, 252
kw_117
Dakota, 178, 252
kw_118
Dakota, 178, 252
kw_119
Dakota, 178, 252
kw_12
Dakota, 154, 232
kw_120
Dakota, 178, 252
kw_121
Dakota, 178, 252
kw_122
Dakota, 179, 253
kw_123
Dakota, 179, 253
kw_124
Dakota, 179, 253
kw_125
Dakota, 179, 253
kw_126
Dakota, 179, 253
kw_127
Dakota, 180, 253
kw_128
Dakota, 180, 254
kw_129
Dakota, 180, 254
kw_13
Dakota, 154, 232
kw_130
Dakota, 180, 254
kw_131
Dakota, 180, 254
kw_132
Dakota, 181, 254
kw_133
Dakota, 181, 255
kw_134
Dakota, 181, 255
kw_135
Dakota, 181, 255
kw_136
Dakota, 182, 255
kw_137
Dakota, 182, 255
kw_138
Dakota, 182, 255
kw_139
Dakota, 182, 256
kw_14
Dakota, 154, 232
kw_140
Dakota, 182, 256

kw_141
Dakota, 182, 256
kw_142
Dakota, 183, 256
kw_143
Dakota, 183, 256
kw_144
Dakota, 183, 256
kw_145
Dakota, 183, 257
kw_146
Dakota, 183, 257
kw_147
Dakota, 183, 257
kw_148
Dakota, 184, 257
kw_149
Dakota, 184, 257
kw_15
Dakota, 155, 233
kw_150
Dakota, 184, 257
kw_151
Dakota, 184, 258
kw_152
Dakota, 184, 258
kw_153
Dakota, 185, 258
kw_154
Dakota, 185, 258
kw_155
Dakota, 185, 258
kw_156
Dakota, 186, 258
kw_157
Dakota, 186, 259
kw_158
Dakota, 186, 259
kw_159
Dakota, 186, 259
kw_16
Dakota, 155, 233
kw_160
Dakota, 186, 259
kw_161
Dakota, 187, 259
kw_162
Dakota, 187, 260
kw_163
Dakota, 187, 260
kw_164
Dakota, 187, 260
kw_165
Dakota, 187, 260
kw_166
Dakota, 187, 260
kw_167
Dakota, 188, 261
kw_168
Dakota, 188, 261
kw_169
Dakota, 188, 261
kw_17
Dakota, 155, 233
kw_170
Dakota, 189, 261
kw_171
Dakota, 189, 261
kw_172
Dakota, 189, 262
kw_173
Dakota, 189, 262
kw_174
Dakota, 190, 262
kw_175
Dakota, 190, 262
kw_176
Dakota, 190, 262
kw_177
Dakota, 190, 262
kw_178
Dakota, 190, 263
kw_179
Dakota, 191, 263
kw_18
Dakota, 156, 233
kw_180
Dakota, 191, 263
kw_181
Dakota, 191, 263
kw_182
Dakota, 191, 263
kw_183
Dakota, 191, 264
kw_184
Dakota, 192, 264
kw_185
Dakota, 192, 264

kw_186
Dakota, 192, 264
kw_187
Dakota, 192, 264
kw_188
Dakota, 192, 265
kw_189
Dakota, 192, 265
kw_19
Dakota, 156, 233
kw_190
Dakota, 193, 265
kw_191
Dakota, 193, 265
kw_192
Dakota, 193, 265
kw_193
Dakota, 193, 265
kw_194
Dakota, 193, 266
kw_195
Dakota, 194, 266
kw_196
Dakota, 194, 266
kw_197
Dakota, 194, 266
kw_198
Dakota, 194, 266
kw_199
Dakota, 194, 266
kw_2
Dakota, 152, 230
kw_20
Dakota, 156, 234
kw_200
Dakota, 195, 267
kw_201
Dakota, 195, 267
kw_202
Dakota, 195, 267
kw_203
Dakota, 195, 267
kw_204
Dakota, 195, 267
kw_205
Dakota, 267
kw_206
Dakota, 196, 268
kw_207
Dakota, 196, 268
kw_208
Dakota, 196, 268
kw_209
Dakota, 196, 268
kw_21
Dakota, 156, 234
kw_210
Dakota, 197, 269
kw_211
Dakota, 197, 269
kw_212
Dakota, 197, 269
kw_213
Dakota, 197, 269
kw_214
Dakota, 197, 269
kw_215
Dakota, 197, 270
kw_216
Dakota, 198, 270
kw_217
Dakota, 198, 270
kw_218
Dakota, 198, 270
kw_219
Dakota, 198, 270
kw_22
Dakota, 156, 234
kw_220
Dakota, 199, 271
kw_221
Dakota, 199, 271
kw_222
Dakota, 200, 271
kw_223
Dakota, 200, 271
kw_224
Dakota, 200, 272
kw_225
Dakota, 200, 272
kw_226
Dakota, 201, 272
kw_227
Dakota, 201, 272
kw_228
Dakota, 201, 272
kw_229
Dakota, 201, 272

kw_23
Dakota, 157, 234
kw_230
Dakota, 202, 273
kw_231
Dakota, 202, 273
kw_232
Dakota, 202, 273
kw_233
Dakota, 202, 273
kw_234
Dakota, 203, 274
kw_235
Dakota, 203, 274
kw_236
Dakota, 203, 274
kw_237
Dakota, 203, 274
kw_238
Dakota, 203, 274
kw_239
Dakota, 204, 274
kw_24
Dakota, 157, 234
kw_240
Dakota, 205, 275
kw_241
Dakota, 205, 275
kw_242
Dakota, 205, 275
kw_243
Dakota, 205, 275
kw_244
Dakota, 205, 275
kw_245
Dakota, 206, 276
kw_246
Dakota, 206, 276
kw_247
Dakota, 206, 276
kw_248
Dakota, 276
kw_249
Dakota, 207, 276
kw_25
Dakota, 157, 234
kw_250
Dakota, 207, 276
kw_251
Dakota, 207, 277
kw_252
Dakota, 207, 277
kw_253
Dakota, 207, 277
kw_254
Dakota, 208, 277
kw_255
Dakota, 208, 277
kw_256
Dakota, 208, 278
kw_257
Dakota, 208, 278
kw_258
Dakota, 208, 278
kw_259
Dakota, 208, 278
kw_26
Dakota, 157, 235
kw_260
Dakota, 209, 278
kw_261
Dakota, 209, 279
kw_262
Dakota, 209, 279
kw_263
Dakota, 209, 279
kw_264
Dakota, 209
kw_265
Dakota, 210, 279
kw_266
Dakota, 210, 279
kw_267
Dakota, 210, 280
kw_268
Dakota, 210, 280
kw_269
Dakota, 210, 280
kw_27
Dakota, 157, 235
kw_270
Dakota, 211, 280
kw_271
Dakota, 211, 280
kw_272
Dakota, 211, 280
kw_273
Dakota, 212, 281

kw_274
Dakota, 212, 281
kw_275
Dakota, 212, 281
kw_276
Dakota, 212, 281
kw_277
Dakota, 212, 281
kw_278
Dakota, 213, 281
kw_279
Dakota, 213, 282
kw_28
Dakota, 157, 235
kw_280
Dakota, 213, 282
kw_281
Dakota, 213, 282
kw_282
Dakota, 214, 282
kw_283
Dakota, 214, 282
kw_284
Dakota, 214, 282
kw_285
Dakota, 215, 283
kw_286
Dakota, 215, 283
kw_287
Dakota, 215, 283
kw_288
Dakota, 215, 283
kw_289
Dakota, 216, 284
kw_29
Dakota, 158, 235
kw_290
Dakota, 216, 284
kw_291
Dakota, 216, 284
kw_292
Dakota, 216, 284
kw_293
Dakota, 216, 284
kw_294
Dakota, 217, 284
kw_295
Dakota, 217, 285
kw_296
Dakota, 217, 285
kw_297
Dakota, 218, 285
kw_298
Dakota, 218, 285
kw_299
Dakota, 218, 285
kw_3
Dakota, 152, 230
kw_30
Dakota, 158, 235
kw_300
Dakota, 218, 286
kw_301
Dakota, 218, 286
kw_302
Dakota, 219, 286
kw_303
Dakota, 219, 287
kw_304
Dakota, 220, 287
kw_305
Dakota, 220, 287
kw_306
Dakota, 220, 287
kw_307
Dakota, 220, 287
kw_308
Dakota, 221, 287
kw_309
Dakota, 221, 288
kw_31
Dakota, 158, 235
kw_310
Dakota, 221, 288
kw_311
Dakota, 221, 288
kw_312
Dakota, 222, 288
kw_313
Dakota, 222, 289
kw_314
Dakota, 222, 289
kw_315
Dakota, 222, 289
kw_316
Dakota, 223, 289
kw_317
Dakota, 223, 290

kw_318
Dakota, 223, 290
kw_319
Dakota, 223, 290
kw_32
Dakota, 158, 236
kw_320
Dakota, 224, 290
kw_321
Dakota, 224, 291
kw_322
Dakota, 224, 291
kw_323
Dakota, 224, 291
kw_324
Dakota, 225, 291
kw_325
Dakota, 225, 292
kw_326
Dakota, 225, 292
kw_327
Dakota, 225, 292
kw_328
Dakota, 226, 292
kw_329
Dakota, 226, 293
kw_33
Dakota, 158, 236
kw_330
Dakota, 226, 293
kw_331
Dakota, 226, 293
kw_332
Dakota, 227, 293
kw_333
Dakota, 227, 294
kw_334
Dakota, 227, 294
kw_335
Dakota, 227, 294
kw_336
Dakota, 227, 294
kw_337
Dakota, 228, 294
kw_338
Dakota, 228, 295
kw_339
Dakota, 228, 295
kw_34
Dakota, 159, 236
kw_340
Dakota, 229, 295
kw_341
Dakota, 229, 295
kw_342
Dakota, 229, 295
kw_343
Dakota, 229, 296
kw_344
Dakota, 296
kw_345
Dakota, 230, 296
kw_346
Dakota, 296
kw_347
Dakota, 297
kw_348
Dakota, 297
kw_349
Dakota, 297
kw_35
Dakota, 159, 236
kw_350
Dakota, 297
kw_351
Dakota, 297
kw_352
Dakota, 298
kw_353
Dakota, 298
kw_354
Dakota, 298
kw_355
Dakota, 298
kw_356
Dakota, 298
kw_357
Dakota, 299
kw_358
Dakota, 299
kw_359
Dakota, 299
kw_36
Dakota, 159, 236
kw_360
Dakota, 299
kw_362
Dakota, 300

kw_37
Dakota, 160, 237
kw_38
Dakota, 160, 237
kw_39
Dakota, 160, 237
kw_4
Dakota, 152, 231
kw_40
Dakota, 161, 237
kw_41
Dakota, 161, 238
kw_42
Dakota, 161, 238
kw_43
Dakota, 161, 238
kw_44
Dakota, 161, 238
kw_45
Dakota, 161, 238
kw_46
Dakota, 162, 239
kw_47
Dakota, 162, 239
kw_48
Dakota, 162, 239
kw_49
Dakota, 162, 239
kw_5
Dakota, 153, 231
kw_50
Dakota, 162, 239
kw_51
Dakota, 162, 239
kw_52
Dakota, 163, 240
kw_53
Dakota, 163, 240
kw_54
Dakota, 163, 240
kw_55
Dakota, 163, 240
kw_56
Dakota, 163, 240
kw_57
Dakota, 163, 240
kw_58
Dakota, 164, 241
kw_59
Dakota, 164, 241
kw_6
Dakota, 153, 231
kw_60
Dakota, 164, 241
kw_61
Dakota, 164, 241
kw_62
Dakota, 165, 241
kw_63
Dakota, 165, 241
kw_64
Dakota, 165, 242
kw_65
Dakota, 166, 242
kw_66
Dakota, 166, 242
kw_67
Dakota, 166, 242
kw_68
Dakota, 166, 242
kw_69
Dakota, 166, 243
kw_7
Dakota, 153, 231
kw_70
Dakota, 167, 243
kw_71
Dakota, 167, 243
kw_72
Dakota, 167, 243
kw_73
Dakota, 168, 243
kw_74
Dakota, 168, 243
kw_75
Dakota, 168, 244
kw_76
Dakota, 168, 244
kw_77
Dakota, 169, 244
kw_78
Dakota, 169, 244
kw_79
Dakota, 170, 244
kw_8
Dakota, 153, 231
kw_80
Dakota, 170, 245

kw_81
Dakota, 170, 245
kw_82
Dakota, 171, 245
kw_83
Dakota, 171, 245
kw_84
Dakota, 171, 245
kw_85
Dakota, 171, 246
kw_86
Dakota, 172, 246
kw_87
Dakota, 172, 246
kw_88
Dakota, 172, 246
kw_89
Dakota, 172, 246
kw_9
Dakota, 153, 232
kw_90
Dakota, 173, 247
kw_91
Dakota, 173, 247
kw_92
Dakota, 173, 247
kw_93
Dakota, 173, 247
kw_94
Dakota, 173, 247
kw_95
Dakota, 174, 248
kw_96
Dakota, 174, 248
kw_97
Dakota, 174, 248
kw_98
Dakota, 174, 248
kw_99
Dakota, 174, 248

lagrangian_merit
Dakota::SurrBasedMinimizer, 887
LeastSq, 525
Dakota::LeastSq, 527
library_mode.cpp, 930
callback_function, 932
fpinit_ASL, 931
main, 932
parallel_input, 933
parallel_interface_plugin, 932
run_dakota_data, 931
run_dakota_mixed, 931
run_dakota_parse, 931
serial_input, 933
serial_interface_plugin, 932
library_split.cpp, 934
LibraryEnvironment, 529
Dakota::LibraryEnvironment, 530
LightWtBaseConstructor, 531
likelihoodScale
Dakota::NonDDREAMBayesCalibration, 644
lin_coeffs_modify_n2s
Dakota::Minimizer, 546
link
Dakota::WorkdirHelper, 923
link_items
Dakota::WorkdirHelper, 922
load
Dakota::Response, 810
Dakota::ResponseRep, 816
load_distribute_points
Dakota::ParamStudy, 759
load_parameters
Dakota::NomadOptimizer, 615
load_scalar
Dakota::ExperimentData, 470
LoadAlgorithmConfig
Dakota::JEGAOptimizer, 522
LoadDakotaResponses
Dakota::JEGAOptimizer, 521
LoadProblemConfig
Dakota::JEGAOptimizer, 522
LoadTheConstraints
Dakota::JEGAOptimizer, 523
LoadTheDesignVariables
Dakota::JEGAOptimizer, 522
LoadTheObjectiveFunctions
Dakota::JEGAOptimizer, 522
LoadTheParameterDatabase
Dakota::JEGAOptimizer, 522
local_eval_concurrency
Dakota::Model, 574
Dakota::NestedModel, 595
local_eval_synchronization
Dakota::Model, 573
Dakota::NestedModel, 595
local_objective_recast_retrieve
Dakota::Optimizer, 732

lookup_by_val
Dakota, 149

MPIManager, 580
MPIPackBuffer, 581
MPIUnpackBuffer, 583
MS1
Dakota::CONMINOptimizer, 368
main
library_mode.cpp, 932
main.cpp, 934
restart_util.cpp, 936
main.cpp, 934
fpinit_ASL, 934
main, 934
make_variable_defaults
Dakota::NIDRProblemDescDB, 603
manage_asv
Dakota::Model, 579
manage_data_recastings
Dakota::DataFitSurrModel, 391
manage_linear_constraints
Dakota::Constraints, 378
MandatoryValue
Dakota::GetLongOpt, 482
map
Dakota::ApplicationInterface, 323
map_domain
Dakota::COLINApplication, 350
map_xi_to_x
Dakota::EfficientSubspaceMethod, 455
master_dynamic_schedule_analyses
Dakota::ApplicationInterface, 325
master_dynamic_schedule_evaluations
Dakota::ApplicationInterface, 326
master_dynamic_schedule_evaluations_nowait
Dakota::ApplicationInterface, 328
master_dynamic_schedule_iterators
Dakota::IteratorScheduler, 519
MatchesWC, 531
matlab_engine_run
Dakota::MatlabInterface, 533
MatlabInterface, 532
maxEvalConcurrency
Dakota::Iterator, 513
mc_api_run
Dakota::TestDriverInterface, 902
MetaIterator, 533
mi_parallel_level
Dakota::ParallelConfiguration, 738
mi_parallel_level_iterator
Dakota::ParallelConfiguration, 738
mindist
Dakota, 148
mindistindx
Dakota, 148
minimize_residuals
Dakota::NL2SOLLeastSq, 606
minimize_surrogates
Dakota::SurrBasedLocalMinimizer, 880
Minimizer, 535
Dakota::Minimizer, 541
MixedVarConstraints, 547
Dakota::MixedVarConstraints, 547
MixedVariables, 548
Dakota::MixedVariables, 549
mode_counts
Dakota::NonDSampling, 713
Model, 549
Dakota::Model, 571
modify_n2s
Dakota::Minimizer, 546
modify_s2n
Dakota::Minimizer, 543
mpi_debug_hold
Dakota, 146

N1
Dakota::CONMINOptimizer, 366
N2
Dakota::CONMINOptimizer, 366
N3
Dakota::CONMINOptimizer, 366
N4
Dakota::CONMINOptimizer, 367
N5
Dakota::CONMINOptimizer, 367
NCSUOptimizer, 585
Dakota::NCSUOptimizer, 587
NIDRProblemDescDB, 599
NL2Res, 604
NL2SOLLeastSq, 604
NLPQLPOptimizer, 607
NLSSOLLeastSq, 611
Dakota::NLSSOLLeastSq, 612
NPSOLOptimizer, 722
Dakota::NPSOLOptimizer, 724
NUMBER_OF_FIELDS
Dakota, 152
Name

Dakota::JEGAOptimizer::Evaluator, 465
need_resp_trans_byvars
Dakota::Minimizer, 543
NestedModel, 588
new_dataset
Dakota::Graphics, 485
nlf0_evaluator
Dakota::SNLLOptimizer, 867
nlf1_evaluator
Dakota::SNLLOptimizer, 867
nlf2_evaluator
Dakota::SNLLOptimizer, 868
nlf2_evaluator_gn
Dakota::SNLLLeastSq, 861
NoDBBaseConstructor, 613
NomadOptimizer, 613
Dakota::NomadOptimizer, 615
NomadOptimizer::Evaluator, 461
NonD, 616
NonDAdaptImpSampling, 626
Dakota::NonDAdaptImpSampling, 628
NonDAdaptiveSampling, 629
Dakota::NonDAdaptiveSampling, 634
NonDBayesCalibration, 634
Dakota::NonDBayesCalibration, 636
NonDCalibration, 636
Dakota::NonDCalibration, 638
NonDCubature, 638
Dakota::NonDCubature, 640
NonDDREAMBayesCalibration, 641
Dakota::NonDDREAMBayesCalibration, 643
NonDExpansion, 644
NonDGPImpSampling, 659
Dakota::NonDGPImpSampling, 661
NonDGPMSABayesCalibration, 662
Dakota::NonDGPMSABayesCalibration, 664
NonDGlobalEvidence, 651
NonDGlobalInterval, 652
NonDGlobalReliability, 655
NonDGlobalSingleInterval, 658
NonDIncremLHSSampling, 665
Dakota::NonDIncremLHSSampling, 666
NonDIntegration, 667
Dakota::NonDIntegration, 669
NonDInterval, 670
NonDLHSEvidence, 672
NonDLHSInterval, 673
NonDLHSSampling, 674
Dakota::NonDLHSSampling, 675, 676
NonDLHSSingleInterval, 676
NonDLocalEvidence, 678
NonDLocalInterval, 679
NonDLocalReliability, 681
NonDLocalSingleInterval, 691
NonDPOFDarts, 692
Dakota::NonDPOFDarts, 695
NonDPolynomialChaos, 695
Dakota::NonDPolynomialChaos, 698
NonDQUESOBayesCalibration, 703
Dakota::NonDQUESOBayesCalibration, 704
NonDQuadrature, 699
Dakota::NonDQuadrature, 701, 702
NonDReliability, 705
NonDSampling, 707
Dakota::NonDSampling, 711
NonDSparseGrid, 714
Dakota::NonDSparseGrid, 716
NonDStochCollocation, 717
Dakota::NonDStochCollocation, 718
NonlinearCGOptimizer, 719
num_samples
Dakota::Analyzer, 313
Dakota::DDACEDesignCompExp, 435
Dakota::FSUDesignCompExp, 475
Dakota::NonDCubature, 640
Dakota::NonDQuadrature, 702
Dakota::NonDSampling, 711
Dakota::NonDSparseGrid, 716
Dakota::PSUADEDesignCompExp, 793

objective
Dakota::Minimizer, 543, 544
objective_eval
Dakota::NCSUOptimizer, 588
objective_gradient
Dakota::Minimizer, 544
objective_hessian
Dakota::Minimizer, 544
objective_reduction
Dakota::Optimizer, 731
operator<<
Dakota::CommandShell, 359
operator=
Dakota::Approximation, 335
Dakota::Constraints, 377
Dakota::Environment, 460
Dakota::Interface, 499
Dakota::Iterator, 509
Dakota::Model, 572

Dakota::ProblemDescDB, 774
Dakota::SharedApproxData, 839
operator==
Dakota, 147, 149
OptDartsOptimizer, 724
OptType
Dakota::GetLongOpt, 482
Optimizer, 727
OptionalValue
Dakota::GetLongOpt, 482
output_helper
Dakota::CommandLineHandler, 357
OutputManager, 732
Dakota::OutputManager, 734
OutputWriter, 735

PMA2_constraint_eval
Dakota::NonDLocalReliability, 687
PMA_constraint_eval
Dakota::NonDLocalReliability, 687
PMA_objective_eval
Dakota::NonDLocalReliability, 687
PRPMultiIndexCache
Dakota, 145
PRPMultiIndexQueue
Dakota, 145
PSUADEDesignCompExp, 790
Dakota::PSUADEDesignCompExp, 792
PStudyDACE, 788
parallel_input
library_mode.cpp, 933
parallel_interface_plugin
library_mode.cpp, 932
ParallelConfiguration, 736
ParallelDirectApplicInterface, 738
ParallelLevel, 739
ParallelLibrary, 742
Dakota::ParallelLibrary, 749
ParamResponsePair, 751
Dakota::ParamResponsePair, 754
ParamStudy, 755
parse
Dakota::Environment, 461
Dakota::GetLongOpt, 483
parse_inputs
Dakota::ProblemDescDB, 774
partial_prp_equality, 760
partial_prp_hash, 760
PecosApproximation, 761
peer_dynamic_schedule_evaluations
Dakota::ApplicationInterface, 327
peer_dynamic_schedule_evaluations_nowait
Dakota::ApplicationInterface, 328
peer_static_schedule_evaluations
Dakota::ApplicationInterface, 326
penalty_merit
Dakota::SurrBasedMinimizer, 887
perform_analysis
Dakota, 148
perform_evaluation_impl
Dakota::COLINApplication, 350
PerformIterations
Dakota::JEGAOptimizer::Driver, 448
po_which
Dakota::WorkdirHelper, 924
pop
Dakota::Approximation, 335
Dakota::PecosApproximation, 764
pop_approximation
Dakota::ApproximationInterface, 341
pop_output_tag
Dakota::OutputManager, 735
post_process
Dakota::ProblemDescDB, 775
post_run
Dakota::Analyzer, 313
Dakota::COLINOptimizer, 354
Dakota::DDACEDesignCompExp, 435
Dakota::FSUDesignCompExp, 475
Dakota::Iterator, 509
Dakota::LeastSq, 527
Dakota::MetaIterator, 534
Dakota::Minimizer, 541
Dakota::Optimizer, 730
Dakota::ParamStudy, 759
Dakota::PSUADEDesignCompExp, 793
Dakota::SNLLLeastSq, 861
pre_output
Dakota::Analyzer, 314
pre_run
Dakota::ConcurrentMetaIterator, 361
Dakota::DDACEDesignCompExp, 435
Dakota::FSUDesignCompExp, 475
Dakota::Iterator, 509
Dakota::ParamStudy, 758
Dakota::PSUADEDesignCompExp, 792
prepare_process_environment
Dakota::ProcessApplicInterface, 779
prepend_path_item

Dakota::WorkdirHelper, 923
prepend_preferred_env_path
Dakota::WorkdirHelper, 922
primary_resp_differencer
Dakota::Minimizer, 545
primary_resp_reducer
Dakota::Optimizer, 731
primary_resp_weighter
Dakota::LeastSq, 528
print_distribution_map
Dakota::NonD, 625
print_distribution_mappings
Dakota::NonD, 624
print_restart
Dakota, 150
print_restart_pdb
Dakota, 150
print_restart_tabular
Dakota, 150
print_results
Dakota::Analyzer, 314
Dakota::ConcurrentMetaIterator, 362
Dakota::Iterator, 510
Dakota::LeastSq, 528
Dakota::Optimizer, 731
Dakota::PStudyDACE, 790
Dakota::RichExtrapVerification, 826
Dakota::SeqHybridMetaIterator, 833
Dakota::SurrBasedMinimizer, 886
Dakota::Verification, 917
print_sobol_indices
Dakota::Analyzer, 315
printControl
Dakota::CONMINOptimizer, 365
Dakota::DOTOptimizer, 446
prior_density
Dakota::NonDDREAMBayesCalibration, 643
prior_sample
Dakota::NonDDREAMBayesCalibration, 643
probDescDB
Dakota::Iterator, 512
Dakota::Model, 579
probability
Dakota::NonDLocalReliability, 690
problem_size
Dakota::NonDDREAMBayesCalibration, 643
problem_value
Dakota::NonDDREAMBayesCalibration, 643
ProblemDescDB, 765
Dakota::ProblemDescDB, 773
ProcessApplicInterface, 775
ProcessHandleApplicInterface, 779
Dakota::ProcessHandleApplicInterface, 781
ProgramOptions, 784
push_output_tag
Dakota::ParallelLibrary, 750
python_convert_int
Dakota::PythonInterface, 795
PythonInterface, 793

quantify_uncertainty
Dakota::NonDDREAMBayesCalibration, 644
Dakota::NonDGPImpSampling, 661
Dakota::NonDGPMSABayesCalibration, 664
Dakota::NonDIncremLHSSampling, 666
Dakota::NonDLHSSampling, 676
Dakota::NonDQUESOBayesCalibration, 705

RIA_constraint_eval
Dakota::NonDLocalReliability, 686
RIA_objective_eval
Dakota::NonDLocalReliability, 686
randomSeed
Dakota::NonDGPMSABayesCalibration, 665
rawResponseMap
Dakota::Interface, 500
read
Dakota::ParamResponsePair, 754
Dakota::ResponseRep, 814, 815
read_annotated
Dakota::ResponseRep, 814
read_challenge_points
Dakota::ApproximationInterface, 341
read_neutral
Dakota, 150
read_tabular
Dakota::MixedVariables, 549
Dakota::RelaxedVariables, 806
Dakota::ResponseRep, 814
read_variables_responses
Dakota::Analyzer, 315
realCntlParmArray
Dakota::DOTOptimizer, 446
rebuild
Dakota::Approximation, 335
Dakota::PecosApproximation, 764
rebuild_approximation
Dakota::ApproximationInterface, 341
RecastModel, 795

Dakota::RecastModel, 801
RecordResponses
Dakota::JEGAOptimizer::Evaluator, 466
recursive_copy
Dakota::WorkdirHelper, 923
recv
Dakota::APPSEvalMgr, 344
reduce_model
Dakota::Optimizer, 731
reduced_space_uq
Dakota::EfficientSubspaceMethod, 454
register_signal_handlers
Dakota, 146
RelaxedVarConstraints, 804
Dakota::RelaxedVarConstraints, 804
RelaxedVariables, 805
Dakota::RelaxedVariables, 806
repair_restart
Dakota, 150
requestVector
Dakota::ActiveSet, 309
reset
Dakota::ResponseRep, 815
reset_inactive
Dakota::ResponseRep, 816
reset_process_environment
Dakota::ProcessApplicInterface, 779
reshape
Dakota::Constraints, 378
Dakota::ResponseRep, 815
resize_best_resp_array
Dakota::Minimizer, 545
resize_best_vars_array
Dakota::Minimizer, 544
resolve_inputs
Dakota::ParallelLibrary, 751
resolve_samples_symbols
Dakota::DDACEDesignCompExp, 435
Response, 806
response_mapping
Dakota::Interface, 499
Dakota::NestedModel, 598
response_modify_n2s
Dakota::Minimizer, 546
response_modify_s2n
Dakota::Minimizer, 543
responseMode
Dakota::SurrogateModel, 891
ResponseRep, 810
Dakota::ResponseRep, 814
restart_util.cpp, 935
main, 936
RestartWriter, 817
restore
Dakota::Approximation, 335
Dakota::PecosApproximation, 765
restore_approximation
Dakota::ApproximationInterface, 341
ResultsDBAny, 818
ResultsEntry
Dakota::ResultsEntry, 821
ResultsEntry< StoredType >, 820
ResultsID, 821
ResultsManager, 822
ResultsNames, 824
retrieve
Dakota::GetLongOpt, 483
returns_multiple_points
Dakota::COLINOptimizer, 353
Dakota::JEGAOptimizer, 524
RichExtrapVerification, 825
run
Dakota::Iterator, 511
run_dakota_data
library_mode.cpp, 931
run_dakota_mixed
library_mode.cpp, 931
run_dakota_parse
library_mode.cpp, 931
run_iterator
Dakota::IteratorScheduler, 517
run_sequential
Dakota::SeqHybridMetaIterator, 833
run_sequential_adaptive
Dakota::SeqHybridMetaIterator, 833

S
Dakota::CONMINOptimizer, 367
SUBMETHOD_COLLABORATIVE
Dakota, 146
SCAL
Dakota::CONMINOptimizer, 368
SCI_FIELD_NAMES
Dakota, 305
SIM, 306
SIM::ParallelDirectApplicInterface
test_local_evaluations, 739
SIM::SerialDirectApplicInterface
test_local_evaluations, 835

SNLLBase, 857
SNLLLeastSq, 859
SNLLOptimizer, 863
    Dakota::SNLLOptimizer, 866, 867
SOLBase, 869
sample_likelihood
    Dakota::NonDDREAMBayesCalibration, 644
sample_to_variables
    Dakota::Analyzer, 313
sampling_reset
    Dakota::NonDCubature, 640
    Dakota::NonDQuadrature, 702
    Dakota::NonDSampling, 711
    Dakota::NonDSparseGrid, 716
save
    Dakota::Response, 810
    Dakota::ResponseRep, 816
scale_model
    Dakota::Minimizer, 542
schedule_iterators
    Dakota::IteratorScheduler, 518
ScilabInterface, 827
secondary_resp_copier
    Dakota::Minimizer, 543
secondary_resp_scaler
    Dakota::Minimizer, 546
send_data_using_get
    Dakota::TrackerHTTP, 904
send_data_using_post
    Dakota::TrackerHTTP, 904
SensAnalysisGlobal, 828
separable_combine
    Dakota::TestDriverInterface, 902
SeparateVariables
    Dakota::JEGAOptimizer::Evaluator, 465
SeqHybridMetaIterator, 831
serial_input
    library_mode.cpp, 933
serial_interface_plugin
    library_mode.cpp, 932
SerialDirectApplicInterface, 834
serve_analyses_asynch
    Dakota::ProcessHandleApplicInterface, 783
serve_analyses_synch
    Dakota::ApplicationInterface, 325
serve_evaluations
    Dakota::ApplicationInterface, 324
serve_evaluations_asynch
    Dakota::ApplicationInterface, 329
serve_evaluations_asynch_peer
    Dakota::ApplicationInterface, 329
serve_evaluations_synch
    Dakota::ApplicationInterface, 329
serve_evaluations_synch_peer
    Dakota::ApplicationInterface, 329
serve_iterators
    Dakota::IteratorScheduler, 519
set_apps_parameters
    Dakota::APPSOptimizer, 346
set_communicators_checks
    Dakota::ApplicationInterface, 325
    Dakota::DirectApplicInterface, 439
    Dakota::ProcessHandleApplicInterface, 782
    Dakota::SysCallApplicInterface, 894
set_compare
    Dakota, 149
set_iterator
    Dakota::IteratorScheduler, 517
set_preferred_path
    Dakota::WorkdirHelper, 924
set_problem
    Dakota::COLINApplication, 349
set_rng
    Dakota::COLINOptimizer, 354
set_solver_parameters
    Dakota::COLINOptimizer, 354
set_u_to_x_mapping
    Dakota::NonD, 625
shape
    Dakota::Constraints, 378
SharedApproxData, 835
    Dakota::SharedApproxData, 838, 839
SharedPecosApproxData, 840
SharedSurfpackApproxData, 843
    Dakota::SharedSurfpackApproxData, 844
SharedVariablesData, 845
SharedVariablesDataRep, 850
    Dakota::SharedVariablesDataRep, 854
shubert1D
    Dakota::TestDriverInterface, 902
SingleModel, 855
smooth_herbie
    Dakota::TestDriverInterface, 902
smooth_herbie1D
    Dakota::TestDriverInterface, 902
solver_setup
    Dakota::COLINOptimizer, 353
spawn_analysis_to_shell

Dakota::SysCallApplicInterface, 894
spawn_evaluation_impl
Dakota::COLINApplication, 349
spawn_evaluation_to_shell
Dakota::SysCallApplicInterface, 894
spawn_input_filter_to_shell
Dakota::SysCallApplicInterface, 894
spawn_output_filter_to_shell
Dakota::SysCallApplicInterface, 895
SpawnApplicInterface, 871
split_filenames
Dakota::ProgramOptions, 788
split_wildcard
Dakota::WorkdirHelper, 922
start_dakota_heartbeat
Dakota, 148
start_grid_computing
Dakota, 148
stop_evaluation_servers
Dakota::ApplicationInterface, 324
stop_grid_computing
Dakota, 148
subModel
Dakota::NestedModel, 598
submit
Dakota::APPSEvalMgr, 344
subordinate_iterator
Dakota::Model, 572
subordinate_model
Dakota::Model, 572
subordinate_models
Dakota::Model, 575
SurfpackApproximation, 872
Dakota::SurfpackApproximation, 874
SurrBasedGlobalMinimizer, 875
SurrBasedLocalMinimizer, 876
SurrBasedMinimizer, 883
surrogate_model
Dakota::Model, 572
SurrogateModel, 888
surrogates_to_surf_data
Dakota::SurfpackApproximation, 875
synch
Dakota::ApplicationInterface, 323
synch_nowait
Dakota::ApplicationInterface, 324
synchronize_derivatives
Dakota::Model, 578
synchronous_local_analyses
Dakota::ProcessApplicInterface, 779
synchronous_local_analysis
Dakota::DirectApplicInterface, 439
Dakota::GridApplicInterface, 487
Dakota::ProcessHandleApplicInterface, 782
Dakota::SysCallApplicInterface, 893
synchronous_local_evaluations
Dakota::ApplicationInterface, 327
SysCallApplicInterface, 892

TANA3Approximation, 895
TaylorApproximation, 897
terminate_modelcenter
Dakota::ParallelLibrary, 750
test_local_evaluations
Dakota::SysCallApplicInterface, 893
SIM::ParallelDirectApplicInterface, 739
SIM::SerialDirectApplicInterface, 835
TestDriverInterface, 898
ToDoubleMatrix
Dakota::JEGAOptimizer, 523
tokenize_env_path
Dakota::WorkdirHelper, 924
tr_ratio_check
Dakota::SurrBasedLocalMinimizer, 881
TrackerHTTP, 903
trendOrder
Dakota::GaussProcApproximation, 481
truth_model
Dakota::Model, 573

uncertain_vars_to_subspace
Dakota::EfficientSubspaceMethod, 454
update
Dakota::ResponseRep, 815
update_actual_model
Dakota::DataFitSurrModel, 392
update_approximation
Dakota::ApproximationInterface, 339, 340
Dakota::DataFitSurrModel, 389
update_augmented_lagrange_multipliers
Dakota::SurrBasedMinimizer, 886
update_filter
Dakota::SurrBasedMinimizer, 887
update_from_actual_model
Dakota::DataFitSurrModel, 392
update_from_sub_model
Dakota::RecastModel, 803
update_from_subordinate_model
Dakota::Model, 573

update_lagrange_multipliers
Dakota::SurrBasedMinimizer, 886
update_level_data
Dakota::NonDLocalReliability, 689
update_mpp_search_data
Dakota::NonDLocalReliability, 689
update_partial
Dakota::ResponseRep, 815
update_penalty
Dakota::SurrBasedLocalMinimizer, 882
update_quasi_hessians
Dakota::Model, 578
update_response
Dakota::Model, 578
usage
Dakota::GetLongOpt, 484
UsageTracker, 905
Dakota::UsageTracker, 906
useDerivs
Dakota::NonDExpansion, 650
uses_method
Dakota::Iterator, 510

VLUncertainInt
Dakota, 302
VLUncertainReal
Dakota, 302
VLUncertainStr
Dakota, 303
VLint, 918
VLreal, 918
VLstr, 919
Valueless
Dakota::GetLongOpt, 482
Var_icheck, 906
var_mp_cbound
Dakota, 305
var_mp_check_cau
Dakota, 303
var_mp_check_ceu
Dakota, 304
var_mp_check_cv
Dakota, 303
var_mp_check_daui
Dakota, 304
var_mp_check_daur
Dakota, 304
var_mp_check_daus
Dakota, 304
var_mp_check_deui
Dakota, 304
var_mp_check_deur
Dakota, 305
var_mp_check_deus
Dakota, 304
var_mp_check_dset
Dakota, 303
var_mp_drange
Dakota, 305
Var_rcheck, 906
Variables, 907
variables_scaler
Dakota::Minimizer, 545
variables_to_sample
Dakota::Analyzer, 315
Dakota::NonDSampling, 712
variablesCompsTotals
Dakota::SharedVariablesDataRep, 854
variance_based_decomp
Dakota::Analyzer, 315
vars_u_to_x_mapping
Dakota::NonD, 625
vars_x_to_u_mapping
Dakota::NonD, 625
Verification, 916
view_aleatory_uncertain_counts
Dakota::NonDSampling, 712
view_design_counts
Dakota::NonDSampling, 712
view_epistemic_uncertain_counts
Dakota::NonDSampling, 713
view_uncertain_counts
Dakota::NonDSampling, 713
volumetric_quality
Dakota::PStudyDACE, 790

wait_local_evaluations
Dakota::SysCallApplicInterface, 893
weight_model
Dakota::LeastSq, 528
which
Dakota::WorkdirHelper, 922
WorkdirHelper, 919
write
Dakota::ParamResponsePair, 754
write_annotated
Dakota::ResponseRep, 814
write_ordered
Dakota, 147
write_tabular

Dakota::ResponseRep, 815