
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 ................................. 325

13 Class Documentation 327
  13.1 ActiveSet Class Reference ................................. 327
  13.2 Analyzer Class Reference .................................. 329
  13.3 ApplicationInterface Class Reference .................... 336
  13.4 Approximation Class Reference ............................ 350
  13.5 ApproximationInterface Class Reference .................. 357
  13.6 APPSEvalMgr Class Reference .............................. 363
  13.7 APPSOptimizer Class Reference ............................ 365
  13.8 BaseConstructor Struct Reference ......................... 368
  13.9 callback_data Struct Reference ............................. 368
  13.10 COLINApplication Class Reference ...................... 369
  13.11 COLINOptimizer Class Reference ......................... 372
  13.12 CollabHybridMetaIterator Class Reference .............. 376
  13.13 CommandLineHandler Class Reference ................... 377
  13.14 CommandShell Class Reference ............................ 379
  13.15 ConcurrentMetaIterator Class Reference ................. 380
  13.16 CONMINOptimizer Class Reference ....................... 383
  13.17 ConsoleRedirector Class Reference ....................... 390
  13.18 Constraints Class Reference .............................. 391
  13.19 DataEnvironment Class Reference ....................... 400
  13.20 DataEnvironmentRep Class Reference ..................... 401
  13.21 DataFitSurrModel Class Reference ....................... 403
  13.22 DataInterface Class Reference ............................ 415
  13.23 DataMethod Class Reference ............................... 416
  13.24 DataMethodRep Class Reference ........................... 417
  13.25 DataModel Class Reference ............................... 430
  13.26 DataModelRep Class Reference ........................... 431
  13.27 DataResponses Class Reference ........................... 436
  13.28 DataResponsesRep Class Reference ....................... 437
<table>
<thead>
<tr>
<th>Section</th>
<th>Class Reference</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.29</td>
<td>DataVariables</td>
<td>441</td>
</tr>
<tr>
<td>13.30</td>
<td>DataVariablesRep</td>
<td>442</td>
</tr>
<tr>
<td>13.31</td>
<td>DDACEDesignExp</td>
<td>456</td>
</tr>
<tr>
<td>13.32</td>
<td>DirectApplicInterface</td>
<td>459</td>
</tr>
<tr>
<td>13.33</td>
<td>DiscrepancyCorrection</td>
<td>463</td>
</tr>
<tr>
<td>13.34</td>
<td>DOTOptimizer</td>
<td>467</td>
</tr>
<tr>
<td>13.35</td>
<td>JEGAOptimizer::Driver</td>
<td>470</td>
</tr>
<tr>
<td>13.36</td>
<td>EffGlobalMinimizer</td>
<td>472</td>
</tr>
<tr>
<td>13.37</td>
<td>EfficientSubspaceMethod</td>
<td>474</td>
</tr>
<tr>
<td>13.38</td>
<td>EmbedHybridMetaIterator</td>
<td>478</td>
</tr>
<tr>
<td>13.39</td>
<td>Environment</td>
<td>480</td>
</tr>
<tr>
<td>13.40</td>
<td>NomadOptimizer::Evaluator</td>
<td>484</td>
</tr>
<tr>
<td>13.41</td>
<td>JEGAOptimizer::Evaluator</td>
<td>486</td>
</tr>
<tr>
<td>13.42</td>
<td>JEGAOptimizer::EvaluatorCreator</td>
<td>491</td>
</tr>
<tr>
<td>13.43</td>
<td>ExecutableEnvironment</td>
<td>492</td>
</tr>
<tr>
<td>13.44</td>
<td>ExperimentData</td>
<td>493</td>
</tr>
<tr>
<td>13.45</td>
<td>ExperimentResponse</td>
<td>498</td>
</tr>
<tr>
<td>13.46</td>
<td>FileReadException</td>
<td>499</td>
</tr>
<tr>
<td>13.47</td>
<td>ForkApplicInterface</td>
<td>500</td>
</tr>
<tr>
<td>13.48</td>
<td>FSUDesignExp</td>
<td>502</td>
</tr>
<tr>
<td>13.49</td>
<td>FunctionEvalFailure</td>
<td>505</td>
</tr>
<tr>
<td>13.50</td>
<td>GaussProcApproximation</td>
<td>505</td>
</tr>
<tr>
<td>13.51</td>
<td>GetLongOpt</td>
<td>510</td>
</tr>
<tr>
<td>13.52</td>
<td>Graphics</td>
<td>513</td>
</tr>
<tr>
<td>13.53</td>
<td>GridApplicInterface</td>
<td>515</td>
</tr>
<tr>
<td>13.54</td>
<td>HierarchSurrModel</td>
<td>516</td>
</tr>
<tr>
<td>13.55</td>
<td>Interface</td>
<td>521</td>
</tr>
<tr>
<td>13.56</td>
<td>Iterator</td>
<td>529</td>
</tr>
<tr>
<td>13.57</td>
<td>IteratorScheduler</td>
<td>542</td>
</tr>
<tr>
<td>13.58</td>
<td>JEGAOptimizer</td>
<td>548</td>
</tr>
<tr>
<td>13.59</td>
<td>LeastSq</td>
<td>554</td>
</tr>
<tr>
<td>13.60</td>
<td>LibraryEnvironment</td>
<td>558</td>
</tr>
<tr>
<td>13.61</td>
<td>LightWtBaseConstructor</td>
<td>560</td>
</tr>
<tr>
<td>13.62</td>
<td>MatchesWC</td>
<td>561</td>
</tr>
<tr>
<td>13.63</td>
<td>MatlabInterface</td>
<td>561</td>
</tr>
<tr>
<td>13.64</td>
<td>MetaIterator</td>
<td>563</td>
</tr>
<tr>
<td>13.65</td>
<td>Minimizer</td>
<td>564</td>
</tr>
<tr>
<td>13.66</td>
<td>MixedVarConstraints</td>
<td>577</td>
</tr>
<tr>
<td>13.67</td>
<td>MixedVariables</td>
<td>578</td>
</tr>
<tr>
<td>13.68</td>
<td>Model</td>
<td>580</td>
</tr>
<tr>
<td>13.69</td>
<td>MPIManager</td>
<td>611</td>
</tr>
<tr>
<td>13.70</td>
<td>MPIPackBuffer</td>
<td>612</td>
</tr>
<tr>
<td>13.71</td>
<td>MPIUnpackBuffer</td>
<td>614</td>
</tr>
<tr>
<td>13.72</td>
<td>NCSUOptimizer</td>
<td>616</td>
</tr>
<tr>
<td>13.73</td>
<td>NestedModel</td>
<td>619</td>
</tr>
<tr>
<td>13.74</td>
<td>NIDRProblemDescDB</td>
<td>630</td>
</tr>
<tr>
<td>13.75</td>
<td>NL2ResStruct</td>
<td>635</td>
</tr>
<tr>
<td>13.76</td>
<td>NL2SOLLeastSq</td>
<td>635</td>
</tr>
<tr>
<td>13.77</td>
<td>NLPQLPOptimizer</td>
<td>638</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>13.12</td>
<td>partial_prp_hash Struct Reference</td>
<td>794</td>
</tr>
<tr>
<td>13.12</td>
<td>PecosApproximation Class Reference</td>
<td>795</td>
</tr>
<tr>
<td>13.12</td>
<td>ProblemDescDB Class Reference</td>
<td>799</td>
</tr>
<tr>
<td>13.13</td>
<td>processApplicInterface Class Reference</td>
<td>809</td>
</tr>
<tr>
<td>13.13</td>
<td>ProcessHandleApplicInterface Class Reference</td>
<td>813</td>
</tr>
<tr>
<td>13.13</td>
<td>ProgramOptions Class Reference</td>
<td>818</td>
</tr>
<tr>
<td>13.13</td>
<td>PStudyDACE Class Reference</td>
<td>823</td>
</tr>
<tr>
<td>13.13</td>
<td>PSUADEDesignCompExp Class Reference</td>
<td>825</td>
</tr>
<tr>
<td>13.13</td>
<td>PythonInterface Class Reference</td>
<td>827</td>
</tr>
<tr>
<td>13.13</td>
<td>QuesoJointPdf&lt; V, M &gt; Class Template Reference</td>
<td>829</td>
</tr>
<tr>
<td>13.13</td>
<td>QuesoVectorRV&lt; V, M &gt; Class Template Reference</td>
<td>830</td>
</tr>
<tr>
<td>13.13</td>
<td>RecastModel Class Reference</td>
<td>831</td>
</tr>
<tr>
<td>13.13</td>
<td>relaxedVarConstraints Class Reference</td>
<td>839</td>
</tr>
<tr>
<td>13.14</td>
<td>RelaxedVariables Class Reference</td>
<td>840</td>
</tr>
<tr>
<td>13.14</td>
<td>Response Class Reference</td>
<td>842</td>
</tr>
<tr>
<td>13.14</td>
<td>RestartWriter Class Reference</td>
<td>849</td>
</tr>
<tr>
<td>13.14</td>
<td>ResultsDBAny Class Reference</td>
<td>850</td>
</tr>
<tr>
<td>13.14</td>
<td>ResultsEntry&lt; StoredType &gt; Class Template Reference</td>
<td>852</td>
</tr>
<tr>
<td>13.14</td>
<td>ResultsFileError Class Reference</td>
<td>853</td>
</tr>
<tr>
<td>13.14</td>
<td>ResultsID Class Reference</td>
<td>854</td>
</tr>
<tr>
<td>13.14</td>
<td>ResultsManager Class Reference has to be shown</td>
<td>855</td>
</tr>
<tr>
<td>13.14</td>
<td>ResultsNames Class Reference</td>
<td>857</td>
</tr>
<tr>
<td>13.14</td>
<td>RichExtrapVerification Class Reference</td>
<td>858</td>
</tr>
<tr>
<td>13.15</td>
<td>citalbInterface Class Reference</td>
<td>860</td>
</tr>
<tr>
<td>13.15</td>
<td>SensAnalysisGlobal Class Reference</td>
<td>861</td>
</tr>
<tr>
<td>13.15</td>
<td>SeqHybridMetaIterator Class Reference</td>
<td>864</td>
</tr>
<tr>
<td>13.15</td>
<td>SerialDirectApplicInterface Class Reference</td>
<td>867</td>
</tr>
<tr>
<td>13.15</td>
<td>sharedApproxData Class Reference</td>
<td>868</td>
</tr>
<tr>
<td>13.15</td>
<td>sharedPecosApproxData Class Reference</td>
<td>873</td>
</tr>
<tr>
<td>13.15</td>
<td>sharedResponseData Class Reference</td>
<td>876</td>
</tr>
<tr>
<td>13.15</td>
<td>sharedResponseDataRep Class Reference</td>
<td>878</td>
</tr>
<tr>
<td>13.15</td>
<td>sharedSurfpackApproxData Class Reference</td>
<td>880</td>
</tr>
<tr>
<td>13.15</td>
<td>sharedVariablesData Class Reference</td>
<td>882</td>
</tr>
<tr>
<td>13.16</td>
<td>sharedVariablesDataRep Class Reference</td>
<td>887</td>
</tr>
<tr>
<td>13.16</td>
<td>SimulationResponse Class Reference</td>
<td>892</td>
</tr>
<tr>
<td>13.16</td>
<td>SingleModel Class Reference</td>
<td>893</td>
</tr>
<tr>
<td>13.16</td>
<td>NLLBase Class Reference</td>
<td>895</td>
</tr>
<tr>
<td>13.16</td>
<td>NLLLeastSq Class Reference</td>
<td>897</td>
</tr>
<tr>
<td>13.16</td>
<td>NLLOptimizer Class Reference</td>
<td>901</td>
</tr>
<tr>
<td>13.16</td>
<td>OLBase Class Reference</td>
<td>907</td>
</tr>
<tr>
<td>13.16</td>
<td>SpawnApplicInterface Class Reference</td>
<td>909</td>
</tr>
<tr>
<td>13.16</td>
<td>SurfpackApproximation Class Reference</td>
<td>910</td>
</tr>
<tr>
<td>13.16</td>
<td>SurrBasedGlobalMinimizer Class Reference</td>
<td>914</td>
</tr>
<tr>
<td>13.17</td>
<td>SurrBasedLocalMinimizer Class Reference</td>
<td>915</td>
</tr>
<tr>
<td>13.17</td>
<td>SurrBasedMinimizer Class Reference</td>
<td>922</td>
</tr>
<tr>
<td>13.17</td>
<td>SurrogateModel Class Reference</td>
<td>926</td>
</tr>
<tr>
<td>13.17</td>
<td>SysCallApplicInterface Class Reference</td>
<td>930</td>
</tr>
<tr>
<td>13.17</td>
<td>TabularDataTruncated Class Reference</td>
<td>934</td>
</tr>
<tr>
<td>13.17</td>
<td>TANA3Approximation Class Reference</td>
<td>934</td>
</tr>
</tbody>
</table>
### CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.17</td>
<td>TaylorApproximation Class Reference</td>
<td>936</td>
</tr>
<tr>
<td>13.17</td>
<td>TestDriverInterface Class Reference</td>
<td>937</td>
</tr>
<tr>
<td>13.17</td>
<td>TrackerHTTP Class Reference</td>
<td>942</td>
</tr>
<tr>
<td>13.17</td>
<td>UsageTracker Class Reference</td>
<td>944</td>
</tr>
<tr>
<td>13.18</td>
<td>Var_i_check Struct Reference</td>
<td>945</td>
</tr>
<tr>
<td>13.18</td>
<td>Var_r_check Struct Reference</td>
<td>945</td>
</tr>
<tr>
<td>13.18</td>
<td>Variables Class Reference</td>
<td>946</td>
</tr>
<tr>
<td>13.18</td>
<td>Verification Class Reference</td>
<td>955</td>
</tr>
<tr>
<td>13.18</td>
<td>VLint Struct Reference</td>
<td>957</td>
</tr>
<tr>
<td>13.18</td>
<td>VLreal Struct Reference</td>
<td>957</td>
</tr>
<tr>
<td>13.18</td>
<td>VPSApproximation Class Reference</td>
<td>958</td>
</tr>
<tr>
<td>13.18</td>
<td>WorkdirHelper Class Reference</td>
<td>962</td>
</tr>
<tr>
<td>14</td>
<td>File Documentation</td>
<td>969</td>
</tr>
<tr>
<td>14.1</td>
<td>dakota_dll_api.cpp File Reference</td>
<td>969</td>
</tr>
<tr>
<td>14.2</td>
<td>dakota_dll_api.h File Reference</td>
<td>970</td>
</tr>
<tr>
<td>14.3</td>
<td>dakota_tabular_io.hpp File Reference</td>
<td>971</td>
</tr>
<tr>
<td>14.4</td>
<td>dll_tester.cpp File Reference</td>
<td>973</td>
</tr>
<tr>
<td>14.5</td>
<td>JEGAOptimizer.cpp File Reference</td>
<td>973</td>
</tr>
<tr>
<td>14.6</td>
<td>JEGAOptimizer.hpp File Reference</td>
<td>974</td>
</tr>
<tr>
<td>14.7</td>
<td>library_mode.cpp File Reference</td>
<td>974</td>
</tr>
<tr>
<td>14.8</td>
<td>library_split.cpp File Reference</td>
<td>978</td>
</tr>
<tr>
<td>14.9</td>
<td>main.cpp File Reference</td>
<td>978</td>
</tr>
<tr>
<td>14.10</td>
<td>restart_util.cpp File Reference</td>
<td>979</td>
</tr>
</tbody>
</table>

### Index

- 153
- 174
- 200
- 201
- 206
- 212
- 291
- 507
- 525
- 529
- 542
- 561
- 573
- 584
- 638
- 644
- 662
- 691
- 727
- 730
- 741
- 754
- 773
- 818
- 842
- 850
- 858
- 862
- 874
- 878
- 879
- 889
- 892
- 909
- 920
- 937
- 938
- 942
- 944
- 945
- 946
- 947
- 949
- 953
- 955
- 956
- 957
- 958
- 959
- 960
- 962
- 968
- 970
- 971
- 973
- 974
- 975
- 976
- 978
- 979
- 980
- 981
Chapter 1

Dakota Developers Manual

Author

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 non-gradient-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 of 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:
1.2. OVERVIEW OF DAKOTA

- Optimization: Optimizer provides a base class for gradient-based (e.g., CONMINOptimizer and SNLOptimizer) 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 Surrogate-Model 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 NonDUESOBayesCalibration.
  - 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.
CHAPTER 1. DAKOTA DEVELOPERS MANUAL

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 ForkApplicationInterface.

- 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.
1.5. ADDITIONAL RESOURCES

- **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/](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/](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.:

```cpp
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.:

```cpp
double classMemberVariable;
```

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

```cpp
int temporary_variable;
```

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

```cpp
const double CONSTANT_VALUE;
```
CHAPTER 2. CODING STYLE GUIDELINES AND CONVENTIONS

2.2.2 Function styles

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

```c
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.:

```c
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.:

```c
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.:

```c
const RealVector& c_vars = model.continuous_variables();
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.:

```c
typedef double Real;
```

\texttt{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):

```c
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.:
2.3. FILE NAMING CONVENTIONS

```c
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.:

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

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

```c
cout << "Numerical gradients using " << finiteDiffStepSize*100. << "%" << finiteDiffType << " differences to 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

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.
2.5. CMAKE STYLE GUIDELINES

- 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](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](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](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 keyword-Type 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_undefined`
3.2 DEPRECATED: MODIFY DAKOTA.INPUT.NSPEC

- **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 section, 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:

```plaintext
[method_setting REAL {method_setting_start, &method_setting_details} ]
```
you would provide a function

```plaintext
void NIDRProblemDescDB::
mетод_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

```plaintext
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:

```plaintext
void NIDRProblemDescDB::
mетод_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

```plaintext
void NIDRProblemDescDB::
mетод_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

```plaintext
[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,

```cpp
... 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:

```cpp
{"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).
3.7 USE GET_<DATA_TYPE>() FUNCTIONS

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:

```cpp
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 \texttt{CMAKE\_INSTALL\_PREFIX/bin/}. The core C/C++ code is in the library \texttt{dakota\_src}, while Fortran code lives in the \texttt{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 \textit{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 \texttt{LibraryEnvironment} ("front end"); see \textit{Basic Dakota library instantiation} and \textit{Configuring Dakota operation}.

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

3. Compiling Dakota and linking into your application (\textit{Linking against the Dakota library}).

Several source code examples demonstrate Dakota library interfaces. The classes \texttt{SIM::SerialDirectApplicInterface} and \texttt{SIM::ParallelDirectApplicInterface} demonstrate serial and parallel simulation function evaluation plug-ins. The file \texttt{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, \texttt{library\_split.cpp} demonstrates running Dakota as a library modular on an MPI sub-communicator.

### 5.2 Basic Dakota library instantiation

The function \texttt{run\_dakota\_parse()} in \texttt{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 \texttt{dakota\_optimization.in}) and a function evaluator derived from a \texttt{DirectApplicInterface} is plugged into the Dakota library environment.

First, an object of type \texttt{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 \texttt{ProgramOptions}. See its class documentation for details.

```c
string dakota\_input\_file = "dakota\_optimization.in";
Dakota::ProgramOptions opts;
opts.input\_file(dakota\_input\_file);
```

Next, a \texttt{LibraryEnvironment} is created, passing the desired settings from \texttt{opts}:

```c
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 \texttt{DirectApplicInterface} is plugged in with a convenience function \texttt{serial\_interface\_plugin()} or \texttt{parallel\_interface\_plugin()}. Finally, the Dakota analysis is run by calling

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

The next two sections offer additional details on (1) alternative and supplementary ways to configure Dakota’s operation (\textit{Configuring Dakota operation}) and (2) how to specialize Dakota’s \texttt{DirectApplicInterface} to provide a function evaluator plugin to Dakota (\textit{Creating a simulator plugin interface}).
5.3. CONFIGURING DAKOTA OPERATION

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:

```cpp
// 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:

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

or from a string literal provided by the wrapping application:

```cpp
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:

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

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.

```cpp
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.:

```cpp
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.

```cpp
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.

```cpp
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:

```cpp
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:

```cpp
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.:

```c++
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:

```c++
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

```c++
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.
5.4. CREATING A SIMULATOR PLUGIN INTERFACE

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:

```cpp
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:

```cpp
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:

```cpp
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

} // class SerialDirectApplicInterface

} // 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_rosenbrook");
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 plug in 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

```cpp
Dakota::ModelList filt_models =
    env.filtered_model_list("single", "direct", "plugin_text_book");
Dakota::ProblemDescDB problem_db = env.problem_description_db();
Dakota::ModelIterator 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.:

```c++
// 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/include/, and Fortran code lives 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 are 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; nidd; teuchos; pecos; pecos_src; lfs; mods; mod; fftpack; sparse
```

While external dependencies will be output as:

```
-- Dakota_TPL_LIBRARIES: /usr/lib64/libcurl.so; /usr/lib64/openmpi/lib/libmipi_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):

  ```
  -Ixpm -Ixm -Ixt -Ixmu -Ix -Ixext -Ix11 -Ism -IICE
  ```
• When configuring with AMPL (HAVEAMPL: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 contiguous 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.
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 Mixed-Variables object depending on the active view. The derived classes size the contiguous storage arrays to accommodate 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 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 RealVector &evars)

Make this acceptably efficient
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 325
Chapter 9

Hierarchical Index

9.1 Class Hierarchy

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

- ActiveSet ................................................. 327
- Approximation .......................................... 350
  - GaussProcApproximation ............................... 505
  - PecosApproximation .................................. 795
  - SurfpackApproximation ................................. 910
  - TANA3Approximation .................................. 934
  - TaylorApproximation .................................. 936
  - VPSApproximation ..................................... 958
- APPSEvalMgr ............................................. 363
- BaseConstructor ......................................... 368
- callback_data .............................................. 368
- COLINApplication ....................................... 369
- CommandShell ............................................ 379
- ConsoleRedirector ........................................ 390
- Constraints ................................................. 391
  - MixedVarConstraints .................................. 577
  - RelaxedVarConstraints ................................ 839
- DataEnvironment ......................................... 400
- DataEnvironmentRep ..................................... 401
- DataInterface ............................................ 415
- DataMethod ............................................... 416
- DataMethodRep ............................................ 417
- DataModel .................................................. 430
- DataModelRep .............................................. 431
- DataResponses ............................................. 436
- DataResponsesRep ........................................ 437
- DataVariables ............................................. 441
- DataVariablesRep ......................................... 442
- DiscrepancyCorrection ................................... 463
- JEGAOptimizer::Driver .................................... 470
- Environment ............................................... 480
<table>
<thead>
<tr>
<th>Class/Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecutableEnvironment</td>
<td>492</td>
</tr>
<tr>
<td>LibraryEnvironment</td>
<td>558</td>
</tr>
<tr>
<td>NomadOptimizer::Evaluator</td>
<td>484</td>
</tr>
<tr>
<td>JEGAOptimizer::Evaluator</td>
<td>486</td>
</tr>
<tr>
<td>JEGAOptimizer::EvaluatorCreator</td>
<td>491</td>
</tr>
<tr>
<td>ExperimentData</td>
<td>493</td>
</tr>
<tr>
<td>FileReadException</td>
<td>499</td>
</tr>
<tr>
<td>ResultsFileError</td>
<td>853</td>
</tr>
<tr>
<td>TabularDataTruncated</td>
<td>934</td>
</tr>
<tr>
<td>FunctionEvalFailure</td>
<td>505</td>
</tr>
<tr>
<td>GetLongOpt</td>
<td>510</td>
</tr>
<tr>
<td>CommandLineHandler</td>
<td>377</td>
</tr>
<tr>
<td>Graphics</td>
<td>513</td>
</tr>
<tr>
<td>Interface</td>
<td>521</td>
</tr>
<tr>
<td>ApplicationInterface</td>
<td>336</td>
</tr>
<tr>
<td>DirectApplicInterface</td>
<td>459</td>
</tr>
<tr>
<td>MatlabInterface</td>
<td>561</td>
</tr>
<tr>
<td>PythonInterface</td>
<td>827</td>
</tr>
<tr>
<td>ScilabInterface</td>
<td>860</td>
</tr>
<tr>
<td>TestDriverInterface</td>
<td>937</td>
</tr>
<tr>
<td>ParallelDirectApplicInterface</td>
<td>772</td>
</tr>
<tr>
<td>SerialDirectApplicInterface</td>
<td>867</td>
</tr>
<tr>
<td>ProcessApplicInterface</td>
<td>809</td>
</tr>
<tr>
<td>ProcessHandleApplicInterface</td>
<td>813</td>
</tr>
<tr>
<td>ForkApplicInterface</td>
<td>500</td>
</tr>
<tr>
<td>SpawnApplicInterface</td>
<td>909</td>
</tr>
<tr>
<td>SysCallApplicInterface</td>
<td>930</td>
</tr>
<tr>
<td>GridApplicInterface</td>
<td>515</td>
</tr>
<tr>
<td>ApproximationInterface</td>
<td>357</td>
</tr>
<tr>
<td>Iterator</td>
<td>529</td>
</tr>
<tr>
<td>Analyzer</td>
<td>329</td>
</tr>
<tr>
<td>NonD</td>
<td>647</td>
</tr>
<tr>
<td>EfficientSubspaceMethod</td>
<td>474</td>
</tr>
<tr>
<td>NonDCalibration</td>
<td>668</td>
</tr>
<tr>
<td>NonDBayesCalibration</td>
<td>666</td>
</tr>
<tr>
<td>NonDDREAMBayesCalibration</td>
<td>672</td>
</tr>
<tr>
<td>NonDGPMSABayesCalibration</td>
<td>694</td>
</tr>
<tr>
<td>NonDQUESOBayesCalibration</td>
<td>735</td>
</tr>
<tr>
<td>NonDExpansion</td>
<td>676</td>
</tr>
<tr>
<td>NonDPolynomialChaos</td>
<td>726</td>
</tr>
<tr>
<td>NonDStochCollocation</td>
<td>751</td>
</tr>
<tr>
<td>NonDIntegration</td>
<td>698</td>
</tr>
<tr>
<td>NonDCubature</td>
<td>670</td>
</tr>
<tr>
<td>NonDQuadrature</td>
<td>731</td>
</tr>
<tr>
<td>NonDSparseGrid</td>
<td>749</td>
</tr>
<tr>
<td>NonDInterval</td>
<td>701</td>
</tr>
<tr>
<td>NonDGlobalInterval</td>
<td>684</td>
</tr>
</tbody>
</table>
9.1. CLASS HIERARCHY

- NonDGlobalEvidence ......................................................... 683
- NonDGlobalSingleInterval .................................................. 690
- NonDLHSInterval .............................................................. 704
- NonDLHSEvidence ............................................................. 703
- NonDLHSSingleInterval ..................................................... 708
- NonDLocalInterval ............................................................ 710
- NonDLocalEvidence ........................................................... 709
- NonDLocalSingleInterval ................................................... 722
- NonDPOFDarts ................................................................. 724
- NonDReliability ............................................................... 740
- NonDGlobalReliability ....................................................... 687
- NonDLocalReliability ......................................................... 713
- NonDSampling ................................................................. 742
- NonDAdaptImpSampling ...................................................... 657
- NonDAdaptiveSampling ...................................................... 661
- NonDGPImpSampling .......................................................... 691
- NonDIncremLHSSampling ................................................... 696
- NonDLHSSampling ............................................................. 706
- PStudyDACE ........................................................................ 823
- DDACEDesignCompExp ......................................................... 456
- FSUDesignCompExp ............................................................. 502
- ParamStudy .......................................................................... 789
- PSUADEDesignCompExp ....................................................... 825
- Verification .......................................................................... 955
- RichExtrapVerification ......................................................... 858
- MetaIterator ........................................................................ 563
- CollabHybridMetaIterator ..................................................... 376
- ConcurrentMetaIterator ......................................................... 380
- EmbedHybridMetaIterator .................................................... 478
- SeqHybridMetaIterator ......................................................... 864
- Minimizer ............................................................................ 564
- LeastSq ............................................................................... 554
  - NL2SOLLeastSq ................................................................. 635
  - NLSSOLLeastSq ................................................................. 643
  - SNLLEastSq ....................................................................... 897
- Optimizer ............................................................................. 761
  - APPSOptimizer ................................................................ 365
  - COLINOptimizer ................................................................ 372
  - CONMINOptimizer ............................................................. 383
  - DOTOptimizer .................................................................. 467
  - JEGAOptimizer .................................................................. 548
  - NCSUOptimizer ................................................................. 616
  - NLPQLPOptimizer ............................................................. 638
  - NomadOptimizer ............................................................... 645
  - NonlinearCGOptimizer ....................................................... 754
  - NPSOLOptimizer .............................................................. 756
  - OptDartsOptimizer ............................................................ 759
  - SNLLOptimizer ................................................................. 901
<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SurrBasedMinimizer</td>
<td>922</td>
</tr>
<tr>
<td>EffGlobalMinimizer</td>
<td>472</td>
</tr>
<tr>
<td>SurrBasedGlobalMinimizer</td>
<td>914</td>
</tr>
<tr>
<td>SurrBasedLocalMinimizer</td>
<td>915</td>
</tr>
<tr>
<td>IteratorScheduler</td>
<td>542</td>
</tr>
<tr>
<td>LightWtBaseConstructor</td>
<td>560</td>
</tr>
<tr>
<td>MatchesWC</td>
<td>561</td>
</tr>
<tr>
<td>Model</td>
<td>580</td>
</tr>
<tr>
<td>NestedModel</td>
<td>619</td>
</tr>
<tr>
<td>RecastModel</td>
<td>831</td>
</tr>
<tr>
<td>SingleModel</td>
<td>893</td>
</tr>
<tr>
<td>SurrogateModel</td>
<td>926</td>
</tr>
<tr>
<td>DataFitSurrModel</td>
<td>403</td>
</tr>
<tr>
<td>HierarchSurrModel</td>
<td>516</td>
</tr>
<tr>
<td>MPIManager</td>
<td>611</td>
</tr>
<tr>
<td>MPIPackBuffer</td>
<td>612</td>
</tr>
<tr>
<td>MPIUnpackBuffer</td>
<td>614</td>
</tr>
<tr>
<td>NL2Res</td>
<td>635</td>
</tr>
<tr>
<td>NoDBBaseConstructor</td>
<td>644</td>
</tr>
<tr>
<td>OutputManager</td>
<td>766</td>
</tr>
<tr>
<td>OutputWriter</td>
<td>769</td>
</tr>
<tr>
<td>ParallelConfiguration</td>
<td>770</td>
</tr>
<tr>
<td>ParallelLevel</td>
<td>773</td>
</tr>
<tr>
<td>ParallelLibrary</td>
<td>776</td>
</tr>
<tr>
<td>ParamResponsePair</td>
<td>785</td>
</tr>
<tr>
<td>partial_prp_equality</td>
<td>794</td>
</tr>
<tr>
<td>partial_prp_hash</td>
<td>794</td>
</tr>
<tr>
<td>ProblemDescDB</td>
<td>799</td>
</tr>
<tr>
<td>NIDRProblemDescDB</td>
<td>630</td>
</tr>
<tr>
<td>ProgramOptions</td>
<td>818</td>
</tr>
<tr>
<td>QuesoJointPdf&lt;V, M&gt;</td>
<td>829</td>
</tr>
<tr>
<td>QuesoVectorRV&lt;V, M&gt;</td>
<td>830</td>
</tr>
<tr>
<td>Response</td>
<td>842</td>
</tr>
<tr>
<td>ExperimentResponse</td>
<td>498</td>
</tr>
<tr>
<td>SimulationResponse</td>
<td>892</td>
</tr>
<tr>
<td>RestartWriter</td>
<td>849</td>
</tr>
<tr>
<td>ResultsDBAny</td>
<td>850</td>
</tr>
<tr>
<td>ResultsEntry&lt;StoredType&gt;</td>
<td>852</td>
</tr>
<tr>
<td>ResultsID</td>
<td>854</td>
</tr>
<tr>
<td>ResultsManager</td>
<td>855</td>
</tr>
<tr>
<td>ResultsNames</td>
<td>857</td>
</tr>
<tr>
<td>SensAnalysisGlobal</td>
<td>861</td>
</tr>
<tr>
<td>SharedApproxData</td>
<td>868</td>
</tr>
<tr>
<td>SharedPecosApproxData</td>
<td>873</td>
</tr>
<tr>
<td>SharedSurfpackApproxData</td>
<td>880</td>
</tr>
<tr>
<td>SharedResponseData</td>
<td>876</td>
</tr>
<tr>
<td>SharedResponseDataRep</td>
<td>878</td>
</tr>
<tr>
<td>SharedVariablesData</td>
<td>882</td>
</tr>
</tbody>
</table>
9.1. CLASS HIERARCHY

SharedVariablesDataRep ................................................................. 887
SNLLEntBase ................................................................. 895
  SNLLEastSq ................................................................. 897
  SNLLOptimizer ............................................................ 901
SOLBase ................................................................. 907
  NLSSOLLeastSq .......................................................... 643
  NPSOLOptimizer ........................................................ 756
TrackerHTTP ............................................................ 942
UsageTracker .............................................................. 944
Var_iCHECK .............................................................. 945
Var_rCHECK .............................................................. 945
Variables ................................................................. 946
  MixedVariables .......................................................... 578
  RelaxedVariables ......................................................... 840
VLint ................................................................. 957
VLreal ................................................................. 957
VLstr ................................................................. 958
WorkdirHelper ............................................................. 962
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. 327
- **Analyzer**: Base class for NonD, DACE, and ParamStudy branches of the iterator hierarchy. 329
- **ApplicationInterface**: Derived class within the interface class hierarchy for supporting interfaces to simulation codes 336
- **Approximation**: Base class for the approximation class hierarchy 350
- **ApproximationInterface**: Derived class within the interface class hierarchy for supporting approximations to simulation-based results 357
- **APPSEvalMgr**: Evaluation manager class for APPSPACK 363
- **APPSOptimizer**: Wrapper class for HOPSPACK 365
- **BaseConstructor**: Dummy struct for overloading letter-envelope constructors 368
- **callback_data**: 368
- **COLINApplication**: 369
- **COLINOptimizer**: Wrapper class for optimizers defined using COLIN 372
- **CollabHybridMetaIterator**: Meta-iterator for hybrid iteration using multiple collaborating optimization and nonlinear least squares methods 376
- **CommandLineHandler**: Utility class for managing command line inputs to DAKOTA 377
- **CommandShell**: Utility class which defines convenience operators for spawning processes with system calls 379
- **ConcurrentMetaIterator**: Meta-iterator for multi-start iteration or pareto set optimization 380
CONMINOptimizer
Wrapper class for the CONMIN optimization library ............................................. 383
ConsoleRedirector ................................................................................................. 390
Constraints
Base class for the variable constraints class hierarchy ............................................. 391
DataEnvironment
Handle class for environment specification data ...................................................... 400
DataEnvironmentRep
Body class for environment specification data ....................................................... 401
DataFitSurrModel
Derived model class within the surrogate model branch for managing data fit surrogates (global and local) ................................................................. 403
DataInterface
Handle class for interface specification data .......................................................... 415
DataMethod
Handle class for method specification data ........................................................... 416
DataMethodRep
Body class for method specification data ............................................................... 417
DataModel
Handle class for model specification data ............................................................. 430
DataModelRep
Body class for model specification data ............................................................... 431
DataResponses
Handle class for responses specification data ....................................................... 436
DataResponsesRep
Body class for responses specification data .......................................................... 437
DataVariables
Handle class for variables specification data .......................................................... 441
DataVariablesRep
Body class for variables specification data ........................................................... 442
DDACEDesignCompExp
Wrapper class for the DDACE design of experiments library ................................. 456
DirectApplicInterface
Derived application interface class which spawns simulation codes and testers using direct procedure calls ................................................................. 459
DiscrepancyCorrection
Base class for discrepancy corrections .................................................................. 463
DOTOptimizer
Wrapper class for the DOT optimization library ................................................... 467
JEGAOptimizer::Driver
A subclass of the JEGA front end driver that exposes the individual protected methods to execute the algorithm ................................................................. 470
EffGlobalMinimizer
Implementation of Efficient Global Optimization/Least Squares algorithms ............ 472
EfficientSubspaceMethod
Efficient Subspace Method (ESM), as proposed by Hany S. Abdel-Khalik ................. 474
EmbedHybridMetaIterator
Meta-iterator for closely-coupled hybrid iteration, typically involving the embedding of local search methods within global search methods ................................. 478
### 10.1. CLASS LIST

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Base class for the environment class hierarchy</td>
<td>480</td>
</tr>
<tr>
<td>NomadOptimizer::Evaluator</td>
<td>NOMAD-based Evaluator class</td>
<td>484</td>
</tr>
<tr>
<td>JEGAOptimizer::Evaluator</td>
<td>An evaluator specialization that knows how to interact with Dakota</td>
<td>486</td>
</tr>
<tr>
<td>JEGAOptimizer::EvaluatorCreator</td>
<td>A specialization of the JEGA::Front::EvaluatorCreator that creates a new instance of a Evaluator</td>
<td>491</td>
</tr>
<tr>
<td>ExecutableEnvironment</td>
<td>Environment corresponding to execution as a stand-alone application</td>
<td>492</td>
</tr>
<tr>
<td>ExperimentData</td>
<td>Interpolation method for interpolating between experimental and model data. I need to work on inputs/outputs to this method. For now, this assumes interpolation of functional data</td>
<td>493</td>
</tr>
<tr>
<td>ExperimentResponse</td>
<td>Container class for response functions and their derivatives. ExperimentResponse provides the body class</td>
<td>498</td>
</tr>
<tr>
<td>FileReadException</td>
<td>Base class for Dakota file read exceptions (to allow catching both tabular and general file truncation issues)</td>
<td>499</td>
</tr>
<tr>
<td>ForkApplicInterface</td>
<td>Derived application interface class which spawns simulation codes using fork/execvp/waitpid</td>
<td>500</td>
</tr>
<tr>
<td>FSUDesignCompExp</td>
<td>Wrapper class for the FSUDace QMC/CVT library</td>
<td>502</td>
</tr>
<tr>
<td>FunctionEvalFailure</td>
<td>Exception class for function evaluation failures</td>
<td>505</td>
</tr>
<tr>
<td>GaussProcApproximation</td>
<td>Derived approximation class for Gaussian Process implementation</td>
<td>505</td>
</tr>
<tr>
<td>GetLongOpt</td>
<td>GetLongOpt is a general command line utility from S. Manoharan (Advanced Computer Research Institute, Lyon, France)</td>
<td>510</td>
</tr>
<tr>
<td>Graphics</td>
<td>Single interface to 2D (motif) and 3D (PLPLOT) graphics; there is only one instance of this OutputManager::dakotaGraphics</td>
<td>513</td>
</tr>
<tr>
<td>GridApplicInterface</td>
<td>Derived application interface class which spawns simulation codes using grid services such as Condor or Globus</td>
<td>515</td>
</tr>
<tr>
<td>HierarchSurrModel</td>
<td>Derived model class within the surrogate model branch for managing hierarchical surrogates (models of varying fidelity)</td>
<td>516</td>
</tr>
<tr>
<td>Interface</td>
<td>Base class for the interface class hierarchy</td>
<td>521</td>
</tr>
<tr>
<td>Iterator</td>
<td>Base class for the iterator class hierarchy</td>
<td>529</td>
</tr>
<tr>
<td>IteratorScheduler</td>
<td>This class encapsulates scheduling operations for concurrent sub-iteration within an outer level context (e.g., meta-iteration, nested models)</td>
<td>542</td>
</tr>
<tr>
<td>JEGAOptimizer</td>
<td>A version of Dakota::Optimizer for instantiation of John Eddy’s Genetic Algorithms (JEGA)</td>
<td>548</td>
</tr>
</tbody>
</table>
CHAPTER 10. CLASS INDEX

LeastSq
Base class for the nonlinear least squares branch of the iterator hierarchy .............. 554

LibraryEnvironment
Environment corresponding to execution as an embedded library .......................... 558

LightWtBaseConstructor
Dummy struct for overloading constructors used in on-the-fly Model instantiations .... 560

MatchesWC
Predicate that returns true when the passed path matches the wildcard with which it was
configured. Currently supports * and ? ......................................................... 561

MatlabInterface ................................................................. 561

MetaIterator
Base class for meta-iterators .......................................................................... 563

Minimizer
Base class for the optimizer and least squares branches of the iterator hierarchy .... 564

MixedVarConstraints
Derived class within the Constraints hierarchy which separates continuous and discrete variables (no domain type array merging) ............................................................... 577

MixedVariables
Derived class within the Variables hierarchy which separates continuous and discrete variables
(no domain type array merging) ........................................................................ 578

Model
Base class for the model class hierarchy .............................................................. 580

MPIManager
Class MPIManager to manage Dakota's MPI world, which may be a subset of MPI_COMM_WORLD ................................................................. 611

MPIPackBuffer
Class for packing MPI message buffers .............................................................. 612

MPIUnpackBuffer
Class for unpacking MPI message buffers .......................................................... 614

NCSUOptimizer
Wrapper class for the NCSU DIRECT optimization library ................................. 616

NestedModel
Derived model class which performs a complete sub-iterator execution within every evaluation
of the model ......................................................................................................... 619

NIDRProblemDescDB
The derived input file database utilizing the new IDR parser ................................. 630

NL2Res
Auxiliary information passed to calcr and calcj via ur ......................................... 635

NL2SOLLeastSq
Wrapper class for the NL2SOL nonlinear least squares library ............................. 635

NLPQLPOptimizer
Wrapper class for the NLPQLP optimization library, Version 2.0 ......................... 638

NLSSOLLeastSq
Wrapper class for the NLSSOL nonlinear least squares library ............................. 643

NoDBBaseConstructor
Dummy struct for overloading constructors used in on-the-fly instantiations without ProblemDescDB support ................................................................. 644

NomadOptimizer
Wrapper class for NOMAD Optimizer ................................................................. 645
10.1. CLASS LIST

NonD
Base class for all nondeterministic iterators (the DAKOTA/UQ branch) ........................................ 647

NonDAdaptImpSampling
Class for the Adaptive Importance Sampling methods within DAKOTA ............................. 657

NonDAdaptiveSampling
Class for testing various Adaptively sampling methods using geometric, statistical, and topological information of the surrogate .................................................. 661

NonDBayesCalibration
Base class for Bayesian inference: generates posterior distribution on model parameters given experimental data ................................................................. 666

NonDCalibration ................................................................. 668

NonDCubature
Derived nondeterministic class that generates N-dimensional numerical cubature points for evaluation of expectation integrals .................................................. 670

NonDDREAMBayesCalibration
Bayesian inference using the DREAM approach ........................................................................ 672

NonDExpansion
Base class for polynomial chaos expansions (PCE) and stochastic collocation (SC) ............ 676

NonDGlobalEvidence
Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ .................. 683

NonDGlobalInterval
Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification ........................................ 684

NonDGlobalReliability
Class for global reliability methods within DAKOTA/UQ .......................................................... 687

NonDGlobalSingleInterval
Class for using global nongradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification ........................................ 690

NonDGPIMpSampling
Class for the Gaussian Process-based Importance Sampling method ........................................ 691

NonDGPMASBayesCalibration
Generates posterior distribution on model parameters given experimental data ................ 694

NonDLHSSampling
Performs incremental LHS sampling for uncertainty quantification .......................................... 696

NonDIntegration
Derived nondeterministic class that generates N-dimensional numerical integration points for evaluation of expectation integrals ........................................... 698

NonDInterval
Base class for interval-based methods within DAKOTA/UQ .................................................... 701

NonDLHSEvidence
Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ .................. 703

NonDLHSInterval
Class for the LHS-based interval methods within DAKOTA/UQ ........................................... 704

NonDLHSSampling
Performs LHS and Monte Carlo sampling for uncertainty quantification ................................. 706

NonDLHSSingleInterval
Class for pure interval propagation using LHS .......................................................................... 708

NonDLLocalEvidence
Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ .................. 709
NonDLocalInterval
Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification .......................................................... 710

NonDLocalReliability
Class for the reliability methods within DAKOTA/UQ ........................................ 713

NonDLocalSingleInterval
Class for using local gradient-based optimization approaches to calculate interval bounds for epistemic uncertainty quantification ....................................................... 722

NonDPOFDarts
Base class for POF Dart methods within DAKOTA/UQ ........................................ 724

NonDPolynomialChaos
Nonintrusive polynomial chaos expansion approaches to uncertainty quantification .... 726

NonDQuadrature
Derived nondeterministic class that generates N-dimensional numerical quadrature points for evaluation of expectation integrals over uncorrelated standard normals/uniforms/exponentials/betas/gammas 731

NonDQUESOBayesCalibration
Bayesian inference using the QUESO library from UT Austin .............................. 735

NonDReliability
Base class for the reliability methods within DAKOTA/UQ ..................................... 740

NonDSampling
Base class for common code between NonDLHSSampling, NonDIncremLHSSampling, and NonDAdaptImpSampling ................................................................. 742

NonDSparseGrid
Derived nondeterministic class that generates N-dimensional Smolyak sparse grids for numerical evaluation of expectation integrals over independent standard random variables ........................................................................................................ 749

NonDStochCollocation
Nonintrusive stochastic collocation approaches to uncertainty quantification ............ 751

NonlinearCGOptimizer ......................................................................................... 754

NPSOLOptimizer
Wrapper class for the NPSOL optimization library .............................................. 756

OptDartsOptimizer
Wrapper class for OptDarts Optimizer ............................................................... 759

Optimizer
Base class for the optimizer branch of the iterator hierarchy .............................. 761

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 ....................................................... 766

OutputWriter ................................................................................................. 769

ParallelConfiguration
Container class for a set of ParallelLevel list iterators that collectively identify a particular multilevel parallel configuration ................................................................. 770

ParallelDirectApplicInterface
Sample derived interface class for testing parallel simulator plug-ins using assign_rep() ................................................................. 772

ParallelLevel
Container class for the data associated with a single level of communicator partitioning . 773
10.1. CLASS LIST

**ParallelLibrary**
Class for partitioning multiple levels of parallelism and managing message passing within these levels ............................................................. 776

**ParamResponsePair**
Container class for a variables object, a response object, and an evaluation id .................. 785

**ParamStudy**
Class for vector, list, centered, and multidimensional parameter studies ....................... 789

**partial\_prp\_equality**
Predicate for comparing ONLY the interfaceId and Vars attributes of PRPair ..................... 794

**partial\_prp\_hash**
Wrapper to delegate to the ParamResponsePair hash_value function ............................... 794

**PecosApproximation**
Derived approximation class for global basis polynomials ........................................... 795

**ProblemDescDB**
The database containing information parsed from the DAKOTA input file ......................... 799

**ProcessApplicInterface**
Derived application interface class that spawns a simulation code using a separate process and communicates with it through files ........................................... 809

**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 ............. 813

**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() ....................... 818

**PStudyDACE**
Base class for managing common aspects of parameter studies and design of experiments methods ................................................................. 823

**PSUADEDesignCompExp**
Wrapper class for the PSUADE library ........................................................................ 825

**PythonInterface** ........................................................................................................ 827

**QuesoJointPdf\(< V, M >**
Dakota specialization of QUESO generic joint PDF ...................................................... 829

**QuesoVectorRV\(< V, M >**
Dakota specialization of QUESO vector-valued random variable .................................... 830

**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 .................................................. 831

**RelaxedVarConstraints**
Derived class within the Constraints hierarchy which employs relaxation of discrete variables 839

**RelaxedVariables**
Derived class within the Variables hierarchy which employs the relaxation of discrete variables 840

**Response**
Container class for response functions and their derivatives. Response provides the enveloper base class ................................................................. 842

**RestartWriter** ............................................................................................................. 849

**ResultsDBAny** ............................................................................................................ 850

**ResultsEntry\(< StoredType >**
Class to manage in-core vs. file database lookups .................................................... 852
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResultsFileError</td>
<td>Exception throw for other results file read error</td>
<td>853</td>
</tr>
<tr>
<td>ResultsID</td>
<td>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</td>
<td>854</td>
</tr>
<tr>
<td>ResultsManager</td>
<td>Results manager for iterator final data</td>
<td>855</td>
</tr>
<tr>
<td>ResultsNames</td>
<td>List of valid names for iterator results</td>
<td>857</td>
</tr>
<tr>
<td>RichExtrapVerification</td>
<td>Class for Richardson extrapolation for code and solution verification</td>
<td>858</td>
</tr>
<tr>
<td>ScilabInterface</td>
<td></td>
<td>860</td>
</tr>
<tr>
<td>SensAnalysisGlobal</td>
<td>Class for a utility class containing correlation calculations and variance-based decomposition</td>
<td>861</td>
</tr>
<tr>
<td>SeqHybridMetaIterator</td>
<td>Method for sequential hybrid iteration using multiple optimization and nonlinear least squares methods on multiple models of varying fidelity</td>
<td>864</td>
</tr>
<tr>
<td>SerialDirectApplicInterface</td>
<td>Sample derived interface class for testing serial simulator plug-ins using assign_rep()</td>
<td>867</td>
</tr>
<tr>
<td>SharedApproxData</td>
<td>Base class for the shared approximation data class hierarchy</td>
<td>868</td>
</tr>
<tr>
<td>SharedPecosApproxData</td>
<td>Derived approximation class for global basis polynomials</td>
<td>873</td>
</tr>
<tr>
<td>SharedResponseData</td>
<td>Container class encapsulating variables data that can be shared among a set of Response instances</td>
<td>876</td>
</tr>
<tr>
<td>SharedResponseDataRep</td>
<td>The representation of a SharedResponseData instance. This representation, or body, may be shared by multiple SharedResponseData handle instances</td>
<td>878</td>
</tr>
<tr>
<td>SharedSurfpackApproxData</td>
<td>Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota</td>
<td>880</td>
</tr>
<tr>
<td>SharedVariablesData</td>
<td>Container class encapsulating variables data that can be shared among a set of Variables instances</td>
<td>882</td>
</tr>
<tr>
<td>SharedVariablesDataRep</td>
<td>The representation of a SharedVariablesData instance. This representation, or body, may be shared by multiple SharedVariablesData handle instances</td>
<td>887</td>
</tr>
<tr>
<td>SimulationResponse</td>
<td>Container class for response functions and their derivatives. SimulationResponse provides the body class</td>
<td>892</td>
</tr>
<tr>
<td>SingleModel</td>
<td>Derived model class which utilizes a single interface to map variables into responses</td>
<td>893</td>
</tr>
<tr>
<td>SNLLBase</td>
<td>Base class for OPT++ optimization and least squares methods</td>
<td>895</td>
</tr>
<tr>
<td>SNLLLeastSq</td>
<td>Wrapper class for the OPT++ optimization library</td>
<td>897</td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>SNLLOptimizer</td>
<td>Wrapper class for the OPT++ optimization library</td>
<td>901</td>
</tr>
<tr>
<td>SOLBase</td>
<td>Base class for Stanford SOL software</td>
<td>907</td>
</tr>
<tr>
<td>SpawnApplicInterface</td>
<td>Derived application interface class which spawns simulation codes using spawnvp</td>
<td>909</td>
</tr>
<tr>
<td>SurfpackApproximation</td>
<td>Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota</td>
<td>910</td>
</tr>
<tr>
<td>SurrBasedGlobalMinimizer</td>
<td>The global surrogate-based minimizer which sequentially minimizes and updates a global surrogate model without trust region controls</td>
<td>914</td>
</tr>
<tr>
<td>SurrBasedLocalMinimizer</td>
<td>Class for provably-convergent local surrogate-based optimization and nonlinear least squares</td>
<td>915</td>
</tr>
<tr>
<td>SurrBasedMinimizer</td>
<td>Base class for local/global surrogate-based optimization/least squares</td>
<td>922</td>
</tr>
<tr>
<td>SurrogateModel</td>
<td>Base class for surrogate models (DataFitSurrModel and HierarchSurrModel)</td>
<td>926</td>
</tr>
<tr>
<td>SysCallApplicInterface</td>
<td>Derived application interface class which spawns simulation codes using system calls</td>
<td>930</td>
</tr>
<tr>
<td>TabularDataTruncated</td>
<td>Exception thrown when data read truncated</td>
<td>934</td>
</tr>
<tr>
<td>TANA3Approximation</td>
<td>Derived approximation class for TANA-3 two-point exponential approximation (a multipoint approximation)</td>
<td>934</td>
</tr>
<tr>
<td>TaylorApproximation</td>
<td>Derived approximation class for first- or second-order Taylor series (a local approximation)</td>
<td>936</td>
</tr>
<tr>
<td>TestDriverInterface</td>
<td></td>
<td>937</td>
</tr>
<tr>
<td>TrackerHTTP</td>
<td>TrackerHTTP: a usage tracking module that uses HTTP/HTTPS via the curl library</td>
<td>942</td>
</tr>
<tr>
<td>UsageTracker</td>
<td>Lightweight class to manage conditionally active Curl-based HTTP tracker via PIMPL</td>
<td>944</td>
</tr>
<tr>
<td>Var_check</td>
<td>Structure for verifying bounds and initial point for string-valued vars</td>
<td>945</td>
</tr>
<tr>
<td>Var_rcheck</td>
<td>Structure for verifying bounds and initial point for real-valued vars</td>
<td>945</td>
</tr>
<tr>
<td>Variables</td>
<td>Base class for the variables class hierarchy</td>
<td>946</td>
</tr>
<tr>
<td>Verification</td>
<td>Base class for managing common aspects of verification studies</td>
<td>955</td>
</tr>
<tr>
<td>VLInt</td>
<td>Structure for validating integer uncertain variable labels, bounds, values</td>
<td>957</td>
</tr>
<tr>
<td>VLreal</td>
<td>Structure for validating real uncertain variable labels, bounds, values</td>
<td>957</td>
</tr>
<tr>
<td>VLstr</td>
<td>Structure for validating string uncertain variable labels, bounds, values</td>
<td>958</td>
</tr>
<tr>
<td>VPSApproximation</td>
<td>Derived approximation class for VPS implementation</td>
<td>958</td>
</tr>
<tr>
<td>WorkdirHelper</td>
<td></td>
<td>962</td>
</tr>
</tbody>
</table>
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 ........................................ 969

- **dakota_dll_api.h**
  
  API for DLL interactions ........................................................................................................ 970

- **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 facilitate rework. These augment (and leverage) those in data_util.h ......................................................... 971

- **dll_tester.cpp**
  
  Test the DLL with a DAKOTA input file .................................................................................... 973

- **JEGAOptimizer.cpp**
  
  Contains the implementation of the JEGAOptimizer class ..................................................... 973

- **JEGAOptimizer.hpp**
  
  Contains the definition of the JEGAOptimizer class ............................................................... 974

- **library_mode.cpp**
  
  File containing a mock simulator main for testing Dakota in library mode ......................... 974

- **library_split.cpp**
  
  File containing a mock simulator main for testing DAKOTA in library mode on a split communicator ......................................................................................................................... 978

- **main.cpp**
  
  File containing the main program for DAKOTA .......................................................................... 978

- **restart_util.cpp**
  
  File containing the DAKOTA restart utility main program ..................................................... 979
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.
- class FileReadException
base class for Dakota file read exceptions (to allow catching both tabular and general file truncation issues)

- class TabularDataTruncated
  exception thrown when data read truncated
- class ResultsFileError
  exception throw for other results file read error
- class FunctionEvalFailure
  exception class for function evaluation failures
- 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 nondeterministic 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.
12.1. DAKOTA NAMESPACE REFERENCE

- class Response
  Container class for response functions and their derivatives. Response provides the enveloper base 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**
  
  Interpolation method for interpolating between experimental and model data. I need to work on inputs/outputs to this method. For now, this assumes interpolation of functional data.

- **class ExperimentResponse**
  
  Container class for response functions and their derivatives. ExperimentResponse provides the body class.

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

- **class FSUDesignCompExp**
  
  Wrapper class for the FSU Dace 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**
  
  This class encapsulates scheduling operations for concurrent sub-iteration within an outer level context (e.g., meta-iteration, nested models).

- **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.
12.1. DAKOTA NAMESPACE REFERENCE

- **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, statistical, 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 non-gradient-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 incremental 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 `NonDLLocalEvidence`  
  Class for the Dempster-Shafer Evidence Theory methods within DAKOTA/UQ.

• class `NonDLLocalInterval`  
  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 `NonDLLocalSingleInterval`  
  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 `QuesoJointPdf`  
  Dakota specialization of QUESO generic joint PDF.

• class `QuesoVectorRV`
12.1. DAKOTA NAMESPACE REFERENCE

*Dakota* specialization of QUESO vector-valued random variable.

- **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 SharedResponseDataRep

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

class SharedResponseData

Container class encapsulating variables data that can be shared among a set of Response instances.
12.1. DAKOTA NAMESPACE REFERENCE

- 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 SimulationResponse
  Container class for response functions and their derivatives. SimulationResponse provides the body class.
- 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.
• class VPSApproximation
  Derived approximation class for VPS implementation.
• struct MatchesWC
  Predicate that returns true when the passed path matches the wildcard with which it was configured. Currently supports * and ?.
• class WorkdirHelper

Typedef
• 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<br>bool > BoolDeque
• typedef boost::dynamic_bitset
  < unsigned long > BitArray
• typedef std::vector<br>BoolDeque > BoolDequeArray
• typedef std::vector<br> Real > RealArray
• typedef std::vector<br> RealArray > Real2DArray
• typedef std::vector<br> int > IntArray
• typedef std::vector<br> IntArray > Int2DArray
• typedef std::vector<br> short > ShortArray
• typedef std::vector<br> unsigned short > UShortArray
• typedef std::vector<br> UShortArray > UShort2DArray
• typedef std::vector
  < UShort2DArray > UShort3DArray
• typedef std::vector<br> size_t > SizetArray
• typedef std::vector<br> SizetArray > Sizet2DArray
• typedef std::vector<br> String > StringArray
• typedef std::vector<br> 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< RealVector > 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::map<IntRealMap> IntRealMapArray

typedef std::vector<RealRealMap> RealRealMapArray

typedef std::vector<StringRealMap> StringRealMapArray

typedef std::vector<IntRealVectorMap> IntRealVectorMap

typedef std::vector<IntRealMatrixMap> IntRealMatrixMap

typedef std::vector<IntActiveSetMap> IntActiveSetMap

typedef std::vector<IntVariablesMap> IntVariablesMap

typedef std::vector<IntResponseMap> IntResponseMap

typedef std::vector<IntArray, size_t> IntArraySizetMap

typedef std::vector<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**
CHAPTER 12. NAMESPACE DOCUMENTATION

• typedef RealRealMap::const_iterator RRMCIter
• typedef IntIntPairRealMap::iterator IIPRMIter
• typedef IntIntPairRealMap::const_iterator IIPRCIter
• 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, std::string > 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)
12.1. DAKOTA NAMESPACE REFERENCE

**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 PRPCacheHIter

typedef PRPCacheOIter PRPCacheIter
default cache iterator <0>

typedef PRPCacheOCIter PRPCacheCIter
default cache const iterator <0>

typedef boost::reverse_iterator
< PRPCacheCIter > PRPCacheCRevIter

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
call function) > file_op_function
12.1. DAKOTA NAMESPACE REFERENCE

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 {
  METHOD_ERROR = -7, 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 {
  TABULAR_NONE = 0, TABULAR_HEADER = 1, TABULAR_EVAL_ID = 2, TABULAR_IFACE_ID = 4,
  TABULAR_EXPER_ANNOT = TABULAR_HEADER | TABULAR_EVAL_ID, TABULAR_ANNOTATED = TABULAR_HEADER | TABULAR_EVAL_ID | TABULAR_IFACE_ID }

  options for tabular columns
- 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, G-
PAIS,
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_P-
ATTERN_SEARCH = (MINIMIZER_BIT | OPTIMIZER_BIT),
OPTPP_PDS, COLINY_BETA, COLINY_COBYLA, COLINY_DIRECT,
COLINY_MULTI_START, COLINY_EA, COLINY_PATTERN_SEARCH, COLINY_SOLIS_WET-
S, 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,
NLPOOL_SQP, DOT_BFGS, DOT_FRCG, DOT_MFD,
DOT_SLP, DOT_SQP, CONMIN_FRCG, CONMIN_MFD,
DL_SOLVER }

• enum {
  SUBMETHOD_DEFAULT = 0, SUBMETHOD_COLLABORATIVE, SUBMETHOD_EMBEDDED, SUB-
METHOD_SEQUENTIAL,
  SUBMETHOD_LHS, SUBMETHOD_RANDOM, SUBMETHOD_INCREMENTAL_LHS, SUBME-
THOD_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, SUB-
METHOD_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_C-
  OVARIANCE }

• 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_UNIFORM,
    EPISTEMIC_UNCERTAIN, EPISTEMIC_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, HOMOTOPIA, 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 { BASE_RESPONSE = 0, SIMULATION_RESPONSE, EXPERIMENT_RESPONSE }

    special values for derived Response type

enum { GENERIC_FNS = 0, OBJECTIVE_FNS, CALIB_TERMS }

    values for primary response types

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 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_DEURV,
    TOTAL_CSV,
    TOTAL_DSV,
    TOTAL_DSIV,
    TOTAL_DSSV,
    TOTAL_DSRV,
    NUM_VC_TOTALS
}

• enum var_t {
    VAR_x1, VAR_x2, VAR_x3, VAR_b,
    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
}

• enum driver_t {
    NO_DRIVER = 0,
    CANTILEVER_BEAM,
    MOD_CANTILEVER_BEAM,
    CYLINDER_HEAD,
    EXTENDED_ROSENROCK,
    GENERALIZED_ROSENROCK,
    LF_ROSENROCK,
    ROSENROCK,
    GERSTNER,
    SCALABLEGERSTNER,
    LOGNORMAL_RATIO,
    MULTIMODAL,
    PLUGIN_ROSENROCK,
    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,
    DAMPED_OSCILLATOR
}

• enum local_data_t {
    VARIABLES_MAP = 1,
    VARIABLES_VECTOR = 2
}

enumeration for how local variables are stored (values must employ a bit representation)
12.1. DAKOTA NAMESPACE REFERENCE

- enum sigtype { NO_SIGMA, SCALAR_SIGMA, DIAGONAL_SIGMA, MATRIX_SIGMA }
  special values for sigmaType
- 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 { DAUVar_poisson = 0, DAUVar_binomial = 1, DAUVar_negative_binomial = 2, DAUVar_geometric = 3, DAUVar_hypergeometric = 4, DAUVar_histogram_point_int = 5, DAUVar_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 { DEUVar_set_int = 0, DEUVar_Nkinds = 2 }
- enum { DEURVar_set_int = 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, CGHESTENES_STIEFEL }
  NonlinearCG update options.
- enum CG_LINESEARCHTYPE { CG_FIXED_STEP, CG_LS_SIMPLE, CG_LS_BRENT, CG_LS_WOLFE }
  NonlinearCG linesearch options.
- enum EvauType { 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

- void read_sized_data (std::istream &, RealVectorArray &va, size_t num_rows, int num_cols)

- void read_fixed_rowsize_data (std::istream &, RealVectorArray &va, int num_cols, bool row_major)

- void read_unsized_data (std::istream &, RealVectorArray &va, bool row_major)

- void read_config_vars_multifile (const std::string &basename, int num_expts, int ncv, RealVectorArray &config_vars)

- void read_config_vars_singlefile (const std::string &basename, int num_expts, int ncv, RealVectorArray &config_vars)

- void read_field_values (const std::string &basename, int expt_num, RealVectorArray &field_vars)

- void read_field_values (const std::string &basename, int expt_num, RealVector &field_vars)

- void read_coord_values (const std::string &basename, int expt_num, RealMatrix &coords)

- void read_coord_values (const std::string &basename, RealMatrix &coords)

- void read_covariance (const std::string &basename, int expt_num, RealMatrix &cov_vals)

- void read_covariance (const std::string &basename, int expt_num, Dakota::CovarianceMatrix::FORMAT format, int num_vals, RealMatrix &cov_vals)

- void copy_data (const RealMatrix &rmat, RealVectorArray &rvarray)

  file reader for VECTOR and MATRIX covariance data

- void copy_data (const RealVectorArray &rvarray, RealMatrix &rmat)

- 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 &sas, SizetMultiArrayConstView smav)

  equality operator for SizetArray and SizetMultiArrayConstView

- bool operator!=(const StringMultiArrayView &smav1, const StringMultiArrayView &smav2)

  inequality operator for StringMultiArray view vs. container

- Real rel_change_L2 (const RealVector &currrv, const RealVector &prevrv)

  Computes relative change between RealVectors using Euclidean L2 norm.

- Real rel_change_L2 (const RealVector &currrv1, const RealVector &prevrv1, const IntVector &curriv, const IntVector &previv, const RealVector &currrv2, const RealVector &prevrv2)

  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)
12.1. DAKOTA NAMESPACE REFERENCE

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)

  create a label by appending a numerical tag to the root_label, o

  void build_label (String &label, const String &root_label, size_t tag, const String &separator = "")

  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 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 (StringArray &label_array, const String &root_label, size_t start_index, size_t num_items)

  void copy_row_vector (const RealMatrix &m, RealMatrix::ordinalType i, std::vector<Real>&row)

  Copies a row of a Teuchos::SerialDenseMatrix to std::vector<Real>

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

  Copy row<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, Teuchos::SerialDenseVector< 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<T > &da, 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<std::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 std::vector<T> (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 Teuchos::SerialDenseVector<OrdinalType, ScalarType>& sdv, Teuchos::SerialDenseVector<OrdinalType, ScalarType>& sdva, OrdinalType2 num_vec, OrdinalType2 vec_len)
  copy SerialDenseVector<> to Array<SerialDenseVector<>>

• template<typename OrdinalType1 , typename OrdinalType2 , typename ScalarType>
  void copy_data (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> &da1, std::vector<T> &da2, size_t start_index2)

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>

• template<typename 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

• template<typename T>
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 key

• 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 &l, const typename ListT::value_type &val)

  return an iterator to the first list element satisfying the predicate 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)
- MetaDataValueType make_metadatavalue (const std::string &)
- MetaDataValueType make_metadatavalue (const std::string &, const std::string &)
- MetaDataValueType make_metadatavalue (const std::string &, const std::string &, const std::string &)
- 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::ostream&).
- 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)`
  - **inequality operator for Response**
- `std::string re_match(const std::string &token, const boost::regex &re)`
  - *Global utility function to ease migration from CielRegExp 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)
CHAPTER 12. NAMESPACE DOCUMENTATION

- 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 *, Dakotafuncs *, 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 RealMatrix const **linear_eq_constraint_targets1** (Optimizer1 *o)
static RealMatrix const **linear_eq_constraint_coeffs1** (Optimizer1 *o)
static int **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 **copy_field_data** (const RealVector &fn_vals, RealMatrix &fn_grad, const RealSymMatrixArray &fn_hess, size_t offset, size_t num_fns, Response &response)
- void **copy_field_data** (const RealVector &fn_vals, RealMatrix &fn_grad, const RealSymMatrixArray &fn_hess, size_t offset, size_t num_fns, short total_asv, Response &response)
- void **interpolate_simulation_field_data** (const Response &sim_resps, const RealMatrix &exp_coords, size_t field_num, short total_asv, size_t interp resp offset, Response &interp_resp)
- void **linear_interpolate_1d** (const RealMatrix &build pts, const RealVector &build_vals, const RealMatrix &build_grads, const RealSymMatrixArray &build_hessians, const RealMatrix &pred_pts, RealVector &pred_vals, RealMatrix &pred_grads, RealSymMatrixArray &pred_hessians)

  Returns the value of at 1D function f and its gradient and hessians (if available) at the points of vector pred pts using linear interpolation. The vector build pts specifies the coordinates of the underlying interval at which the values (build_vals) of the function f are known. The length of output pred vals is equal to the length of pred pts.

  This function assumes the build pts is in ascending order.
- void **symmetric_eigenvalue_decomposition** (const RealSymMatrix &matrix, RealVector &eigenvalues, RealMatrix &eigenvectors)

  Computes the eigenvalues and, optionally, eigenvectors of a real symmetric matrix A.
12.1. DAKOTA NAMESPACE REFERENCE

- **bool get_positive_definite_covariance_from_hessian**(const RealSymMatrix &hessian, RealMatrix &covariance)
  
  Compute the covariance \( C \) of a hessian matrix \( H \), i.e., \( C=\text{inv}(H) \) The hessian is enforced to be positive definite by setting any negative eigenvalues to zero. The state of eigenvalue truncation is returned.

- **template<typename O , typename T > int binary_search**(T target, Teuchos::SerialDenseVector<O, T> &data)
  
  find the interval containing a target value. This function assumes the data is in ascending order.

- **void copy_residuals**(const RealVector &fn_vals, RealMatrix &fn_grad, const RealSymMatrixArray &fn_hess, size_t offset, size_t num_fns, Response &response)
  
  Copy a field into a response object.

- **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, const 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)
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 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_ContinuousState (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 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)
- 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 specification)

- 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)
- static void Vgen_HistogramPtIntUnc (DataVariablesRep *dv, size_t offset)
- static void Vchk_HistogramPtStrUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void Vgen_HistogramPtStrUnc (DataVariablesRep *dv, size_t offset)
- static void Vchk_HistogramPtRealUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void Vgen_HistogramPtRealUnc (DataVariablesRep *dv, size_t offset)
- static void Vchk_DiscreteIntervalUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void Vgen_DiscreteIntervalUnc (DataVariablesRep *dv, size_t offset)
- static void Vchk_ContinuousIntervalUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void Vgen_ContinuousIntervalUnc (DataVariablesRep *dv, size_t offset)

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 Vchk_HistogramPtIntUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void Vgen_HistogramPtIntUnc (DataVariablesRep *dv, size_t offset)
- static void Vchk_HistogramPtStrUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void Vgen_HistogramPtStrUnc (DataVariablesRep *dv, size_t offset)
- static void Vchk_HistogramPtRealUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void Vgen_HistogramPtRealUnc (DataVariablesRep *dv, size_t offset)
- static void Vchk_ContinuousIntervalUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void Vgen_ContinuousIntervalUnc (DataVariablesRep *dv, size_t offset)
- static void Vchk_DiscreteIntervalUnc (DataVariablesRep *dv, size_t offset, Var_Info *vi)
- static void Vgen_DiscreteIntervalUnc (DataVariablesRep *dv, size_t offset)
- static void Vchk_DIset (size_t num_v, 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_nds, IntVector *input_dsi, IntSet-Array &dsi_all, IntVector &dsi_init_pt)
check discrete sets of integers (design and state variables); error if a duplicate value is specified error if not ordered to prevent user confusion

- static void Vchk_DIsset (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 error if not ordered to prevent user confusion

- static void Vchk_DSset (size_t num_v, const char *kind, IntArrayOfString *input_ndss, StringArray *input_dss, StringSetArray &dss_all, StringArray &dss_init_pt)
- static void Vchk_DSset (size_t num_v, const char *kind, IntArrayOfString *input_ndss, StringArray *input_dss, RealVector *input_dsdp, StringRealMapArray &dsdss_vals_probs, StringArray &dsdss_init_pt)
- static void Vchk_DSset (size_t num_v, const char *kind, IntArrayOfReal *input_ndsr, RealVector *input_dsr, RealSetArray &dsr_all, RealVector &dsr_init_pt)
- static void Vchk_DSSet (size_t num_v, const char *kind, IntArrayOfString *input_ndsr, RealVector *input_dsr, RealVector *input_dsdp, RealVector &dsdss_init_pt)
- static void Vchk_Adjacency (size_t num_v, const char *kind, const IntArray &num_e, const IntVector &input_ddsa, RealMatrixArray &dda_all)

static bool check_LUV_size (size_t num_v, IntVector &L, IntVector &U, IntVector &V, bool aggregate_LUV=false, size_t offset)

static bool check_LUV_size (size_t num_v, StringArray &L, StringArray &U, StringArray &V, bool aggregate_LUV=false, size_t offset)

static bool check_LUV_size (size_t num_v, RealVector &L, RealVector &U, RealVector &V, bool aggregate_LUV=false, size_t offset)

static void Vgen_DIsset (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_DIsset (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_DSset (size_t num_v, RealSetArray &sets, RealVector &L, RealVector &U, RealVector &V, bool aggregate_LUV=false, size_t offset=0)
- static void Vgen_DSSet (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)
12.1. DAKOTA NAMESPACE REFERENCE

- 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_jcheck *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, IntVector **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 &rrprm, 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 &iiprm, RealVector **probs, IntVector **lb, IntVector **ub)

Flatten integer-valued interval uncertain variable intervals and probabilities back into separate arrays.

- static void var_julbl (const char *keyname, Values *val, VarLabel *vl)
- static Iface_mp_Rlit MP3 (failAction, recoveryFnVals, recover)
- static Iface_mp_iilit 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_ (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_iilit2 MP3 (replacementType, numberRetained, chc)
• static Method_mp_iilit2 MP3 (replacementType, numberRetained, elitist)
• static Method_mp_iilit2z MP3 (replacementType, numberRetained, random)
• static Method_mp_iilit2z MP3 (crossoverType, numCrossPoints, multi_point_binary)
• static Method_mp_lit2z MP3 (crossoverType, numCrossPoints, multi_point_parameterized_binary)
• static Method_mp_lit2z 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, adaptive_metropolis)
• static Method_mp_lit MP2 (mcmcType, delayed_rejection)
• static Method_mp_lit MP2 (mcmcType, dram)
• static Method_mp_lit MP2 (mcmcType, metropolis_hastings)
• static Method_mp_lit MP2 (mcmcType, multilevel)
• 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 (proposalCovInputType, diagonal)
• static Method_mp_lit MP2 (proposalCovInputType, matrix)
• static Method_mp_lit MP2 (proposalCovType, derivatives)
• 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 (proposalCovInputType, diagonal)
• static Method_mp_lit MP2 (proposalCovInputType, matrix)
• static Method_mp_lit MP2 (proposalCovType, derivatives)
• 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, num2)
• 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_ute 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)
12.1. DAKOTA NAMESPACE REFERENCE

- 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_ (proposalCovData)
• 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_ (expansionExportFile)
• static UShortArray MP_ (expansionImportFile)
• 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_ (proposalCovFile)
• 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_ (adaptPosteriorRefine)
• static bool MP_ (approxImportActive)
• 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_ (fixedSeedFlag)
• static bool MP_ (defaultSeedFlag)
• static bool MP_ (randomSeedFlag)
• 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_ (neighborOrder)
• static int MP_ (newSolnsGenerated)
• static int MP_ (numChains)
• static int MP_ (numIterations)
• static int MP_ (numSolnsGenerated)
• 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_ (proposalCovUpdates)
• 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 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, LOCAL_ADAPTIVE_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 (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_type MP2s (approxExportFormat, TABULAR_NONE)
- static Method_mp_type MP2s (approxExportFormat, TABULAR_HEADER)
- static Method_mp_type MP2s (approxExportFormat, TABULAR_EVAL_ID)
- static Method_mp_type MP2s (approxExportFormat, TABULAR_IFACE_ID)
- static Method_mp_type MP2s (approxExportFormat, TABULAR_ANNOTATED)
- static Method_mp_type MP2s (approxImportFormat, TABULAR_NONE)
- static Method_mp_type MP2s (approxImportFormat, TABULAR_HEADER)
- static Method_mp_type MP2s (approxImportFormat, TABULAR_EVAL_ID)
- static Method_mp_type MP2s (approxImportFormat, TABULAR_IFACE_ID)
- static Method_mp_type MP2s (approxImportFormat, TABULAR_ANNOTATED)
- static Method_mp_type MP2s (integrationRefine, AIS)
- static Method_mp_type MP2s (integrationRefine, IS)
- static Method_mp_type MP2s (integrationRefine, MMAIS)
- static Method_mp_type MP2s (methodName, ASYNCH_PATTERN_SEARCH)
- static Method_mp_type MP2s (methodName, COLINY_BETA)
- static Method_mp_type MP2s (methodName, COLINY_COBYLA)
- static Method_mp_type MP2s (methodName, COLINY_DIRECT)
- static Method_mp_type MP2s (methodName, COLINY_EA)
- static Method_mp_type MP2s (methodName, COLINY_PATTERN_SEARCH)
- static Method_mp_type MP2s (methodName, COLINY_SOLIS_WETS)
- static Method_mp_type MP2s (methodName, CONMIN_FRCG)
- static Method_mp_type MP2s (methodName, CONMIN_MFD)
- static Method_mp_type MP2s (methodName, DACE)
- static Method_mp_type MP2s (methodName, DOT_BFGS)
- static Method_mp_type MP2s (methodName, DOT_FRCG)
- static Method_mp_type MP2s (methodName, DOT_MMFD)
- static Method_mp_type MP2s (methodName, DOT_SLP)
- static Method_mp_type MP2s (methodName, DOT_SQP)
- static Method_mp_type MP2s (methodName, EFFICIENT_GLOBAL)
- static Method_mp_type MP2s (methodName, FSU_CVT)
- static Method_mp_type MP2s (methodName, FSU_HALTON)
- static Method_mp_type MP2s (methodName, FSU_HAMMERSLEY)
- static Method_mp_type MP2s (methodName, HYBRID)
- static Method_mp_type MP2s (methodName, MESH_ADAPTIVE_SEARCH)
- static Method_mp_type MP2s (methodName, MOGA)
- static Method_mp_type MP2s (methodName, MULTI_START)
- static Method_mp_type MP2s (methodName, NCSU_DIRECT)
- static Method_mp_type MP2s (methodName, NL2SOL)
- static Method_mp_type MP2s (methodName, NLPQL_SQP)
- static Method_mp_type MP2s (methodName, NLSSOL_SQP)
- static Method_mp_type MP2s (methodName, ADAPTIVE_SAMPLING)
- static Method_mp_type MP2s (methodName, BAYES_CALIBRATION)
- static Method_mp_type MP2s (methodName, EFFICIENT_SUBSPACE)
- static Method_mp_type MP2s (methodName, GENIE_DIRECT)
• static Method $\text{mp\_utype MP2s (methodName, GENIE\_OPT\_DARTS)}$
• static Method $\text{mp\_utype MP2s (methodName, GPAIS)}$
• static Method $\text{mp\_utype MP2s (methodName, GLOBAL\_EVIDENCE)}$
• static Method $\text{mp\_utype MP2s (methodName, GLOBAL\_INTERVAL\_EST)}$
• static Method $\text{mp\_utype MP2s (methodName, GLOBAL\_RELIABILITY)}$
• static Method $\text{mp\_utype MP2s (methodName, IMPORTANCE\_SAMPLING)}$
• static Method $\text{mp\_utype MP2s (methodName, LOCAL\_EVIDENCE)}$
• static Method $\text{mp\_utype MP2s (methodName, LOCAL\_INTERVAL\_EST)}$
• static Method $\text{mp\_utype MP2s (methodName, LOCAL\_RELIABILITY)}$
• static Method $\text{mp\_utype MP2s (methodName, POF\_DARTS)}$
• static Method $\text{mp\_utype MP2s (methodName, POLYNOMIAL\_CHAOSS)}$
• static Method $\text{mp\_utype MP2s (methodName, RANDOM\_SAMPLING)}$
• static Method $\text{mp\_utype MP2s (methodName, STOCH\_COLLOCATION)}$
• static Method $\text{mp\_utype MP2s (methodName, NONLINEAR\_CG)}$
• static Method $\text{mp\_utype MP2s (methodName, NPSOL\_SQP)}$
• static Method $\text{mp\_utype MP2s (methodName, OPTPP\_CG)}$
• static Method $\text{mp\_utype MP2s (methodName, OPTPP\_FD\_NEWTON)}$
• static Method $\text{mp\_utype MP2s (methodName, OPTPP\_G\_NEWTON)}$
• static Method $\text{mp\_utype MP2s (methodName, OPTPP\_NEWTON)}$
• static Method $\text{mp\_utype MP2s (methodName, OPTPP\_PDS)}$
• static Method $\text{mp\_utype MP2s (methodName, OPTPP\_Q\_NEWTON)}$
• static Method $\text{mp\_utype MP2s (methodName, PARETO\_SET)}$
• static Method $\text{mp\_utype MP2s (methodName, PSUADE\_MOAT)}$
• static Method $\text{mp\_utype MP2s (methodName, RICHARDSON\_EXTRAP)}$
• static Method $\text{mp\_utype MP2s (methodName, SOGA)}$
• static Method $\text{mp\_utype MP2s (methodName, SURROGATE\_BASED\_GLOBAL)}$
• static Method $\text{mp\_utype MP2s (methodName, SURROGATE\_BASED\_LOCAL)}$
• static Method $\text{mp\_utype MP2s (methodName, VECTOR\_PARAMETER\_STUDY)}$
• static Method $\text{mp\_utype MP2s (methodName, LIST\_PARAMETER\_STUDY)}$
• static Method $\text{mp\_utype MP2s (methodName, CENTERED\_PARAMETER\_STUDY)}$
• static Method $\text{mp\_utype MP2s (methodName, MULTIDIM\_PARAMETER\_STUDY)}$
• static Method $\text{mp\_utype MP2s (pstudyFileFormat, TABULAR\_NONE)}$
• static Method $\text{mp\_utype MP2s (pstudyFileFormat, TABULAR\_HEADER)}$
• static Method $\text{mp\_utype MP2s (pstudyFileFormat, TABULAR\_EVAL\_ID)}$
• static Method $\text{mp\_utype MP2s (pstudyFileFormat, TABULAR\_IFACE\_ID)}$
• static Method $\text{mp\_utype MP2s (pstudyFileFormat, TABULAR\_ANNOTATED)}$
• static Method $\text{mp\_utype MP2s (reliabilitySearchType, AMV\_PLUS\_U)}$
• static Method $\text{mp\_utype MP2s (reliabilitySearchType, AMV\_PLUS\_X)}$
• static Method $\text{mp\_utype MP2s (reliabilitySearchType, AMV\_U)}$
• static Method $\text{mp\_utype MP2s (reliabilitySearchType, AMV\_X)}$
• static Method $\text{mp\_utype MP2s (reliabilitySearchType, EGRA\_U)}$
• static Method $\text{mp\_utype MP2s (reliabilitySearchType, EGRA\_X)}$
• static Method $\text{mp\_utype MP2s (reliabilitySearchType, NO\_APPROX)}$
• static Method $\text{mp\_utype MP2s (reliabilitySearchType, TANA\_U)}$
• static Method $\text{mp\_utype MP2s (reliabilitySearchType, TANA\_X)}$
• static Method $\text{mp\_utype MP2s (sampleType, SUBMETHOD\_INCREMENTAL\_LHS)}$
• static Method_mp_utype \texttt{MP2s} (sampleType, \texttt{SUBMETHOD_INCREMENTAL_RANDOM})
• static Method_mp_utype \texttt{MP2s} (sampleType, \texttt{SUBMETHOD_LHS})
• static Method_mp_utype \texttt{MP2s} (sampleType, \texttt{SUBMETHOD_RANDOM})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_COLLABORATIVE})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_EMBEDDED})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_SEQUENTIAL})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_DREAM})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_GPMSA})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_QUESO})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_NIP})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_SQP})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_EA})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_EGO})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_SBO})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_LHS})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHODRANDOM})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHODOA_LHS})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_OAS})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_BOX_BEHNKEN})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_CENTRAL_COMPOSITE})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{SUBMETHOD_GRID})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{CONVERGE_ORDER})
• static Method_mp_utype \texttt{MP2s} (subMethod, \texttt{CONVERGE_QOI})
• static Method_mp_utype \texttt{MP2s} (surrogateFnIndices)
• static Model_mp_lit \texttt{MP2} (approxPointReuse, all)
• static Model_mp_lit \texttt{MP2} (approxPointReuse, none)
• static Model_mp_lit \texttt{MP2} (marsInterpolation, linear)
• static Model_mp_lit \texttt{MP2} (marsInterpolation, cubic)
• static Model_mp_lit \texttt{MP2} (approxPoint, nested)
• static Model_mp_lit \texttt{MP2} (modelType, single)
• static Model_mp_lit \texttt{MP2} (modelType, surrogate)
• static Model_mp_lit \texttt{MP2} (surrogateType, hierarchical)
• static Model_mp_lit \texttt{MP2} (surrogateType, global_gaussian)
• static Model_mp_lit \texttt{MP2} (surrogateType, global_kriging)
• static Model_mp_lit \texttt{MP2} (surrogateType, global_mars)
• static Model_mp_lit \texttt{MP2} (surrogateType, global_moving_least_squares)
• static Model_mp_lit \texttt{MP2} (surrogateType, global_neural_network)
• static Model_mp_lit \texttt{MP2} (surrogateType, global_polynomial)
• static Model_mp_lit \texttt{MP2} (surrogateType, global_radial Basis)
• static Model_mp_lit \texttt{MP2} (surrogateType, global_voronoi surrogate)
• static Model_mp_lit \texttt{MP2} (surrogateType, local_taylor)
• static Model_mp_lit \texttt{MP2} (surrogateType, multipoint_tana)
• static Model_mp_lit \texttt{MP2} (trendOrder, constant)
• static Model_mp_lit \texttt{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 Model_mp_utype MP2s (approxChallengeFormat, TABULAR_NONE)
• static Model_mp_utype MP2s (approxChallengeFormat, TABULAR_HEADER)
• static Model_mp_utype MP2s (approxChallengeFormat, TABULAR_EVAL_ID)
• static Model_mp_utype MP2s (approxChallengeFormat, TABULAR_IFACE_ID)
• static Model_mp_utype MP2s (approxChallengeFormat, TABULAR_ANNOTATED)
• static Real MP_(annRange)
• static Real MP_ (discontGradThresh)
• static Real MP_ (discontJumpThresh)
• 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_ (decompCellType)
• 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_ (crossValidateFlag)
• static bool MP_ (decompDiscontDetect)
• static bool MP_ (hierarchicalTags)
- static bool MP_(modelUseDerivsFlag)
- static bool MP_(piecewiseDecomp)
- static bool MP_(pointSelection)
- static bool MP_(pressFlag)
- 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_(decompSupportLayers)
- 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_(coordList)
- static RealVector MP_(expConfigVars)
- static RealVector MP_(expObservations)
- 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 (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 (coordDataFileName)
• static String MP (scalarDataFileName)
• static String MP (idResponses)
• static StringArray MP (nonlinearEqScaleTypes)
• static StringArray MP (nonlinearIneqScaleTypes)
• static StringArray MP (primaryRespFnScaleTypes)
• static StringArray MP (primaryRespFnSense)
• static StringArray MP (responseLabels)
• static StringArray MP (varianceType)
• static bool MP (calibrationDataFlag)
• static bool MP (centralHess)
• static bool MP (interpolateFlag)
• static bool MP (ignoreBounds)
• static bool MP (readFieldCoords)
• 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 (numNonlinearEqConstraints)
• static size_t MP (numNonlinearIneqConstraints)
• static size_t MP (numObjectiveFunctions)
• static size_t MP (numResponseLeastSqTerms)
• static size_t MP (numScalarLeastSqTerms)
• static size_t MP (numScalarObjectiveFunctions)
• static size_t MP (numScalarResponseFunctions)
• static Resp_mp_utype MP2s (scalarDataFormat, TABULAR_NONE)
• static Resp_mp_utype MP2s (scalarDataFormat, TABULAR_HEADER)
• static Resp_mp_utype MP2s (scalarDataFormat, TABULAR_EVAL_ID)
• static Resp_mp_utype MP2s (scalarDataFormat, TABULAR_EXPERSANNOT)
• static Env_mp_utytype MP2s (postRunInputFormat, TABULAR_NONE)
• static Env_mp_utytype MP2s (postRunInputFormat, TABULAR_HEADER)
• static Env_mp_utytype MP2s (postRunInputFormat, TABULAR_EVAL_ID)
• static Env_mp_utytype MP2s (postRunInputFormat, TABULAR_IFACE_ID)
• static Env_mp_utytype MP2s (postRunInputFormat, TABULAR_ANNOTATED)
• static Env_mp_utytype MP2s (preRunOutputFormat, TABULAR_NONE)
• static Env_mp_utytype MP2s (preRunOutputFormat, TABULAR_HEADER)
• static Env_mp_utytype MP2s (preRunOutputFormat, TABULAR_EVAL_ID)
• static Env_mp_utytype MP2s (preRunOutputFormat, TABULAR_IFACE_ID)
• static Env_mp_utytype MP2s (preRunOutputFormat, TABULAR_ANNOTATED)
• static Env_mp_utytype MP2s (tabularFormat, TABULAR_NONE)
• static Env_mp_utytype MP2s (tabularFormat, TABULAR_HEADER)
• static Env_mp_utytype MP2s (tabularFormat, TABULAR_EVAL_ID)
• static Env_mp_utytype MP2s (tabularFormat, TABULAR_IFACE_ID)
• static Env_mp_utytype MP2s (tabularFormat, TABULAR_ANNOTATED)
• 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)
12.1. DAKOTA NAMESPACE REFERENCE

- 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_(ddsia)
- static IntVector VP_(ddssa)
- static IntVector VP_(ddsra)
- static IntVector VP_(dusi)
• static IntArray VP_ (nddsi)
• static IntArray VP_ (ndss)
• static IntArray VP_ (nddr)
• static IntArray VP_ (ndss)
• static IntArray VP_ (nddr)
• static IntArray VP_ (ndusi)
• static IntArray VP_ (nduss)
• static IntArray VP_ (ndusr)
• static IntArray VP_ (nhip)
• static IntArray VP_ (nhip)
• static IntArray VP_ (nhsp)
• static IntArray VP_ (nhsp)
• static IntArray VP_ (nCI)
• static IntArray VP_ (nDl)
• 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)
12.1. DAKOTA NAMESPACE REFERENCE

- static RealVector VP_ (ddsr)
- static RealVector VP_ (dssr)
- static RealVector VP_ (dusr)
- static RealVector VP_ (CIlb)
- static RealVector VP_ (Club)
- static RealVector VP_ (Clp)
- 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_ (hpic)
- static RealVector VP_ (hpz)
- static RealVector VP_ (hpza)
- 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_ (discreteUncSetIntLabels)
- static StringArray MP_ (discreteUncSetStrLabels)
- static StringArray MP_ (discreteUncSetRealLabels)
- 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 (betaUncBetacs, 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, std::numeric_limits<Real>::infinity())
• static Var_brv MP2s (lognormalUncVars, 0.)
• static Var_brv MP2s (lognormalUncZetas, 0.)
• static Var_brv MP2s (loguniformUncLowerBnds, 0.)
• static Var_brv MP2s (loguniformUncUpperBnds, std::numeric_limits<Real>::infinity())
• static Var_brv MP2s (loguniformUncVars, 0.)
• static Var_brv MP2s (normalUncStdDevs, 0.)
• static Var_brv MP2s (poissonUncLambdas, 0.)
• static Var_brv MP2s (triangularUncLowerBnds,-std::numeric_limits<Real>::infinity())
• static Var_brv MP2s (triangularUncUpperBnds, std::numeric_limits<Real>::infinity())
• static Var_brv MP2s (uniformUncLowerBnds,-std::numeric_limits<Real>::infinity())
• static Var_brv MP2s (uniformUncUpperBnds, std::numeric_limits<Real>::infinity())
• 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)

• template<class ContainerT>
  void flatten_num_array (const std::vector< ContainerT >&input_array, IntArray *pia)
12.1. DAKOTA NAMESPACE REFERENCE

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 *liv, int *lv, Real *v, int *ui, void *ur, Vf)
- void dn2f_ (int *n, int *p, Real *x, Calcrj, int *iv, int *lv, Real *v, int *ui, void *ur, Vf)
- void dn2g_ (int *n, int *p, Real *x, Calcrj, Calcrj, int *iv, int *lv, Real *v, int *ui, void *ur, Vf)
- void dn2gb_ (int *n, int *p, Real *x, Real *b, Calcrj, Calcrj, int *iv, int *lv, Real *v, int *ui, void *ur, Vf)
- void divset_ (int *, int *, int *, int *, Real *)
- double dr7mdc_ (int *)
- static void Rswapchk (NI2Misc *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 & operator>>(MPIUnpackBuffer &s, ParallelLevel &pl) 
  
  MPIPackBuffer & operator<< (MPIPackBuffer &s, const ParallelLevel &pl)
  
  MPIPackBuffer & operator<< (MPIPackBuffer &s, const ParallelLevel &pl)
  
  std::istream & operator>>(std::istream &s, ParamResponsePair &pair)
  
  std::istream & operator>>(std::istream &s, ParamResponsePair &pair)
  
  MPIUnpackBuffer & operator>> (MPIUnpackBuffer &s, ParamResponsePair &pair)
  
  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 &pr)

hash_value for ParamResponsePairs stored in a PRPMultiIndex

- PRPCacheHIter hashedCacheBegin (PRPCache &pr_cache)

hashed definition of cache begin

- PRPCacheHIter hashedCacheEnd (PRPCache &pr_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 &pr_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 &pr_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

- PRPCacheHIter lookup_by_nearby_val (PRPMultiIndexCache &pr_cache, const String &search_interface_id, const Variables &search_vars, const ActiveSet &search_set, Real tol)

- PRPCacheHIter lookup_by_ids (PRPMultiIndexCache &pr_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
12.1. DAKOTA NAMESPACE REFERENCE

- PRPCacheOIter `lookup_by_ids` (PRPMultiIndexCache &prp_cache, const IntStringPair &search_ids, 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 ParamResponsePair &search_pr)
  find a ParamResponsePair within a PRPMultiIndexQueue based on interface id, variables, and ActiveSet search data
- PRPQueueHIter `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_usage` (std::ostream &s)
  print restart utility help message
- void `print_restart` (StringArray pos_args, String print_dest)
  print a restart file
- void `print_restart_pdb` (StringArray pos_args, String print_dest)
  print a restart file (PDB format)
- void `print_restart_tabular` (StringArray pos_args, String print_dest, unsigned short tabular_format)
  print a restart file (tabular format)
- void `read_neutral` (StringArray pos_args)
  read a restart file (neutral file format)
- void `repair_restart` (StringArray pos_args, String identifier_type)
  repair a restart file by removing corrupted evaluations
- void `concatenate_restart` (StringArray pos_args)
  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_path)

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_mg
  dummy MPIManager for ref initialization
- ProgramOptions dummy_prg_opt
  dummy ProgramOptions for ref initialization
- OutputManager dummy_out_mg
  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
- 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 const int MPI_COMM_WORLD = 1
- static const int MPI_COMM_NULL = 0
- static const int MPI_ANY_TAG = -1
- static void * MPI_REQUEST_NULL = NULL
- static GuiKeyWord kw_1 [3]
- static GuiKeyWord kw_2 [3]
- static GuiKeyWord kw_3 [2]
- static GuiKeyWord kw_4 [3]
12.1. DAKOTA NAMESPACE REFERENCE

- static GuiKeyWord kw_5 [3]
- static GuiKeyWord kw_6 [2]
- static GuiKeyWord kw_7 [1]
- static GuiKeyWord kw_8 [1]
- static GuiKeyWord kw_9 [2]
- static GuiKeyWord kw_10 [3]
- static GuiKeyWord kw_11 [5]
- static GuiKeyWord kw_12 [15]
- static GuiKeyWord kw_13 [1]
- static GuiKeyWord kw_14 [4]
- static GuiKeyWord kw_15 [1]
- static GuiKeyWord kw_16 [4]
- static GuiKeyWord kw_17 [1]
- static GuiKeyWord kw_18 [8]
- static GuiKeyWord kw_19 [9]
- static GuiKeyWord kw_20 [12]
- static GuiKeyWord kw_21 [2]
- static GuiKeyWord kw_22 [2]
- static GuiKeyWord kw_23 [3]
- static GuiKeyWord kw_24 [2]
- static GuiKeyWord kw_25 [2]
- static GuiKeyWord kw_26 [9]
- static GuiKeyWord kw_27 [2]
- static GuiKeyWord kw_28 [1]
- static GuiKeyWord kw_29 [1]
- static GuiKeyWord kw_30 [2]
- static GuiKeyWord kw_31 [4]
- static GuiKeyWord kw_32 [3]
- static GuiKeyWord kw_33 [3]
- static GuiKeyWord kw_34 [3]
- static GuiKeyWord kw_35 [3]
- static GuiKeyWord kw_36 [4]
- static GuiKeyWord kw_37 [2]
- static GuiKeyWord kw_38 [3]
- static GuiKeyWord kw_39 [2]
- static GuiKeyWord kw_40 [15]
- static GuiKeyWord kw_41 [7]
- static GuiKeyWord kw_42 [2]
- static GuiKeyWord kw_43 [19]
- static GuiKeyWord kw_44 [3]
- static GuiKeyWord kw_45 [3]
- static GuiKeyWord kw_46 [3]
- static GuiKeyWord kw_47 [4]
- static GuiKeyWord kw_48 [6]
- static GuiKeyWord kw_49 [2]
- static GuiKeyWord kw_50 [2]
- static GuiKeyWord kw_51 [1]
- static GuiKeyWord kw_52 [5]
- static GuiKeyWord kw_53 [6]
- static GuiKeyWord kw_54 [1]
- static GuiKeyWord kw_55 [2]
- static GuiKeyWord kw_56 [2]
- static GuiKeyWord kw_57 [4]
- static GuiKeyWord kw_58 [2]
- static GuiKeyWord kw_59 [3]
- static GuiKeyWord kw_60 [3]
- static GuiKeyWord kw_61 [3]
- static GuiKeyWord kw_62 [4]
- static GuiKeyWord kw_63 [10]
- static GuiKeyWord kw_64 [3]
- static GuiKeyWord kw_65 [3]
- static GuiKeyWord kw_66 [3]
- static GuiKeyWord kw_67 [4]
- static GuiKeyWord kw_68 [6]
- static GuiKeyWord kw_69 [2]
- static GuiKeyWord kw_70 [2]
- static GuiKeyWord kw_71 [1]
- static GuiKeyWord kw_72 [5]
- static GuiKeyWord kw_73 [9]
- static GuiKeyWord kw_74 [9]
- static GuiKeyWord kw_75 [4]
- static GuiKeyWord kw_76 [7]
- static GuiKeyWord kw_77 [8]
- static GuiKeyWord kw_78 [2]
- static GuiKeyWord kw_79 [12]
- static GuiKeyWord kw_80 [3]
- static GuiKeyWord kw_81 [2]
- static GuiKeyWord kw_82 [3]
- static GuiKeyWord kw_83 [2]
- static GuiKeyWord kw_84 [5]
- static GuiKeyWord kw_85 [4]
- static GuiKeyWord kw_86 [15]
- static GuiKeyWord kw_87 [3]
- static GuiKeyWord kw_88 [2]
- static GuiKeyWord kw_89 [2]
- static GuiKeyWord kw_90 [18]
- static GuiKeyWord kw_91 [14]
- static GuiKeyWord kw_92 [12]
- static GuiKeyWord kw_93 [10]
- static GuiKeyWord kw_94 [1]
- static GuiKeyWord kw_95 [15]
- static GuiKeyWord kw_96 [15]
• static GuiKeyWord kw_97 [3]
• static GuiKeyWord kw_98 [3]
• static GuiKeyWord kw_99 [2]
• static GuiKeyWord kw_100 [3]
• static GuiKeyWord kw_101 [4]
• static GuiKeyWord kw_102 [7]
• static GuiKeyWord kw_103 [9]
• static GuiKeyWord kw_104 [3]
• static GuiKeyWord kw_105 [1]
• static GuiKeyWord kw_106 [9]
• static GuiKeyWord kw_107 [1]
• static GuiKeyWord kw_108 [11]
• static GuiKeyWord kw_109 [3]
• static GuiKeyWord kw_110 [3]
• static GuiKeyWord kw_111 [3]
• static GuiKeyWord kw_112 [4]
• static GuiKeyWord kw_113 [2]
• static GuiKeyWord kw_114 [3]
• static GuiKeyWord kw_115 [2]
• static GuiKeyWord kw_116 [11]
• static GuiKeyWord kw_117 [2]
• static GuiKeyWord kw_118 [2]
• static GuiKeyWord kw_119 [3]
• static GuiKeyWord kw_120 [2]
• static GuiKeyWord kw_121 [3]
• static GuiKeyWord kw_122 [3]
• static GuiKeyWord kw_123 [2]
• static GuiKeyWord kw_124 [3]
• static GuiKeyWord kw_125 [4]
• static GuiKeyWord kw_126 [5]
• static GuiKeyWord kw_127 [12]
• static GuiKeyWord kw_128 [2]
• static GuiKeyWord kw_129 [3]
• static GuiKeyWord kw_130 [3]
• static GuiKeyWord kw_131 [2]
• static GuiKeyWord kw_132 [3]
• static GuiKeyWord kw_133 [4]
• static GuiKeyWord kw_134 [5]
• static GuiKeyWord kw_135 [8]
• static GuiKeyWord kw_136 [2]
• static GuiKeyWord kw_137 [1]
• static GuiKeyWord kw_138 [1]
• static GuiKeyWord kw_139 [3]
• static GuiKeyWord kw_140 [3]
• static GuiKeyWord kw_141 [3]
• static GuiKeyWord kw_142 [4]
• static GuiKeyWord kw_143 [2]
• static GuiKeyWord kw_144 [3]
• static GuiKeyWord kw_145 [2]
• static GuiKeyWord kw_146 [2]
• static GuiKeyWord kw_147 [16]
• static GuiKeyWord kw_148 [2]
• static GuiKeyWord kw_149 [1]
• static GuiKeyWord kw_150 [2]
• static GuiKeyWord kw_151 [1]
• static GuiKeyWord kw_152 [1]
• static GuiKeyWord kw_153 [5]
• static GuiKeyWord kw_154 [1]
• static GuiKeyWord kw_155 [2]
• static GuiKeyWord kw_156 [8]
• static GuiKeyWord kw_157 [2]
• static GuiKeyWord kw_158 [3]
• static GuiKeyWord kw_159 [2]
• static GuiKeyWord kw_160 [12]
• static GuiKeyWord kw_161 [3]
• static GuiKeyWord kw_162 [4]
• static GuiKeyWord kw_163 [3]
• static GuiKeyWord kw_164 [2]
• static GuiKeyWord kw_165 [1]
• static GuiKeyWord kw_166 [1]
• static GuiKeyWord kw_167 [2]
• static GuiKeyWord kw_168 [3]
• static GuiKeyWord kw_169 [2]
• static GuiKeyWord kw_170 [7]
• static GuiKeyWord kw_171 [3]
• static GuiKeyWord kw_172 [5]
• static GuiKeyWord kw_173 [4]
• static GuiKeyWord kw_174 [10]
• static GuiKeyWord kw_175 [1]
• static GuiKeyWord kw_176 [2]
• static GuiKeyWord kw_177 [4]
• static GuiKeyWord kw_178 [2]
• static GuiKeyWord kw_179 [7]
• static GuiKeyWord kw_180 [17]
• static GuiKeyWord kw_181 [2]
• static GuiKeyWord kw_182 [5]
• static GuiKeyWord kw_183 [3]
• static GuiKeyWord kw_184 [1]
• static GuiKeyWord kw_185 [6]
• static GuiKeyWord kw_186 [3]
• static GuiKeyWord kw_187 [2]
• static GuiKeyWord kw_188 [1]
12.1. DAKOTA NAMESPACE REFERENCE

- static GuiKeyWord kw_189 [3]
- static GuiKeyWord kw_190 [1]
- static GuiKeyWord kw_191 [2]
- static GuiKeyWord kw_192 [4]
- static GuiKeyWord kw_193 [22]
- static GuiKeyWord kw_194 [1]
- static GuiKeyWord kw_195 [1]
- static GuiKeyWord kw_196 [7]
- static GuiKeyWord kw_197 [2]
- static GuiKeyWord kw_198 [5]
- static GuiKeyWord kw_199 [10]
- static GuiKeyWord kw_200 [2]
- static GuiKeyWord kw_201 [2]
- static GuiKeyWord kw_202 [3]
- static GuiKeyWord kw_203 [2]
- static GuiKeyWord kw_204 [10]
- static GuiKeyWord kw_205 [1]
- static GuiKeyWord kw_206 [2]
- static GuiKeyWord kw_207 [4]
- static GuiKeyWord kw_208 [2]
- static GuiKeyWord kw_209 [3]
- static GuiKeyWord kw_210 [4]
- static GuiKeyWord kw_211 [2]
- static GuiKeyWord kw_212 [3]
- static GuiKeyWord kw_213 [1]
- static GuiKeyWord kw_214 [1]
- static GuiKeyWord kw_215 [2]
- static GuiKeyWord kw_216 [2]
- static GuiKeyWord kw_217 [1]
- static GuiKeyWord kw_218 [17]
- static GuiKeyWord kw_219 [3]
- static GuiKeyWord kw_220 [6]
- static GuiKeyWord kw_221 [3]
- static GuiKeyWord kw_222 [3]
- static GuiKeyWord kw_223 [6]
- static GuiKeyWord kw_224 [3]
- static GuiKeyWord kw_225 [2]
- static GuiKeyWord kw_226 [4]
- static GuiKeyWord kw_227 [3]
- static GuiKeyWord kw_228 [2]
- static GuiKeyWord kw_229 [5]
- static GuiKeyWord kw_230 [2]
- static GuiKeyWord kw_231 [30]
- static GuiKeyWord kw_232 [1]
- static GuiKeyWord kw_233 [4]
- static GuiKeyWord kw_234 [1]
• static GuiKeyWord kw_235 [13]
• static GuiKeyWord kw_236 [3]
• static GuiKeyWord kw_237 [3]
• static GuiKeyWord kw_238 [2]
• static GuiKeyWord kw_239 [3]
• static GuiKeyWord kw_240 [2]
• static GuiKeyWord kw_241 [2]
• static GuiKeyWord kw_242 [4]
• static GuiKeyWord kw_243 [2]
• static GuiKeyWord kw_244 [4]
• static GuiKeyWord kw_245 [2]
• static GuiKeyWord kw_246 [28]
• static GuiKeyWord kw_247 [2]
• static GuiKeyWord kw_248 [13]
• static GuiKeyWord kw_249 [12]
• static GuiKeyWord kw_250 [11]
• static GuiKeyWord kw_251 [3]
• static GuiKeyWord kw_252 [4]
• static GuiKeyWord kw_253 [16]
• static GuiKeyWord kw_254 [5]
• static GuiKeyWord kw_255 [2]
• static GuiKeyWord kw_256 [1]
• static GuiKeyWord kw_257 [10]
• static GuiKeyWord kw_258 [4]
• static GuiKeyWord kw_259 [5]
• static GuiKeyWord kw_260 [2]
• static GuiKeyWord kw_261 [2]
• static GuiKeyWord kw_262 [2]
• static GuiKeyWord kw_263 [2]
• static GuiKeyWord kw_264 [4]
• static GuiKeyWord kw_265 [20]
• static GuiKeyWord kw_266 [15]
• static GuiKeyWord kw_267 [7]
• static GuiKeyWord kw_268 [2]
• static GuiKeyWord kw_269 [7]
• static GuiKeyWord kw_270 [1]
• static GuiKeyWord kw_271 [4]
• static GuiKeyWord kw_272 [6]
• static GuiKeyWord kw_273 [13]
• static GuiKeyWord kw_274 [4]
• static GuiKeyWord kw_275 [90]
• static GuiKeyWord kw_276 [1]
• static GuiKeyWord kw_277 [2]
• static GuiKeyWord kw_278 [7]
• static GuiKeyWord kw_279 [2]
• static GuiKeyWord kw_280 [1]
• static GuiKeyWord kw_281 [3]
• static GuiKeyWord kw_282 [4]
• static GuiKeyWord kw_283 [6]
• static GuiKeyWord kw_284 [2]
• static GuiKeyWord kw_285 [2]
• static GuiKeyWord kw_286 [3]
• static GuiKeyWord kw_287 [3]
• static GuiKeyWord kw_288 [3]
• static GuiKeyWord kw_289 [2]
• static GuiKeyWord kw_290 [4]
• static GuiKeyWord kw_291 [7]
• static GuiKeyWord kw_292 [2]
• static GuiKeyWord kw_293 [3]
• static GuiKeyWord kw_294 [4]
• static GuiKeyWord kw_295 [2]
• static GuiKeyWord kw_296 [3]
• static GuiKeyWord kw_297 [3]
• static GuiKeyWord kw_298 [5]
• static GuiKeyWord kw_299 [2]
• static GuiKeyWord kw_300 [3]
• static GuiKeyWord kw_301 [4]
• static GuiKeyWord kw_302 [5]
• static GuiKeyWord kw_303 [3]
• static GuiKeyWord kw_304 [23]
• static GuiKeyWord kw_305 [6]
• static GuiKeyWord kw_306 [3]
• static GuiKeyWord kw_307 [2]
• static GuiKeyWord kw_308 [2]
• static GuiKeyWord kw_309 [5]
• static GuiKeyWord kw_310 [7]
• static GuiKeyWord kw_311 [2]
• static GuiKeyWord kw_312 [3]
• static GuiKeyWord kw_313 [6]
• static GuiKeyWord kw_314 [2]
• static GuiKeyWord kw_315 [6]
• static GuiKeyWord kw_316 [4]
• static GuiKeyWord kw_317 [6]
• static GuiKeyWord kw_318 [8]
• static GuiKeyWord kw_319 [18]
• static GuiKeyWord kw_320 [4]
• static GuiKeyWord kw_321 [10]
• static GuiKeyWord kw_322 [2]
• static GuiKeyWord kw_323 [1]
• static GuiKeyWord kw_324 [2]
• static GuiKeyWord kw_325 [8]
• static GuiKeyWord kw_326 [4]
• static GuiKeyWord kw_327 [6]
• static GuiKeyWord kw_328 [8]
• static GuiKeyWord kw_329 [15]
• static GuiKeyWord kw_330 [4]
• static GuiKeyWord kw_331 [4]
• static GuiKeyWord kw_332 [8]
• static GuiKeyWord kw_333 [7]
• static GuiKeyWord kw_334 [1]
• static GuiKeyWord kw_335 [2]
• static GuiKeyWord kw_336 [19]
• static GuiKeyWord kw_337 [6]
• static GuiKeyWord kw_338 [11]
• static GuiKeyWord kw_339 [5]
• static GuiKeyWord kw_340 [12]
• static GuiKeyWord kw_341 [10]
• static GuiKeyWord kw_342 [8]
• static GuiKeyWord kw_343 [8]
• static GuiKeyWord kw_344 [1]
• static GuiKeyWord kw_345 [7]
• static GuiKeyWord kw_346 [1]
• static GuiKeyWord kw_347 [7]
• static GuiKeyWord kw_348 [7]
• static GuiKeyWord kw_349 [3]
• static GuiKeyWord kw_350 [9]
• static GuiKeyWord kw_351 [8]
• static GuiKeyWord kw_352 [7]
• static GuiKeyWord kw_353 [7]
• static GuiKeyWord kw_354 [6]
• static GuiKeyWord kw_355 [3]
• static GuiKeyWord kw_356 [9]
• static GuiKeyWord kw_357 [9]
• static GuiKeyWord kw_358 [8]
• static GuiKeyWord kw_359 [3]
• static GuiKeyWord kw_360 [5]
• static GuiKeyWord kw_361 [7]
• static GuiKeyWord kw_362 [7]
• static GuiKeyWord kw_363 [4]
• static GuiKeyWord kw_364 [7]
• static GuiKeyWord kw_365 [11]
• static GuiKeyWord kw_366 [6]
• static GuiKeyWord kw_367 [6]
• static GuiKeyWord kw_368 [6]
• static GuiKeyWord kw_369 [3]
• static GuiKeyWord kw_370 [5]
• static GuiKeyWord kw_371 [2]
• static GuiKeyWord kw_372 [4]
• static GuiKeyWord kw_373 [11]
• static GuiKeyWord kw_374 [7]
• static GuiKeyWord kw_375 [5]
• static GuiKeyWord kw_376 [11]
• static GuiKeyWord kw_377 [3]
• static GuiKeyWord kw_378 [9]
• static GuiKeyWord kw_379 [7]
• static GuiKeyWord kw_380 [7]
• static GuiKeyWord kw_381 [34]
• static GuiKeyWord kw_382 [6]
• static KeyWord kw_1 [3]
• static KeyWord kw_2 [3]
• static KeyWord kw_3 [2]
• static KeyWord kw_4 [3]
• static KeyWord kw_5 [3]
• static KeyWord kw_6 [2]
• static KeyWord kw_7 [1]
• static KeyWord kw_8 [1]
• static KeyWord kw_9 [2]
• static KeyWord kw_10 [3]
• static KeyWord kw_11 [5]
• static KeyWord kw_12 [15]
• static KeyWord kw_13 [1]
• static KeyWord kw_14 [4]
• static KeyWord kw_15 [1]
• static KeyWord kw_16 [4]
• static KeyWord kw_17 [1]
• static KeyWord kw_18 [8]
• static KeyWord kw_19 [9]
• static KeyWord kw_20 [12]
• static KeyWord kw_21 [2]
• static KeyWord kw_22 [2]
• static KeyWord kw_23 [3]
• static KeyWord kw_24 [2]
• static KeyWord kw_25 [2]
• static KeyWord kw_26 [9]
• static KeyWord kw_27 [1]
• static KeyWord kw_28 [2]
• static KeyWord kw_29 [2]
• static KeyWord kw_30 [1]
• static KeyWord kw_31 [1]
• static KeyWord kw_32 [2]
• static KeyWord kw_33 [4]
• static KeyWord kw_34 [4]
• static KeyWord kw_35 [3]
• static KeyWord kw_36 [3]
• static KeyWord kw_37 [3]
• static KeyWord kw_38 [3]
• static KeyWord kw_39 [4]
• static KeyWord kw_40 [2]
• static KeyWord kw_41 [3]
• static KeyWord kw_42 [2]
• static KeyWord kw_43 [11]
• static KeyWord kw_44 [9]
• static KeyWord kw_45 [7]
• static KeyWord kw_46 [2]
• static KeyWord kw_47 [11]
• static KeyWord kw_48 [3]
• static KeyWord kw_49 [3]
• static KeyWord kw_50 [3]
• static KeyWord kw_51 [4]
• static KeyWord kw_52 [6]
• static KeyWord kw_53 [2]
• static KeyWord kw_54 [2]
• static KeyWord kw_55 [1]
• static KeyWord kw_56 [5]
• static KeyWord kw_57 [6]
• static KeyWord kw_58 [1]
• static KeyWord kw_59 [2]
• static KeyWord kw_60 [2]
• static KeyWord kw_61 [4]
• static KeyWord kw_62 [2]
• static KeyWord kw_63 [7]
• static KeyWord kw_64 [3]
• static KeyWord kw_65 [3]
• static KeyWord kw_66 [3]
• static KeyWord kw_67 [4]
• static KeyWord kw_68 [4]
• static KeyWord kw_69 [3]
• static KeyWord kw_70 [3]
• static KeyWord kw_71 [3]
• static KeyWord kw_72 [4]
• static KeyWord kw_73 [6]
• static KeyWord kw_74 [2]
• static KeyWord kw_75 [2]
• static KeyWord kw_76 [1]
• static KeyWord kw_77 [5]
• static KeyWord kw_78 [3]
• static KeyWord kw_79 [8]
• static KeyWord kw_80 [4]
• static KeyWord kw_81 [5]
• static KeyWord kw_82 [3]
- static KeyWord kw_83 [2]
- static KeyWord kw_84 [4]
- static KeyWord kw_85 [2]
- static KeyWord kw_86 [8]
- static KeyWord kw_87 [3]
- static KeyWord kw_88 [2]
- static KeyWord kw_89 [3]
- static KeyWord kw_90 [2]
- static KeyWord kw_91 [5]
- static KeyWord kw_92 [4]
- static KeyWord kw_93 [11]
- static KeyWord kw_94 [2]
- static KeyWord kw_95 [3]
- static KeyWord kw_96 [2]
- static KeyWord kw_97 [2]
- static KeyWord kw_98 [12]
- static KeyWord kw_99 [8]
- static KeyWord kw_100 [4]
- static KeyWord kw_101 [3]
- static KeyWord kw_102 [1]
- static KeyWord kw_103 [14]
- static KeyWord kw_104 [7]
- static KeyWord kw_105 [3]
- static KeyWord kw_106 [3]
- static KeyWord kw_107 [2]
- static KeyWord kw_108 [3]
- static KeyWord kw_109 [4]
- static KeyWord kw_110 [7]
- static KeyWord kw_111 [5]
- static KeyWord kw_112 [3]
- static KeyWord kw_113 [1]
- static KeyWord kw_114 [8]
- static KeyWord kw_115 [1]
- static KeyWord kw_116 [11]
- static KeyWord kw_117 [3]
- static KeyWord kw_118 [3]
- static KeyWord kw_119 [3]
- static KeyWord kw_120 [4]
- static KeyWord kw_121 [2]
- static KeyWord kw_122 [3]
- static KeyWord kw_123 [2]
- static KeyWord kw_124 [7]
- static KeyWord kw_125 [2]
- static KeyWord kw_126 [2]
- static KeyWord kw_127 [3]
- static KeyWord kw_128 [2]
• static KeyWord kw_129 [3]
• static KeyWord kw_130 [3]
• static KeyWord kw_131 [2]
• static KeyWord kw_132 [3]
• static KeyWord kw_133 [4]
• static KeyWord kw_134 [5]
• static KeyWord kw_135 [8]
• static KeyWord kw_136 [2]
• static KeyWord kw_137 [3]
• static KeyWord kw_138 [3]
• static KeyWord kw_139 [2]
• static KeyWord kw_140 [3]
• static KeyWord kw_141 [4]
• static KeyWord kw_142 [5]
• static KeyWord kw_143 [7]
• static KeyWord kw_144 [2]
• static KeyWord kw_145 [1]
• static KeyWord kw_146 [1]
• static KeyWord kw_147 [3]
• static KeyWord kw_148 [3]
• static KeyWord kw_149 [3]
• static KeyWord kw_150 [3]
• static KeyWord kw_151 [4]
• static KeyWord kw_152 [2]
• static KeyWord kw_153 [3]
• static KeyWord kw_154 [2]
• static KeyWord kw_155 [2]
• static KeyWord kw_156 [14]
• static KeyWord kw_157 [2]
• static KeyWord kw_158 [3]
• static KeyWord kw_159 [1]
• static KeyWord kw_160 [2]
• static KeyWord kw_161 [1]
• static KeyWord kw_162 [1]
• static KeyWord kw_163 [5]
• static KeyWord kw_164 [1]
• static KeyWord kw_165 [2]
• static KeyWord kw_166 [6]
• static KeyWord kw_167 [2]
• static KeyWord kw_168 [3]
• static KeyWord kw_169 [2]
• static KeyWord kw_170 [8]
• static KeyWord kw_171 [3]
• static KeyWord kw_172 [4]
• static KeyWord kw_173 [3]
• static KeyWord kw_174 [2]
• static KeyWord kw_175 [1]
• static KeyWord kw_176 [1]
• static KeyWord kw_177 [2]
• static KeyWord kw_178 [3]
• static KeyWord kw_179 [2]
• static KeyWord kw_180 [7]
• static KeyWord kw_181 [3]
• static KeyWord kw_182 [5]
• static KeyWord kw_183 [4]
• static KeyWord kw_184 [10]
• static KeyWord kw_185 [1]
• static KeyWord kw_186 [2]
• static KeyWord kw_187 [4]
• static KeyWord kw_188 [2]
• static KeyWord kw_189 [5]
• static KeyWord kw_190 [9]
• static KeyWord kw_191 [2]
• static KeyWord kw_192 [5]
• static KeyWord kw_193 [3]
• static KeyWord kw_194 [1]
• static KeyWord kw_195 [6]
• static KeyWord kw_196 [7]
• static KeyWord kw_197 [3]
• static KeyWord kw_198 [2]
• static KeyWord kw_199 [1]
• static KeyWord kw_200 [3]
• static KeyWord kw_201 [1]
• static KeyWord kw_202 [2]
• static KeyWord kw_203 [4]
• static KeyWord kw_204 [8]
• static KeyWord kw_205 [1]
• static KeyWord kw_206 [1]
• static KeyWord kw_207 [5]
• static KeyWord kw_208 [2]
• static KeyWord kw_209 [5]
• static KeyWord kw_210 [10]
• static KeyWord kw_211 [2]
• static KeyWord kw_212 [2]
• static KeyWord kw_213 [3]
• static KeyWord kw_214 [2]
• static KeyWord kw_215 [6]
• static KeyWord kw_216 [1]
• static KeyWord kw_217 [2]
• static KeyWord kw_218 [4]
• static KeyWord kw_219 [2]
• static KeyWord kw_220 [2]
• static KeyWord kw_221 [1]
• static KeyWord kw_222 [3]
• static KeyWord kw_223 [4]
• static KeyWord kw_224 [1]
• static KeyWord kw_225 [2]
• static KeyWord kw_226 [3]
• static KeyWord kw_227 [1]
• static KeyWord kw_228 [1]
• static KeyWord kw_229 [2]
• static KeyWord kw_230 [2]
• static KeyWord kw_231 [1]
• static KeyWord kw_232 [17]
• static KeyWord kw_233 [3]
• static KeyWord kw_234 [6]
• static KeyWord kw_235 [3]
• static KeyWord kw_236 [3]
• static KeyWord kw_237 [6]
• static KeyWord kw_238 [3]
• static KeyWord kw_239 [2]
• static KeyWord kw_240 [4]
• static KeyWord kw_241 [3]
• static KeyWord kw_242 [2]
• static KeyWord kw_243 [3]
• static KeyWord kw_244 [2]
• static KeyWord kw_245 [25]
• static KeyWord kw_246 [1]
• static KeyWord kw_247 [4]
• static KeyWord kw_248 [1]
• static KeyWord kw_249 [8]
• static KeyWord kw_250 [3]
• static KeyWord kw_251 [3]
• static KeyWord kw_252 [2]
• static KeyWord kw_253 [3]
• static KeyWord kw_254 [2]
• static KeyWord kw_255 [2]
• static KeyWord kw_256 [4]
• static KeyWord kw_257 [2]
• static KeyWord kw_258 [4]
• static KeyWord kw_259 [2]
• static KeyWord kw_260 [23]
• static KeyWord kw_261 [2]
• static KeyWord kw_262 [3]
• static KeyWord kw_263 [4]
• static KeyWord kw_264 [2]
• static KeyWord kw_265 [4]
• static KeyWord kw_266 [3]
• static KeyWord kw_267 [3]
• static KeyWord kw_268 [4]
• static KeyWord kw_269 [7]
• static KeyWord kw_270 [5]
• static KeyWord kw_271 [2]
• static KeyWord kw_272 [1]
• static KeyWord kw_273 [8]
• static KeyWord kw_274 [3]
• static KeyWord kw_275 [5]
• static KeyWord kw_276 [2]
• static KeyWord kw_277 [2]
• static KeyWord kw_278 [2]
• static KeyWord kw_279 [2]
• static KeyWord kw_280 [4]
• static KeyWord kw_281 [6]
• static KeyWord kw_282 [5]
• static KeyWord kw_283 [7]
• static KeyWord kw_284 [2]
• static KeyWord kw_285 [7]
• static KeyWord kw_286 [1]
• static KeyWord kw_287 [4]
• static KeyWord kw_288 [6]
• static KeyWord kw_289 [13]
• static KeyWord kw_290 [4]
• static KeyWord kw_291 [90]
• static KeyWord kw_292 [1]
• static KeyWord kw_293 [2]
• static KeyWord kw_294 [7]
• static KeyWord kw_295 [2]
• static KeyWord kw_296 [1]
• static KeyWord kw_297 [3]
• static KeyWord kw_298 [4]
• static KeyWord kw_299 [6]
• static KeyWord kw_300 [2]
• static KeyWord kw_301 [2]
• static KeyWord kw_302 [3]
• static KeyWord kw_303 [3]
• static KeyWord kw_304 [3]
• static KeyWord kw_305 [2]
• static KeyWord kw_306 [4]
• static KeyWord kw_307 [7]
• static KeyWord kw_308 [2]
• static KeyWord kw_309 [3]
• static KeyWord kw_310 [4]
• static KeyWord kw_311 [2]
• static KeyWord kw_312 [3]
• static KeyWord kw_313 [3]
• static KeyWord kw_314 [5]
• static KeyWord kw_315 [2]
• static KeyWord kw_316 [3]
• static KeyWord kw_317 [4]
• static KeyWord kw_318 [5]
• static KeyWord kw_319 [3]
• static KeyWord kw_320 [23]
• static KeyWord kw_321 [6]
• static KeyWord kw_322 [3]
• static KeyWord kw_323 [1]
• static KeyWord kw_324 [2]
• static KeyWord kw_325 [2]
• static KeyWord kw_326 [5]
• static KeyWord kw_327 [7]
• static KeyWord kw_328 [2]
• static KeyWord kw_329 [3]
• static KeyWord kw_330 [6]
• static KeyWord kw_331 [2]
• static KeyWord kw_332 [6]
• static KeyWord kw_333 [4]
• static KeyWord kw_334 [6]
• static KeyWord kw_335 [8]
• static KeyWord kw_336 [18]
• static KeyWord kw_337 [4]
• static KeyWord kw_338 [8]
• static KeyWord kw_339 [3]
• static KeyWord kw_340 [2]
• static KeyWord kw_341 [1]
• static KeyWord kw_342 [2]
• static KeyWord kw_343 [8]
• static KeyWord kw_344 [4]
• static KeyWord kw_345 [6]
• static KeyWord kw_346 [8]
• static KeyWord kw_347 [15]
• static KeyWord kw_348 [4]
• static KeyWord kw_349 [4]
• static KeyWord kw_350 [7]
• static KeyWord kw_351 [1]
• static KeyWord kw_352 [2]
• static KeyWord kw_353 [19]
• static KeyWord kw_354 [6]
• static KeyWord kw_355 [11]
• static KeyWord kw_356 [5]
• static KeyWord kw_357 [12]
• static KeyWord kw_358 [10]
12.1. DAKOTA NAMESPACE REFERENCE

- static KeyWord kw_359 [8]
- static KeyWord kw_360 [8]
- static KeyWord kw_361 [11]
- static KeyWord kw_362 [7]
- static KeyWord kw_363 [1]
- static KeyWord kw_364 [7]
- static KeyWord kw_365 [7]
- static KeyWord kw_366 [3]
- static KeyWord kw_367 [9]
- static KeyWord kw_368 [8]
- static KeyWord kw_369 [7]
- static KeyWord kw_370 [7]
- static KeyWord kw_371 [6]
- static KeyWord kw_372 [3]
- static KeyWord kw_373 [9]
- static KeyWord kw_374 [9]
- static KeyWord kw_375 [8]
- static KeyWord kw_376 [3]
- static KeyWord kw_377 [5]
- static KeyWord kw_378 [7]
- static KeyWord kw_379 [7]
- static KeyWord kw_380 [4]
- static KeyWord kw_381 [7]
- static KeyWord kw_382 [11]
- static KeyWord kw_383 [6]
- static KeyWord kw_384 [6]
- static KeyWord kw_385 [6]
- static KeyWord kw_386 [3]
- static KeyWord kw_387 [5]
- static KeyWord kw_388 [2]
- static KeyWord kw_389 [4]
- static KeyWord kw_390 [11]
- static KeyWord kw_391 [7]
- static KeyWord kw_392 [5]
- static KeyWord kw_393 [11]
- static KeyWord kw_394 [3]
- static KeyWord kw_395 [9]
- static KeyWord kw_396 [7]
- static KeyWord kw_397 [7]
- static KeyWord kw_398 [34]
- static KeyWord kw_399 [6]

```c
FILE * nидt
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 DAUVLbl [DAUVar_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 VReal 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 VInt 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., DAUVLbl, with the contiguous container in which they are stored.

• static VStr 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_ceui []
• 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_ichck 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.
12.1. DAKOTA NAMESPACE REFERENCE

- static time \( t_{\text{start\_time}} \)
- const char * \( \text{SCI\_FIELD\_NAMES} \) []
- const int \( \text{SCI\_NUMBER\_OF\_FIELDS} \) = 26
- const int \( \text{LARGE\_SCALE} \) = 100

\( a \) (perhaps arbitrary) definition of large scale; choose a large-scale algorithm if \( \text{numVars} \geq \text{LARGE\_SCALE} \)

- const double \( \text{POW\_VAL} \) = 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
CHAPTER 12. NAMESPACE DOCUMENTATION

- 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.i.n.r.o.u.t.l
- 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]
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_tots, 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_tots, const Teuchos::SerialDenseVector<OrdinalType, ScalarType1> & c_vector, const Teuchos::SerialDenseVector<OrdinalType, ScalarType2> & di_vector, const Teuchos::SerialDenseVector<OrdinalType, ScalarType3> & 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.

void symmetric_eigenvalue_decomposition ( const RealSymMatrix & matrix, RealVector & eigenvalues, RealMatrix & eigenvectors )

Computes the eigenvalues and, optionally, eigenvectors of a real symmetric matrix A.
Eigenvalues are returned in ascending order.
References symmetric_eigenvalue_decomposition().
Referred by get_positive_definite_covariance_from_hessian(), and symmetric_eigenvalue_decomposition().

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

Gets the Euclidean distance between x1 and x2
Referred 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().
Referred by getRmax().

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().
Referred 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().
Referred 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::prpResponse, and ParamResponsePair::prpVariables.

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::variables().
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 Model::db_lookup(), ApplicationInterface::duplication_detect(), SurrBasedLocalMinimizer::find_center_approx(), Minimizer::local_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 ( StringArray pos_args, 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 evls. 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 ( StringArray pos_args, 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 ( StringArray pos_args, String print_dest, unsigned short tabular_format )
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::adv(), Variables::adr(), 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::response(), ParamResponsePair::variables(), ParamResponsePair::write_tabular(), and ParamResponsePair::write_tabular_labels().
Referenced by main().

void read_neutral ( StringArray pos_args )
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 abort_handler(), and ParamResponsePair::read_annotated().
Referenced by main().

void repair_restart ( StringArray pos_args, 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::response().
Referenced by main().

void concatenate_restart ( StringArray pos_args )
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, 
  disolver_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_eq_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", "xDLabels", "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[3] [static]
Initial value:
= {
    "eval.id", 8, 0, 2, 0, 49,
    "header", 8, 0, 1, 0, 47,
    "interface.id", 8, 0, 3, 0, 51
}

1480 distinct keywords (plus 205 aliases)

GuiKeyWord kw_2[3] [static]
Initial value:
= {
    "annotated", 8, 0, 1, 0, 43,
    *custom.annotated*, 8, 3, 1, 0, 45, kw_1,
    "freeform", 8, 0, 1, 0, 53
}

GuiKeyWord kw_3[2] [static]
Initial value:
= {
    "input", 11, 3, 1, 0, 41, kw_2,
    "output", 11, 0, 2, 0, 55
}

GuiKeyWord kw_4[3] [static]
Initial value:
= {
    "eval.id", 8, 0, 2, 0, 27,
    "header", 8, 0, 1, 0, 25,
    "interface.id", 8, 0, 3, 0, 29
}

GuiKeyWord kw_5[3] [static]
Initial value:
= {
    "annotated", 8, 0, 1, 0, 21,
    *custom.annotated*, 8, 3, 1, 0, 23, kw_4,
    "freeform", 8, 0, 1, 0, 31
}

GuiKeyWord kw_6[2] [static]
Initial value:
= {
    "input", 11, 0, 1, 0, 17,
    "output", 11, 3, 2, 0, 19, kw_5
}
GuiKeyWord kw_7[1] [static]
Initial value:

= {
    "stop restart", 0x29, 0, 1, 0, 11
}

GuiKeyWord kw_8[1] [static]
Initial value:

= {
    "results output file", 11, 0, 1, 0, 79, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
    EnvCommands.html#EnvOutput
}

GuiKeyWord kw_9[2] [static]
Initial value:

= {
    "input", 11, 0, 1, 0, 35,
    "output", 11, 0, 2, 0, 37
}

GuiKeyWord kw_10[3] [static]
Initial value:

= {
    "eval id", 8, 0, 2, 0, 69,
    "header", 8, 0, 1, 0, 67,
    "interface id", 8, 0, 3, 0, 71
}

GuiKeyWord kw_11[5] [static]
Initial value:

= {
    "annotated", 8, 0, 2, 0, 63,
    "custom annotated", 8, 3, 2, 0, 65, kw_10,
    "freeform", 8, 0, 2, 0, 73,
    "tabular data file", 11, 0, 1, 0, 61,
    "tabular graphics file", 3, 0, 1, 0, 60
}
GuiKeyWord kw_12[15]  [static]

Initial value:
= {
    {"check",8,0,1,0,3},
    {"error_file",11,0,3,0,7},
    {"graphics",8,0,9,0,57,0,0,0,0,0,"{Graphics flag} EnvCommands.html#EnvOutput"},
    {"method_pointer",3,0,13,0,80},
    {"output_file",11,0,2,0,5},
    {"output_precision",0x29,0,11,0,75,0,0,0,0,0,"{Numeric output precision} EnvCommands.html#EnvOutput"},
    {"post_run",8,2,8,0,39,kw_33},
    {"pre_run",8,2,6,0,15,kw_35},
    {"read_restart",11,0,14,0,9,kw_37},
    {"results_output",8,1,12,0,77,kw_8,0,0,0,0,0,"{Enable results output} EnvCommands.html#EnvOutput"},
    {"run",8,2,7,0,33,kw_9},
    {"tabular_data",8,0,10,0,59,kw_11},
    {"tabular_graphics_data",0,5,10,0,58,kw_11},
    {"top_method_pointer",11,0,13,0,81,0,0,0,0,0,0,"{Method pointer} EnvCommands.html#EnvMethPtr"},
    {"write_restart",11,0,5,0,13}
}

GuiKeyWord kw_13[1]  [static]

Initial value:
= {
    {"cache_tolerance",10,0,1,0,2697}
}

GuiKeyWord kw_14[4]  [static]

Initial value:
= {
    {"active_set_vector",8,0,1,0,2691},
    {"evaluation_cache",8,0,2,0,2693},
    {"restart_file",8,0,4,0,2699},
    {"strict_cache_equality",8,1,3,0,2695,kw_13}
}

GuiKeyWord kw_15[1]  [static]

Initial value:
= {
    {"processors_per_analysis",0x19,0,1,0,2667,0,0,0,0,0,0,"{Number of processors per analysis server} InterfCommands.html#InterfApplicDF"}
}

GuiKeyWord kw_16[4]  [static]

Initial value:
= {
    {"abort",8,0,1,1,2691,0,0,0,0,0,"{CHOOSE failure mitigation}"},
    {"continuation",8,0,1,1,2687},
    {"recover",14,0,1,1,2685},
    {"retry",9,0,1,1,2683}
}
GuiKeyWord kw_17[1]  [static]

Initial value:

```c
={
    {"numpy",8,0,1,0,2673,0,0,0,0,0,0,"{Python NumPy dataflow
InterfCommands.html#InterfApplicMSP"},
}
}
```

GuiKeyWord kw_18[8]  [static]

Initial value:

```c
={
    {"copy_files",15,0,5,0,2661,0,0,0,0,0,0,"{copy files} InterfCommands.html#InterfApplicF"},
    {"dir_save",8,0,3,0,2657,0,0,0,0,0,0,"{Save work directory
InterfCommands.html#InterfApplicF"},
    {"directory_save",8,0,3,0,2657,0,0,0,0,0,0,"{Save work directory
InterfCommands.html#InterfApplicF"},
    {"replace",8,0,6,0,2663}
}
```

GuiKeyWord kw_19[9]  [static]

Initial value:

```c
={
    {"allow_existing_results",8,0,3,0,2641,0,0,0,0,0,0,"{Allow existing results files
InterfCommands.html#InterfApplicF"},
    {"aprepro",8,0,5,0,2644,0,0,0,0,0,0,"{Aprepro parameters file format
InterfCommands.html#InterfApplicF"},
    {"file_save",8,0,7,0,2649,0,0,0,0,0,0,"{Parameters and results file saving
InterfCommands.html#InterfApplicF"},
    {"file_tag",8,0,6,0,2647,0,0,0,0,0,0,"{Parameters and results file tagging
InterfCommands.html#InterfApplicF"},
    {"parameters_file",11,0,1,0,2637,0,0,0,0,0,0,"{Parameters file name
InterfCommands.html#InterfApplicF"},
    {"results_file",11,0,2,0,2639,0,0,0,0,0,0,"{Results file name
InterfCommands.html#InterfApplicF"},
    {"verbatim",8,0,4,0,2643,0,0,0,0,0,0,"{Verbatim driver/filter invocation syntax
InterfCommands.html#InterfApplicF"},
    {"work_directory",8,8,8,0,2651,kw_18[0,0,0,0,0,0,"{Create work directory
InterfCommands.html#InterfApplicF"}
}
```

GuiKeyWord kw_20[12]  [static]

Initial value:

```c
={
    {"analysis_components",15,0,1,0,2627,0,0,0,0,0,0,"{Additional identifiers for use by the
analysis_drivers InterfCommands.html#InterfApplic"},
    {"deactivate",8,0,6,0,2689,kw_14[0,0,0,0,0,0,"{Feature deactivation
InterfCommands.html#InterfApplic"},
    {"direct",8,1,4,1,2665,kw_15[0,0,0,0,0,0,"{CHOOSE interface type}{Direct function interface
InterfCommands.html#InterfApplicDF"},
    {"failure_capture",8,4,5,0,2679,kw_16[0,0,0,0,0,0,"{Failure capturing
InterfCommands.html#InterfApplic"},
```
GuiKeyWord kw_21[2] [static]  
Initial value:  
=  
  
\{"master",8,0,1,1,2731},  
\{"peer",8,0,1,1,2733}\}  

GuiKeyWord kw_22[2] [static]  
Initial value:  
=  
  
\{"dynamic",8,0,1,1,2707},  
\{"static",8,0,1,1,2709}\}  

GuiKeyWord kw_23[3] [static]  
Initial value:  
=  
  
\{"analysis\_concurrency",0x19,0,3,0,2711,0,0,0,0,0,"{Asynchronous analysis concurrency}  
InterfCommands.html#InterfIndControl"},  
\{"evaluation\_concurrency",0x19,0,1,0,2703,0,0,0,0,0,"{Asynchronous evaluation concurrency}  
InterfCommands.html#InterfIndControl"},  
\{"local\_evaluation\_scheduling",8,2,2,0,2705,kw_22,0,0,0,0,0,"{Local evaluation scheduling}  
InterfCommands.html#InterfIndControl"\}  

GuiKeyWord kw_24[2] [static]  
Initial value:  
=  
  
\{"dynamic",8,0,1,1,2721},  
\{"static",8,0,1,1,2723\}
GuiKeyWord kw_25[2] [static]
Initial value:
= {
    {"master", 8, 0, 1, 1, 2717},
    {"peer", 8, 2, 1, 1, 2719, kw_24, 0, 0, 0, 0, 0},
    {Peer scheduling of evaluations}
    InterfCommands.html#InterfIndControl*}
}

GuiKeyWord kw_26[9] [static]
Initial value:
= {
    {"algebraic_mappings", 11, 0, 2, 0, 2623, 0, 0, 0, 0, 0, 0},
    {Algebraic mappings file}
    InterfCommands.html#InterfAlgebraic*],
    {"analysis_drivers", 15, 12, 3, 0, 2625, kw_20, 0, 0, 0, 0, 0},
    {Analysis drivers}
    InterfCommands.html#InterfApplic*},
    {"analysis_scheduling", 8, 2, 0, 2729, kw_21, 0, 0, 0, 0, 0, 0},
    {Message passing configuration for scheduling of analyses}
    InterfCommands.html#InterfIndControl*},
    {"analysis_servers", 0x19, 0, 8, 0, 2727, 0, 0, 0, 0, 0, 0},
    {Number of analysis servers}
    InterfCommands.html#InterfIndControl*],
    {"evaluation_scheduling", 8, 2, 6, 0, 2715, kw_25, 0, 0, 0, 0, 0},
    {Message passing configuration for scheduling of evaluations}
    InterfCommands.html#InterfIndControl*],
    {"evaluation_servers", 0x19, 0, 5, 0, 2713, 0, 0, 0, 0, 0, 0},
    {Number of evaluation servers}
    InterfCommands.html#InterfIndControl*],
    {"asynchronous", 8, 3, 4, 0, 2701, kw_23, 0, 0, 0, 0, 0, 0},
    {Asynchronous interface usage}
    InterfCommands.html#InterfIndControl*],
    {\"id_interface\", 11, 0, 1, 0, 2621, 0, 0, 0, 0, 0},
    {Interface set identifier}
    InterfCommands.html#InterfIndControl*],
    {\"processors_per_evaluation\", 0x19, 0, 7, 0, 2725, 0, 0, 0, 0, 0, 0},
    {Number of processors per evaluation server}
    InterfCommands.html#InterfIndControl*]}

GuiKeyWord kw_27[2] [static]
Initial value:
= {
    {"complementary", 8, 0, 1, 1, 1311},
    {"cumulative", 8, 0, 1, 1, 1309}
}

GuiKeyWord kw_28[1] [static]
Initial value:
= {
    {"num_gen_reliability_levels", 13, 0, 1, 0, 1319, 0, 0, 0, 0, 0, 0, 0},
    {Number of generalized reliability levels}
    MethodCommands.html#MethodNonD*}
}

GuiKeyWord kw_29[1] [static]
Initial value:
= {
    {"num_probability_levels", 13, 0, 1, 0, 1315, 0, 0, 0, 0, 0, 0},
    {Number of probability levels}
    MethodCommands.html#MethodNonD*}
GuiKeyWord kw_30[2]  [static]
Initial value:
= 
  
  
GuiKeyWord kw_31[4]  [static]
Initial value:
= 
  
GuiKeyWord kw_32[3]  [static]
Initial value:
= 
  
GuiKeyWord kw_33[3]  [static]
Initial value:
= 
  
GuiKeyWord kw_34[3]  [static]
Initial value:
= 
  
GuiKeyWord kw_35[3]  [static]
Initial value:
= 
  

GuiKeyWord kw_36[4]  [static]
Initial value:
= {
    "active_only", 8, 0, 2, 0, 1177,
    "annotated", 8, 0, 1, 0, 1165,
    "custom.annotated", 8, 3, 1, 0, 1167, kw_35,
    "freeform", 8, 0, 1, 0, 1175
}

GuiKeyWord kw_37[2]  [static]
Initial value:
= {
    "parallel", 8, 0, 1, 1, 1207,
    "series", 8, 0, 1, 1, 1205
}

GuiKeyWord kw_38[3]  [static]
Initial value:
= {
    "gen.reliabilities", 8, 0, 1, 1, 1201,
    "probabilities", 8, 0, 1, 1, 1199,
    "system", 8, 2, 2, 0, 1203, kw_37
}

GuiKeyWord kw_39[2]  [static]
Initial value:
= {
    "compute", 8, 3, 2, 0, 1197, kw_38,
    "num_response_levels", 13, 0, 1, 0, 1195
}

GuiKeyWord kw_40[15]  [static]
Initial value:
= {
    "batch_selection", 8, 4, 3, 0, 1151, kw_31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    "batch_size", 9, 0, 4, 0, 1161, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    "distribution", 8, 2, 12, 0, 1307, kw_27, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    "emulator_samples", 9, 0, 1, 0, 1141, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    "export_points_file", 11, 3, 6, 0, 1179, kw_33, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    "fitness_metric", 8, 3, 2, 0, 1143, kw_34, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    "gen_reliability_levels", 14, 1, 14, 0, 1317, kw_28, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    "import_points_file", 11, 4, 5, 0, 1163, kw_36, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    "misc.options", 15, 0, 8, 0, 1209,
    "model_pointer", 11, 0, 9, 0, 1905,
}
12.1. DAKOTA NAMESPACE REFERENCE

```plaintext
{"probability.levels",14,113,0,1313,kw29,0.,0.,0.,0.,"[Probability levels]"
MethodCommands.html#MethodNonD"},
{"response.levels",14,2,7,0,1193,kw29,0.,0.,0.,0.,"[Response levels]"
MethodCommands.html#MethodNonD"},
{"rng",8,2,15,0,1321,kw30,0.,0.,0.,0.,"[Random number generator]"
MethodCommands.html#MethodNonDMC"},
{"samples",9,0,10,0,1645,0,0.,0.,0.,"[Number of samples]"
MethodCommands.html#MethodNonDMC"},
{"seed",0x19,0,11,0,1647,0,0.,0.,0.,"[Refinement seed]"
MethodCommands.html#MethodNonDLocalRel"}
```

GuiKeyWord kw_41[7] [static]
Initial value:
```plaintext
= {
  {"merit1",8,0,1,1,371,0.,0.,0.,0.,"[CHOOSE merit function]"},
  {"merit1_smooth",8,0,1,1,371},
  {"merit2",8,0,1,1,375},
  {"merit2_smooth",8,0,1,1,377,0.,0.,0.,0.,"@"},
  {"merit2_squared",8,0,1,1,379},
  {"merit_max",8,0,1,1,367},
  {"merit_max_smooth",8,0,1,1,369}
}
```

GuiKeyWord kw_42[2] [static]
Initial value:
```plaintext
= {
  {"blocking",8,0,1,1,361,0.,0.,0.,0.,"[CHOOSE synchronization]"},
  {"nonblocking",8,0,1,1,363,0.,0.,0.,0.,"@"}
}
```

GuiKeyWord kw_43[19] [static]
Initial value:
```plaintext
= {
  {"constraint_penalty",10,0,1,1,381,0.,0.,0.,0.,"[Constraint penalty]"
MethodCommands.html#MethodAPPSDC"},
  {"contraction_factor",10,0,2,0,353,0.,0.,0.,0.,"[Pattern contraction factor]"
MethodCommands.html#MethodAPPSDC"},
  {"initial_delta",10,0,1,0,351,0.,0.,0.,0.,"[Initial offset value]"
MethodCommands.html#MethodAPPSDC"},
  {"linear_equality_constraint_matrix",14,0,15,0,533,0.,0.,0.,0.,0.,"[Linear equality coefficient matrix]"
MethodCommands.html#MethodMin"},
  {"linear_equality_scale_types",15,0,17,0,537,0.,0.,0.,0.,0.,"[Linear equality scaling types]"
MethodCommands.html#MethodMin"},
  {"linear_equality_scales",14,0,18,0,539,0.,0.,0.,0.,0.,"[Linear equality scales]"
MethodCommands.html#MethodMin"},
  {"linear_equality_targets",14,0,16,0,535,0.,0.,0.,0.,0.,"[Linear equality targets]"
MethodCommands.html#MethodMin"},
  {"linear_inequality_constraint_matrix",14,0,10,0,523,0.,0.,0.,0.,0.,"[Linear inequality coefficient matrix]"
MethodCommands.html#MethodMin"},
  {"linear_inequality_lower_bounds",14,0,11,0,525,0.,0.,0.,0.,0.,"[Linear inequality lower bounds]"
MethodCommands.html#MethodMin"},
  {"linear_inequality_scale_types",15,0,13,0,529,0.,0.,0.,0.,0.,"[Linear inequality scaling types]"
MethodCommands.html#MethodMin"},
  {"linear_inequality_scales",14,0,14,0,531,0.,0.,0.,0.,0.,"[Linear inequality scales]"
MethodCommands.html#MethodMin"},
  {"linear_inequality_upper_bounds",14,0,12,0,527,0.,0.,0.,0.,0.,"[Linear inequality upper bounds]"
MethodCommands.html#MethodMin"},
  {"merit_function",8,7,6,0,365,kw_41,0.,0.,0.,0.,"[Merit function]"
MethodCommands.html#MethodAPPSDC"},
```
CHAPTER 12. NAMESPACE DOCUMENTATION

{"model_pointer",11,0,9,0,1905},
{"smoothing_factor",10,0,8,0,383,0,0,.0,.0,.0,"{Smoothing factor} MethodCommands.html#MethodAPPSDC"},
{"solution_accuracy",2,0,4,0,356},
{"solution_target",10,0,4,0,357,0,0,.0,.0,.0,"{Solution target} MethodCommands.html#MethodAPPSDC"},
{"synchronization",6,2,5,0,359,kw,42,0,.0,.0,.0,"{Evaluation synchronization} MethodCommands.html#MethodAPPSDC"},
{"threshold_delta",10,0,3,0,355,0,0,.0,.0,.0,"{Threshold for offset values} MethodCommands.html#MethodAPPSDC"}

GuiKeyWord kw_44[3] [static]
Initial value:

= {
  "eval_id",8,0,2,0,1563},
  "header",8,0,1,0,1561},
  "interface_id",8,0,3,0,1565
}

GuiKeyWord kw_45[3] [static]
Initial value:

= {
  "annotated",8,0,1,0,1557},
  "custom_annotated",8,3,1,0,1559,kw_44},
  "freeform",8,0,1,0,1567
}

GuiKeyWord kw_46[3] [static]
Initial value:

= {
  "active_only",8,0,2,0,1547},
  "annotated",8,0,1,0,1541},
  "interface_id",8,0,3,0,1549
}

GuiKeyWord kw_47[4] [static]
Initial value:

= {
  "active_only",8,0,2,0,1553},
  "annotated",8,0,1,0,1541},
  "custom_annotated",8,3,1,0,1543,kw_46},
  "freeform",8,0,1,0,1551}
GuiKeyWord kw.48[6] [static]
Initial value:

= {
"dakota".8,0,1,1,1533,
"emulator_samples".8,0,2,0,1535,
"export_points_file".11,3,5,0,1555,kw.45,
"import_points_file".11,4,4,0,1539,kw.47,
"posterior_adaptive".8,0,3,6,1537,
"surfpack".8,0,1,1,1531
}

GuiKeyWord kw.49[2] [static]
Initial value:

= {
"collocation_ratio".10,0,1,1,1575,
"posterior_adaptive".8,0,2,0,1577
}

GuiKeyWord kw.50[2] [static]
Initial value:

= {
"expansion_order".13,2,1,1,1573,kw.49,
"sparse_grid_level".13,0,1,1,1571
}

GuiKeyWord kw.51[1] [static]
Initial value:

= {
"sparse_grid_level".13,0,1,1,1581
}

GuiKeyWord kw.52[5] [static]
Initial value:

= {
"gaussian_process".8,6,1,1,1529,kw.48,
"kriging".0,6,1,1,1528,kw.48,
"pce".8,2,1,1,1569,kw.50,
"sc".8,1,1,1,1570,kw.51,
"use_derivatives".8,0,2,0,1583
}
GuiKeyWord kw_53[6] [static]

Initial value:

```c
= {
   {"chains",0x29,0,1,0,1,1517,0,3.,0.,0.,0.,"Number of chains"}
   MethodCommands.html#MethodNonDBayesCalib"},
   {"crossover_chain_pairs",0x29,0,3,0,1521,0,0.,0.,0.,0.,"Number of chain pairs used in crossover"}
   MethodCommands.html#MethodNonDBayesCalib"},
   {"gr_threshold",0x29,0,4,0,1523,0,0.,0.,0.,0.,"Gelman-Rubin Threshold for convergence"}
   MethodCommands.html#MethodNonDBayesCalib"},
   {"jump_step",0x29,5,0,1525,0,0.,0.,0.,0.,"Jump-Step"}
   MethodCommands.html#MethodNonDBayesCalib"},
   {"num_cr",0x29,2,0,1519,0,1.,0.,0.,0.,"Number of candidate points used in burn-in adaptation"} MethodCommands.html#MethodNonDBayesCalib"
}```

GuiKeyWord kw_54[1] [static]

Initial value:

```c
= {
   {"proposal_updates",9,0,1,0,1499}
}
```

GuiKeyWord kw_55[2] [static]

Initial value:

```c
= {
   {"diagonal",8,0,1,1,1511},
   {"matrix",8,0,1,1513}
}
```

GuiKeyWord kw_56[2] [static]

Initial value:

```c
= {
   {"diagonal",8,0,1,1,1505},
   {"matrix",8,0,1,1507}
}
```

GuiKeyWord kw_57[4] [static]

Initial value:

```c
= {
   {"derivatives",8,1,1,1,1497,kw_54},
   {"filename",11,2,1,1,1509,kw_55},
   {"prior",8,0,1,1,1501},
   {"values",14,3,1,1,1503,kw_56}
}
```
GuiKeyWord kw.58[2]  [static]
Initial value:

= {
    {"mt19937", 8, 0, 1, 1491},
    {"rnum2", 8, 0, 1, 1493}
}

GuiKeyWord kw.59[3]  [static]
Initial value:

= {
    {"eval_id", 8, 0, 2, 0, 1473},
    {"header", 8, 0, 1, 0, 1471},
    {"interface_id", 8, 0, 3, 0, 1475}
}

GuiKeyWord kw.60[3]  [static]
Initial value:

= {
    {"annotated", 8, 0, 1, 0, 1467},
    {"custom_annotated", 8, 3, 1, 0, 1469, kw.59},
    {"freeform", 8, 0, 1, 0, 1477}
}

GuiKeyWord kw.61[3]  [static]
Initial value:

= {
    {"eval_id", 8, 0, 2, 0, 1457},
    {"header", 8, 0, 1, 0, 1455},
    {"interface_id", 8, 0, 3, 0, 1459}
}

GuiKeyWord kw.62[4]  [static]
Initial value:

= {
    {"active_only", 8, 0, 2, 0, 1463},
    {"annotated", 8, 0, 1, 0, 1451},
    {"custom_annotated", 8, 3, 1, 0, 1453, kw.61},
    {"freeform", 8, 0, 1, 0, 1461}
}
GuiKeyWord kw_63[10]  [static]

Initial value:

```c
= {
    {"adaptive_metropolis",8,0,4,0,1483},
    {"delayed_rejection",8,0,4,0,1481},
    {"drum",8,0,4,0,1479},
    {"emulator_samples",9,0,1,1,1447},
    {"export_points_file",11,3,3,0,1465,kw_60},
    {"import_points_file",11,4,2,0,1449,kw_62},
    {"metropolis_hastings",8,0,4,0,1485},
    {"multilevel",8,0,4,0,1487},
    {"proposal_covariance",8,4,6,0,1495,kw_57},
    {"rng",8,2,5,0,1489,kw_58,0,0,0,"{Random seed generator}
     MethodCommands.html#MethodNonDBayesCalib"}
};
```

GuiKeyWord kw_64[3]  [static]

Initial value:

```c
= {
    {"eval_id",8,0,2,0,1421},
    {"header",8,0,1,0,1419},
    {"interface_id",8,0,3,0,1423}
};
```

GuiKeyWord kw_65[3]  [static]

Initial value:

```c
= {
    {"annotated",8,0,1,0,1415},
    {"custom_annotated",8,3,1,0,1417,kw_64},
    {"freeform",8,0,1,0,1425}
};
```

GuiKeyWord kw_66[3]  [static]

Initial value:

```c
= {
    {"eval_id",8,0,2,0,1405},
    {"header",8,0,1,0,1403},
    {"interface_id",8,0,3,0,1407}
};
```

GuiKeyWord kw_67[4]  [static]

Initial value:

```c
= {
    {"active_only",8,0,2,0,1411},
    {"annotated",8,0,1,0,1399},
    {"custom_annotated",8,3,1,0,1401,kw_66},
    {"freeform",8,0,1,0,1409}
};
```
GuiKeyWord kw.68[6]  [static]
Initial value:

= {
    {"dakota",8,0,1,1,1391},
    {"emulator_samples",8,0,2,0,1393},
    {"export_points_file",11,3,5,0,1413,kw.65},
    {"import_points_file",11,4,4,0,1397,kw.67},
    {"posterior_adaptive",8,0,3,0,1395},
    {"surfpack",8,0,1,1,1389}
}

GuiKeyWord kw.69[2]  [static]
Initial value:

= {
    {"collocation_ratio",10,0,1,1,1433},
    {"posterior_adaptive",8,0,2,0,1435}
}

GuiKeyWord kw.70[2]  [static]
Initial value:

= {
    {"expansion_order",13,2,1,1,1431,kw.69},
    {"sparse_grid_level",13,0,1,1,1429}
}

GuiKeyWord kw.71[1]  [static]
Initial value:

= {
    {"sparse_grid_level",13,0,1,1,1439}
}

GuiKeyWord kw.72[5]  [static]
Initial value:

= {
    {"gaussian_process",8,6,1,1,1387,kw.68},
    {"kriging",0,6,1,1,1386,kw.68},
    {"pce",8,2,1,1,1427,kw.70},
    {"sc",8,1,1,1,1437,kw.71},
    {"use_derivatives",8,0,2,0,1441}
CHAPTER 12. NAMESPACE DOCUMENTATION

GuiKeyWord kw_73[9]  [static]

Initial value:
= {
  "{adaptive_metropolis","8,0,3,0,1483"},
  "{delayed_rejection","8,0,3,0,1481"},
  "{dra","8,0,3,0,1479"},
  "{emulator","8,5,1,0,1385,kw_72"},
  "{logit_transform","8,0,2,0,1443"},
  "{metropolis_hastings","8,0,3,0,1485"},
  "{multilevel","8,0,3,0,1487"},
  "{proposal_covariance","8,4,5,0,1495,kw_57"},
  "{rng","8,2,4,0,1489,kw_56,0,0,0,0,"Random seed generator MethodCommands.html#MethodNonDBayesCalib"} } 

GuiKeyWord kw_74[9]  [static]

Initial value:
= {
  "{calibrate_sigma","8,0,4,0,1589,0,0,0,0,0,"Calibrate sigma flag MethodCommands.html#MethodNonDBayesCalib"},
  "{dream","8,6,1,1,1515,kw_73"},
  "{gmpsa","8,10,5,1,1445,kw_65"},
  "{likelihood_scale","10,0,3,0,1587,0,0,0,0,0,"Likelihood scale factor MethodCommands.html#MethodNonDBayesCalib"},
  "{model_pointer","11,0,5,0,1905"},
  "{queso","8,9,1,1,1389,kw_73"},
  "{samples","9,0,6,0,1645,0,0,0,0,0,"Number of samples MethodCommands.html#MethodNonOMC"},
  "{seed","8,9,0,7,0,1647,0,0,0,0,0,"[Refinement seed MethodCommands.html#MethodNonDLocalRel"},
  "{standardized_space","8,0,2,0,1585"} }

GuiKeyWord kw_75[4]  [static]

Initial value:
= {
  "{deltas_per_variable","5,0,2,2,1888"},
  "{model_pointer","11,0,3,0,1905"},
  "{step_vector","14,0,1,1,1887,0,0,0,0,0,"Step vector MethodCommands.html#MethodPSCPS"},
  "{steps_per_variable","13,0,2,2,1889,0,0,0,0,0,0,"Number of steps per variable MethodCommands.html#MethodPSCPS"} }

GuiKeyWord kw_76[7]  [static]

Initial value:
= {
  "{beta_solver_name","11,0,1,1,671"},
  "{misc_options","15,0,6,0,679,0,0,0,0,0,0,"Specify miscellaneous options MethodCommands.html#MethodSOLIBDC"},
  "{model_pointer","11,0,2,0,1905"},
  "{seed","8,19,5,4,0,679,0,0,0,0,0,0,"Random seed for stochastic pattern search MethodCommands.html#MethodSOLIBPS"},
  "{show_misc_options","8,0,5,0,677,0,0,0,0,0,0,"Show miscellaneous options MethodCommands.html#MethodSOLIBDC"},
  "{solution_accuracy","2,0,3,0,672"},
  "{solution_target","10,0,3,0,673,0,0,0,0,0,0,"Desired solution target MethodCommands.html#MethodSOLIBDC"} }
GuiKeyWord kw_77[8] [static]

Initial value:

```plaintext
= {
  {"initial_delta",10,0,6,0,589,0,0,0,0,0,*{Initial offset value
MethodCommands.html#MethodSCOLIBPS"},
  {"misc_options",15,0,5,0,679,0,0,0,0,0,*{Specify miscellaneous options
MethodCommands.html#MethodSCOLIBDC"},
  {"model_pointer",11,0,1,0,1905},
  {"seed",0x19,0,3,0,675,0,0,0,0,0,*{Random seed for stochastic pattern search
MethodCommands.html#MethodSCOLIBPS"},
  {"show_misc_options",8,0,4,0,677,0,0,0,0,0,*{Show miscellaneous options
MethodCommands.html#MethodSCOLIBDC"},
  {"solution_accuracy",2,0,2,0,672},
  {"solution_target",10,0,2,0,673,0,0,0,0,0,*{Desired solution target
MethodCommands.html#MethodSCOLIBDC"},
  {"threshold_delta",10,0,7,0,591,0,0,0,0,0,*{Threshold for offset values
MethodCommands.html#MethodSCOLIBPS"}
}
```

GuiKeyWord kw_78[2] [static]

Initial value:

```plaintext
= {
  {"all_dimensions",8,0,1,1,593},
  {"major_dimension",8,0,1,1,597}
}
```

GuiKeyWord kw_79[12] [static]

Initial value:

```plaintext
= {
  {"constraint_penalty",10,0,6,0,609,0,0,0,0,0,*{Constraint penalty
MethodCommands.html#MethodSCOLIBDC"},
  {"division",8,2,1,0,599,kw_78,0,0,0,0,*{Box subdivision approach
MethodCommands.html#MethodSCOLIBDC"},
  {"global_balance_parameter",10,0,2,0,601,0,0,0,0,0,*{Global search balancing parameter
MethodCommands.html#MethodSCOLIBDC"},
  {"local_balance_parameter",10,0,3,0,603,0,0,0,0,0,*{Local search balancing parameter
MethodCommands.html#MethodSCOLIBDC"},
  {"max_boxsize_limit",10,0,4,0,605,0,0,0,0,0,*{Maximum boxsize limit
MethodCommands.html#MethodSCOLIBDC"},
  {"min_boxsize_limit",10,0,5,0,607,0,0,0,0,0,*{Minimum boxsize limit
MethodCommands.html#MethodSCOLIBDC"},
  {"misc_options",15,0,11,0,679,0,0,0,0,0,*{Specify miscellaneous options
MethodCommands.html#MethodSCOLIBDC"},
  {"model_pointer",11,0,7,0,1905},
  {"seed",0x19,0,9,0,675,0,0,0,0,0,*{Random seed for stochastic pattern search
MethodCommands.html#MethodSCOLIBPS"},
  {"show_misc_options",8,0,10,0,677,0,0,0,0,0,*{Show miscellaneous options
MethodCommands.html#MethodSCOLIBDC"},
  {"solution_accuracy",2,0,8,0,672},
  {"solution_target",10,0,8,0,673,0,0,0,0,0,*{Desired solution target
MethodCommands.html#MethodSCOLIBDC"}
}
```

GuiKeyWord kw_80[3] [static]

Initial value:

```plaintext
= {
  {"blend",8,0,1,1,645},
  {"two_point",8,0,1,1,643},
  {"uniform",8,0,1,1,647}
}
```
GuiKeyWord kw_81[2] [static]
Initial value:
= {
    "linear_rank",8,0,1,1,625,
    "merit_function",8,0,1,1,627
}

GuiKeyWord kw_82[3] [static]
Initial value:
= {
    "flat_file",11,0,1,1,621,
    "simple_random",8,0,1,1,617,
    "unique_random",8,0,1,1,619
}

GuiKeyWord kw_83[2] [static]
Initial value:
= {
    "mutation_range",9,0,2,0,663,0,0,0,0,0,0,"{Mutation range}
    MethodCommands.html#MethodSCOLIBEA",
    "mutation_scale",10,0,1,0,661,0,0,0,0,0,"{Mutation scale}
    MethodCommands.html#MethodSCOLIBEA"
}

GuiKeyWord kw_84[5] [static]
Initial value:
= {
    "non_adaptive",8,0,2,0,665,0,0,0,0,0,"{Non-adaptive mutation flag}
    MethodCommands.html#MethodSCOLIBEA",
    "offset_cauchy",8,2,1,1,657,kw_83,
    "offset_normal",8,2,1,1,655,kw_83,
    "offset_uniform",8,2,1,1,659,kw_83,
    "replace_uniform",8,0,1,1,653
}

GuiKeyWord kw_85[4] [static]
Initial value:
= {
    "chc",9,0,1,1,633,0,0,0,0,0,"{CHC replacement type} MethodCommands.html#MethodSCOLIBEA"
},
    "elitist",9,0,1,1,635,0,0,0,0,0,"{Elitist replacement type}
    MethodCommands.html#MethodSCOLIBEA",
    "new_solutions_generated",9,0,2,0,637,0,0,0,0,0,"{New solutions generated}
    MethodCommands.html#MethodSCOLIBEA",
    "random",9,0,1,1,631,0,0,0,0,0,"{Random replacement type}
    MethodCommands.html#MethodSCOLIBEA"}
GuiKeyWord kw_86[15]  [static]

Initial value:

= {
    \{"constraint\_penalty",10,0,9,0,667\},
    \{"crossover\_rate",10,0,5,0,639,0,0,0,0,0,0,0,0,0,0\},
    \{"crossover\_type",8,1,6,0,641,kw_80,0,0,0,0,0,0,0,0,0,0\},
    \{"fitness\_type",8,2,3,0,623,kw_81,0,0,0,0,0,0,0,0,0,0\},
    \{"initialization\_type",8,3,2,0,615,kw_82,0,0,0,0,0,0,0,0,0,0\},
    \{"misc\_options",15,0,14,0,679,0,0,0,0,0,0,0,0,0,0,0,0,0\},
    \{"model\_pointer",11,0,10,0,1905\},
    \{"mutation\_rate",10,0,7,0,649,0,0,0,0,0,0,0,0,0,0,0,0,0\},
    \{"mutation\_type",8,5,6,0,651,kw_84,0,0,0,0,0,0,0,0,0,0\},
    \{"population\_size",0x19,0,1,0,613,0,0,0,0,0,0,0,0,0,0,0,0,0\},
    \{"replacement\_type",8,4,4,0,629,kw_85,0,0,0,0,0,0,0,0,0,0\},
    \{"seed",0x19,0,12,0,675,0,0,0,0,0,0,0,0,0,0,0,0,0\},
    \{"solution\_accuracy",2,0,11,0,672\},
    \{"solution\_target",10,0,11,0,673,0,0,0,0,0,0,0,0,0,0,0,0,0\}
}
GuiKeyWord kw.90[18] [static]

Initial value:

```c
= {
    "constant_penalty",8,0,1,0,543,0,0,0,0,0,0,"[Control of dynamic penalty]
MethodCommands.html#MethodSCOLIBPS",
    "constraint_penalty",10,0,17,0,585,0,0,0,0,0,0,"[Constraint penalty]
MethodCommands.html#MethodSCOLIBPS",
    "contraction_factor",10,0,16,0,583,0,0,0,0,0,0,"[Pattern contraction factor]
MethodCommands.html#MethodSCOLIBPS",
    "expand_after_success",9,0,3,0,547,0,0,0,0,0,0,"{Number of consecutive improvements
before expansion} MethodCommands.html#MethodSCOLIBPS",
    "exploratory_moves",8,3,7,0,559,kw,ось,0,0,0,0,"{Exploratory moves selection}
MethodCommands.html#MethodSCOLIBPS",
    "initial_delta",10,0,14,0,589,0,0,0,0,0,0,"{Initial offset value}
MethodCommands.html#MethodSCOLIBPS",
    "misc_options",15,0,13,0,679,0,0,0,0,0,0,"{Specify miscellaneous options}
MethodCommands.html#MethodSCOLIBDC",
    "model_pointer",11,0,9,0,1905,
    "no_expansion",8,0,2,0,545,0,0,0,0,0,0,"{No expansion flag}
MethodCommands.html#MethodSCOLIBPS",
    "pattern_base",8,2,4,0,549,kw,88,0,0,0,0,"{Pattern basis selection}
MethodCommands.html#MethodSCOLIBPS",
    "seed",0x19,0,11,0,675,0,0,0,0,0,0,"{Random seed for stochastic pattern search}
MethodCommands.html#MethodSCOLIBPS",
    "show_misc_options",8,0,12,0,677,0,0,0,0,0,0,"{Show miscellaneous options}
MethodCommands.html#MethodSCOLIBDC",
    "solution_accuracy",2,0,10,0,672,
    "solution_target",10,0,10,0,673,0,0,0,0,0,0,"{Desired solution target}
MethodCommands.html#MethodSCOLIBDC",
    "stochastic",8,0,5,0,555,0,0,0,0,0,0,"{Stochastic pattern search}
MethodCommands.html#MethodSCOLIBPS",
    "synchronization",8,2,8,0,567,kw,89,0,0,0,0,0,"{Evaluation synchronization}
MethodCommands.html#MethodSCOLIBPS",
    "threshold_delta",10,0,15,0,591,0,0,0,0,0,0,"{Threshold for offset values}
MethodCommands.html#MethodSCOLIBPS",
    "total_pattern_size",9,0,6,0,557,0,0,0,0,0,0,"{Total number of points in pattern}
MethodCommands.html#MethodSCOLIBPS"
}
```

GuiKeyWord kw.91[14] [static]

Initial value:

```c
= {
    "constant_penalty",8,0,4,0,581,0,0,0,0,0,0,"[Control of dynamic penalty]
MethodCommands.html#MethodSCOLIBPS",
    "constraint_penalty",10,0,13,0,585,0,0,0,0,0,0,"[Constraint penalty]
MethodCommands.html#MethodSCOLIBPS",
    "contraction_factor",10,0,12,0,583,0,0,0,0,0,0,"{Pattern contraction factor}
MethodCommands.html#MethodSCOLIBPS",
    "expand_after_success",9,0,3,0,579,0,0,0,0,0,0,"{Number of consecutive improvements
before expansion} MethodCommands.html#MethodSCOLIBPS",
    "initial_delta",10,0,10,0,589,0,0,0,0,0,0,"{Initial offset value}
MethodCommands.html#MethodSCOLIBPS",
    "model_pointer",11,0,5,0,1905,
    "no_expansion",8,0,2,0,577,0,0,0,0,0,0,"{No expansion flag}
MethodCommands.html#MethodSCOLIBPS",
    "seed",0x19,0,7,0,675,0,0,0,0,0,0,"{Random seed for stochastic pattern search}
MethodCommands.html#MethodSCOLIBPS",
    "show_misc_options",8,0,8,0,677,0,0,0,0,0,0,"{Show miscellaneous options}
MethodCommands.html#MethodSCOLIBDC",
    "solution_accuracy",2,0,6,0,672,
    "solution_target",10,0,6,0,673,0,0,0,0,0,0,"{Desired solution target}
MethodCommands.html#MethodSCOLIBPS",
    "threshold_delta",10,0,11,0,591,0,0,0,0,0,0,"{Threshold for offset values}
MethodCommands.html#MethodSCOLIBPS"
```
GuiKeyWord kw_92 [static]

Initial value:

= {
  {"frcg", 8, 0, 1, 1285},
  {"linear_equality_constraint_matrix", 14, 0, 8, 0, 533, 0, 0, 0, 0},
  {"linear_equality_coefficient_matrix" MethodCommands.html#MethodMin"},
  {"linear_equality_scaling_types", 15, 0, 10, 0, 537, 0, 0, 0, 0},
  {"linear_equality_scales", 14, 0, 11, 0, 539, 0, 0, 0, 0},
  {"linear_equality_target", MethodCommands.html#MethodMin"},
  {"linear_inequality_constraint_matrix", 14, 0, 3, 0, 523, 0, 0, 0, 0},
  {"linear_inequality_coefficient_matrix" MethodCommands.html#MethodMin"},
  {"linear_inequality_lower_bounds", 14, 0, 4, 0, 525, 0, 0, 0, 0},
  {"linear_inequality_scale_types", 15, 0, 6, 0, 529, 0, 0, 0, 0},
  {"linear_inequality_scaling_types" MethodCommands.html#MethodMin"},
  {"linear_inequality_scales", 14, 0, 7, 0, 531, 0, 0, 0, 0},
  {"linear_inequality_upper_bounds", 14, 0, 5, 0, 527, 0, 0, 0, 0},
  {"mfd", 8, 0, 1, 1287},
  {"model_pointer", 11, 0, 2, 0, 1905}
}

GuiKeyWord kw_93 [static]

Initial value:

= {
  {"linear_equality_constraint_matrix", 14, 0, 7, 0, 533, 0, 0, 0, 0},
  {"linear_equality_coefficient_matrix" MethodCommands.html#MethodMin"},
  {"linear_equality_scaling_types", 15, 0, 9, 0, 537, 0, 0, 0, 0},
  {"linear_equality_scales", 14, 0, 10, 0, 539, 0, 0, 0, 0},
  {"linear_equality_target", MethodCommands.html#MethodMin"},
  {"linear_inequality_constraint_matrix", 14, 0, 2, 0, 523, 0, 0, 0, 0},
  {"linear_inequality_coefficient_matrix" MethodCommands.html#MethodMin"},
  {"linear_inequality_lower_bounds", 14, 0, 3, 0, 525, 0, 0, 0, 0},
  {"linear_inequality_scale_types", 15, 0, 5, 0, 529, 0, 0, 0, 0},
  {"linear_inequality_scaling_types" MethodCommands.html#MethodMin"},
  {"linear_inequality_scales", 14, 0, 6, 0, 531, 0, 0, 0, 0},
  {"linear_inequality_upper_bounds", 14, 0, 4, 0, 527, 0, 0, 0, 0},
  {"model_pointer", 11, 0, 1, 0, 1905}
}

GuiKeyWord kw_94 [static]

Initial value:

= {
  {"drop_tolerance", 10, 0, 1, 1613}
}
GuiKeyWord kw_95[15] [static]
Initial value:

```cpp
= {
    {"box_behnken",8,0,1,1,1603,0,0,0,0,0,0,"[CHOOSE DACE type]",
     {"central_composite",8,0,1,1,1605},
     {"fixed_seed",8,0,5,1605,0,0,0,0,0,0,"[Fixed seed flag] MethodCommands.html#MethodDDACE"},
    {"main_effects",8,0,1,1,1601,0,0,0,0,0,0,"[Main effects] MethodCommands.html#MethodDDACE"},
    {"model_pointer",11,0,7,0,1905,
     {"oa_lhs",8,0,1,1,1601},
     {"oas",8,0,1,1,1597},
     {quality_metrics",8,0,3,0,1609,0,0,0,0,0,0,"[Quality metrics] MethodCommands.html#MethodDDACE"},
    {"random",8,0,1,1,1595},
    {"samples",9,0,8,0,1645,0,0,0,0,0,0,"[Number of samples] MethodCommands.html#MethodNonDMonaco"},
    {"seed",0x19,0,9,0,1647,0,0,0,0,0,0,"[Refinement seed] MethodCommands.html#MethodNonDLocalRel"},
    {"symbols",9,0,6,0,1617,0,0,0,0,0,0,"[Number of symbols] MethodCommands.html#MethodDDACE"},
    {"variance_based_decomp",8,1,4,0,1611,kw_94,0,0,0,0,0,0,"[Variance based decomposition] MethodCommands.html#MethodDDACE"}
}

GuiKeyWord kw_96[15] [static]
Initial value:

```cpp
= {
    {"bfgs",8,0,1,1,273},
    {"fgp",8,0,1,1,269},
    {linear_equality_constraint_matrix",14,0,8,0,533,0,0,0,0,0,0,"[Linear equality coefficient matrix] MethodCommands.html#MethodMin"},
    {linear_equality_scale_types",15,0,10,0,537,0,0,0,0,0,0,"[Linear equality scaling types] MethodCommands.html#MethodMin"},
    {linear_equality_scales",14,0,11,0,539,0,0,0,0,0,0,"[Linear equality scales] MethodCommands.html#MethodMin"},
    {linear_equality_targets",14,0,9,0,535,0,0,0,0,0,0,"[Linear equality targets] MethodCommands.html#MethodMin"},
    {linear_inequality_constraint_matrix",14,0,3,0,523,0,0,0,0,0,0,"[Linear inequality coefficient matrix] MethodCommands.html#MethodMin"},
    {linear_inequality_lower_bounds",14,0,4,0,525,0,0,0,0,0,0,"[Linear inequality lower bounds] MethodCommands.html#MethodMin"},
    {linear_inequality_scale_types",15,0,6,0,529,0,0,0,0,0,0,"[Linear inequality scaling types] MethodCommands.html#MethodMin"},
    {linear_inequality_scales",14,0,7,0,531,0,0,0,0,0,0,"[Linear inequality scales] MethodCommands.html#MethodMin"},
    {linear_inequality_upper_bounds",14,0,5,0,527,0,0,0,0,0,0,"[Linear inequality upper bounds] MethodCommands.html#MethodMin"},
    {"mmfd",8,0,1,1,271},
    {"model_pointer",11,0,2,0,1905},
    {"slp",8,0,1,1,275},
    {"sqp",8,0,1,1,277}
}

GuiKeyWord kw_97[3] [static]
Initial value:

```cpp
= {
    {"eval_id",8,0,2,0,753},
    {"header",8,0,1,0,751},
    {"interface_id",8,0,3,0,755}
}
```
GuiKeyWord kw_98[3] [static]
Initial value:

```c
= {
    {"annotated", 8, 0, 1, 0, 747},
    {"custom.annotated", 8, 3, 1, 0, 749, kw_97},
    {"freeform", 8, 0, 1, 0, 757}
}
```

GuiKeyWord kw_99[2] [static]
Initial value:

```c
= {
    {"dakota", 8, 0, 1, 1, 725},
    {"surfpack", 8, 0, 1, 1, 723}
}
```

GuiKeyWord kw_100[3] [static]
Initial value:

```c
= {
    {"eval_id", 8, 0, 2, 0, 737},
    {"header", 8, 0, 1, 0, 733},
    {"interface_id", 8, 0, 3, 0, 739}
}
```

GuiKeyWord kw_101[4] [static]
Initial value:

```c
= {
    {"active_only", 8, 0, 2, 0, 743},
    {"annotated", 8, 0, 1, 0, 731},
    {"custom.annotated", 8, 3, 1, 0, 733, kw_100},
    {"freeform", 8, 0, 1, 0, 741}
}
```

GuiKeyWord kw_102[7] [static]
Initial value:

```c
= {
    {"export.points_file", 11, 3, 4, 0, 745, kw_98, 0, 0, 0, 0, "[File name for exporting approximation-based samples from evaluating the GP] MethodCommands.html#MethodEG"},
    {"gaussian.process", 8, 2, 1, 0, 721, kw_99, 0, 0, 0, 0, "[GP selection] MethodCommands.html#MethodEG"},
    {"import.points_file", 11, 4, 3, 0, 729, kw_101, 0, 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, 720, kw_99},
    {"model_pointer", 11, 0, 6, 0, 1905},
    {"seed", 0x19, 0, 5, 0, 759, 0, 0, 0, 0, 0, "[Random seed] MethodCommands.html#MethodEG"},
    {"use_derivatives", 8, 0, 2, 0, 727, 0, 0, 0, 0, 0, "[Derivative usage] MethodCommands.html#MethodEG"}
}
```
GuiKeyWord kw_103[9]  [static]
Initial value:
= {
   {"batch_size",9,0,2,0,1241},
   {"distribution",8,2,6,0,1307,kw_103,0,0,0,0,"Distribution type MethodCommands.html#MethodNonD"},
   {"emulator_samples",9,0,1,0,1239},
   {{"gen_reliability_levels",14,1,8,0,1317,kw_103,0,0,0,0,"{Generalized reliability levels MethodCommands.html#MethodNonD"}},
   {{"model_pointer",11,0,2,0,1905},
   {"probability_levels",14,1,7,0,1313,kw_103,0,0,0,0,"{Probability levels MethodCommands.html#MethodNonD"}},
   {"rng",8,2,9,0,1321,kw_103,0,0,0,0,"{Random number generator MethodCommands.html#MethodNonD"}},
   {"samples",9,0,4,0,1645,0,0,0,0,0,"{Number of samples MethodCommands.html#MethodNonDMC"}},
   {"seed",0x19,0,5,0,1647,0,0,0,0,0,"{Refinement seed MethodCommands.html#MethodNonDLocalRel"}}
}

GuiKeyWord kw_104[3]  [static]
Initial value:
= {
   {"grid",8,0,1,1,1633,0,0,0,0,0,*"{CHOOSE trial type}"},
   {"halton",8,0,1,1,1635},
   {"random",8,0,1,1,1637,0,0,0,0,0,*"@"}
}

GuiKeyWord kw_105[1]  [static]
Initial value:
= {
   {"drop_tolerance",10,0,1,0,1627}
}

GuiKeyWord kw_106[9]  [static]
Initial value:
= {
   {"fixed_seed",8,0,4,0,1629,0,0,0,0,0,"{Fixed seed flag MethodCommands.html#MethodFSUDace"}},
   {"latinize",8,0,1,0,1621,0,0,0,0,0,"{Latinization of samples MethodCommands.html#MethodFSUDace"}},
   {"model_pointer",11,0,7,0,1985},
   {"num_trials",9,0,4,0,1633,0,0,0,0,0,"{Number of trials MethodCommands.html#MethodFSUDace"}},
   {"samples",9,5,8,0,1645,0,0,0,0,0,"{Number of samples MethodCommands.html#MethodNonDMC"}},
   {"seed",0x19,0,9,0,1647,0,0,0,0,0,"{Refinement seed MethodCommands.html#MethodNonDLocalRel"}},
   {"trial_type",8,3,5,0,1631,kw_104,0,0,0,0,"{Trial type MethodCommands.html#MethodFSUDace"}},
   {"variance_based_decomp",8,1,3,0,1625,kw_105,0,0,0,0,"{Variance based decomposition MethodCommands.html#MethodFSUDace"}}
}
GuiKeyWord kw_107[1]  [static]
Initial value:
= {
    {"drop_tolerance",10,0,1,0,1845}
}

GuiKeyWord kw_108[11]  [static]
Initial value:
= {
    {"fixed_sequence",8,0,6,0,1849,0,0,0,0,0,"[Fixed sequence flag]
MethodCommands.html#MethodFSUDACE"},
    {"halton",8,0,1,1,1853,0,0,0,0,0,"[CHOOSE sequence type]"},
    {"hammersley",8,0,1,1,1837},
    {"latinize",8,0,2,0,1839,0,0,0,0,0,"[Latinization of samples]
MethodCommands.html#MethodFSUDACE"},
    {"model_pointer",11,0,10,0,1905},
    {"prime_base",13,0,9,0,1855,0,0,0,0,0,"[Prime bases for sequences]
MethodCommands.html#MethodFSUDACE"},
    {"quality_metrics",8,0,3,0,1841,0,0,0,0,0,"[Quality metrics]
MethodCommands.html#MethodFSUDACE"},
    {"samples",9,0,5,0,1847,0,0,0,0,0,"[Number of samples taken in the MCMC sampling]
MethodCommands.html#MethodNonDBayesCalib"},
    {"sequence_leap",13,0,8,0,1853,0,0,0,0,0,"[Sequence leaping indices]
MethodCommands.html#MethodFSUDACE"},
    {"sequence_start",13,0,7,0,1851,0,0,0,0,0,"[Sequence starting indices]
MethodCommands.html#MethodFSUDACE"},
    {"variance_based_decomp",8,1,4,0,1843,kw_107,0,0,0,0,0,"[Variance based decomposition]
MethodCommands.html#MethodFSUDACE"}
}

GuiKeyWord kw_109[3]  [static]
Initial value:
= {
    {"eval_id",8,0,2,0,1117},
    {"header",8,0,1,0,1115},
    {"interface_id",8,0,3,0,1119}
}

GuiKeyWord kw_110[3]  [static]
Initial value:
= {
    {"annotated",8,0,1,0,1111},
    {"custom_annotated",8,3,1,0,1113,kw_109},
    {"freeform",8,0,1,0,1121}
}

GuiKeyWord kw_111[3]  [static]
Initial value:
= {
    {"eval_id",8,0,2,0,1101},
    {"header",8,0,1,0,1099},
    {"interface_id",8,0,3,0,1103}
}
GuiKeyWord kw_112[4] [static]
Initial value:
= {
  "active_only", 8, 0, 2, 0, 1107,
  "annotated", 8, 0, 1, 0, 1095,
  "custom_annotated", 8, 3, 1, 0, 1097, kw_111,
  "freeform", 8, 0, 1, 0, 1105
}

GuiKeyWord kw_113[2] [static]
Initial value:
= {
  "parallel", 8, 0, 1, 1, 1137,
  "series", 8, 0, 1, 1, 1135
}

GuiKeyWord kw_114[3] [static]
Initial value:
= {
  "gen_reliabilities", 8, 0, 1, 1, 1131,
  "probabilities", 8, 0, 1, 1, 1129,
  "system", 8, 2, 2, 0, 1133, kw_113
}

GuiKeyWord kw_115[2] [static]
Initial value:
= {
  "compute", 8, 3, 2, 0, 1127, kw_114,
  "num_response_levels", 13, 0, 1, 0, 1125
}

GuiKeyWord kw_116[11] [static]
Initial value:
= {
  "distribution", 8, 2, 8, 0, 1307, kw_27, 0, 0, 0, 0, "Distribution type",
  MethodCommands.html#MethodNonDNon*,
  "emulator_samples", 9, 0, 1, 0, 1091,
  "export_points_file", 11, 3, 3, 0, 1109, kw_110, 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, 1317, kw_28, 0, 0, 0, 0, "Generalized reliability levels",
  MethodCommands.html#MethodNonDNon*,
  "import_points_file", 11, 4, 2, 0, 1093, kw_112, 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, 1905,
  "probability_levels", 14, 1, 9, 0, 1313, kw_29, 0, 0, 0, 0, "Probability levels",
  MethodCommands.html#MethodNonDNon*,
  "response_levels", 14, 2, 4, 0, 1123, kw_115,
  "rng", 8, 2, 11, 0, 1321, kw_30, 0, 0, 0, 0, "Random number generator",
  MethodCommands.html#MethodNonDMC",
  "samples", 9, 5, 6, 0, 1645, 0, 0, 0, 0, 0, "Number of samples MethodCommands.html#MethodNonDMC",
  "seed", 0x19, 0, 7, 0, 1647, 0, 0, 0, 0, 0, "Refinement seed",
  MethodCommands.html#MethodNonDLocalRel*}
GuiKeyWord kw_117[2] [static]  
Initial value:

```c
= {
    {"model_pointer",11,0,2,0,1905},
    {"seed",0x19,0,1,0,717,0,0.,0.,0.,0.,0.,0.,0.,0.},{Random seed} MethodCommands.html#MethodNonDMC"
}
```

GuiKeyWord kw_118[2] [static]  
Initial value:

```c
= {
    {"parallel",8,0,1,1,1305},
    {"series",8,0,1,1,1303}
}
```

GuiKeyWord kw_119[3] [static]  
Initial value:

```c
= {
    {"gen_reliabilities",8,0,1,1,1299},
    {"probabilities",8,0,1,1,1297},
    {"system",8,2,2,0,1301,kw_118}
}
```

GuiKeyWord kw_120[2] [static]  
Initial value:

```c
= {
    {"compute",8,3,2,0,1295,kw_119},
    {"num_response_levels",13,0,1,0,1293}
}
```

GuiKeyWord kw_121[3] [static]  
Initial value:

```c
= {
    {"eval_id",8,0,2,0,1281},
    {"header",8,0,1,0,1279},
    {"interface_id",8,0,3,0,1283}
}
```

GuiKeyWord kw_122[3] [static]  
Initial value:

```c
= {
    {"annotated",8,0,1,0,1275},
    {"custom_annotated",8,3,1,0,1277,kw_121},
    {"freeform",9,0,1,0,1285}
}
```
GuiKeyWord kw_123[2] [static]
Initial value:
= {
    {"dakota",8,0,1,1,1253},
    {"surfpack",8,0,1,1,1251}
}

GuiKeyWord kw_124[3] [static]
Initial value:
= {
    {"eval_id",8,0,2,0,1265},
    {"header",8,0,1,0,1263},
    {"interface_id",8,0,3,0,1267}
}

GuiKeyWord kw_125[4] [static]
Initial value:
= {
    {"active_only",8,0,2,0,1271},
    {"annotated",8,0,1,0,1259},
    {"custom.annotated",8,3,1,0,1261,kw_124},
    {"freeform",8,0,1,0,1269}
}

GuiKeyWord kw_126[5] [static]
Initial value:
= {
    {"export_points_file",11,3,4,0,1273,kw_122},
    {"gaussian_process",8,0,1,0,1249,kw_123},
    {"import_points_file",11,4,3,0,1257,kw_125,0.,0.,0.,0,"{File containing points to evaluate MethodCommands.html#MethodPSLPS"},
    {"kriging",0,2,1,0,1248,kw_123},
    {"use_derivatives",8,0,2,0,1255}
}

GuiKeyWord kw_127[12] [static]
Initial value:
= {
    {"distribution",8,2,6,0,1307,kw_27,0.,0.,0.,0,"{Distribution type MethodCommands.html#MethodNonD"},
    {"ea",8,0,1,0,1287},
    {"ego",8,5,1,0,1247,kw_126},
    {"gen_reliability_levels",14,1,8,0,1317,kw_28,8,0.,0.,0.,0,"{Generalized reliability levels MethodCommands.html#MethodNonD"},
    {"lhs",8,0,1,0,1289},
    {"model_pointer",11,0,3,0,1905},
    {"probability_levels",14,1,7,0,1313,kw_23,0.,0.,0.,0,"{Probability levels MethodCommands.html#MethodNonD"},
    {"response_levels",14,2,2,0,1291,kw_122},
    {"rng",8,2,9,0,1321,kw_30,0.,0.,0.,0,"{Random number generator MethodCommands.html#MethodNonDMC"},
    {"samples",9,5,4,0,1645,0.,0.,0.,0.,0,"{Number of samples MethodCommands.html#MethodNonDMC"}
GuiKeyWord kw_128[2] [static]

Initial value:

= {
    {"mt19937", 8, 0, 1, 1, 1377},
    {"rnum2", 8, 0, 1, 1, 1379}
}

GuiKeyWord kw_129[3] [static]

Initial value:

= {
    {"eval.id", 8, 0, 2, 0, 1365},
    {"header", 8, 0, 1, 0, 1363},
    {"interface.id", 8, 0, 3, 0, 1367}
}

GuiKeyWord kw_130[3] [static]

Initial value:

= {
    {"annotated", 8, 0, 1, 0, 1359},
    {"custom.annotated", 8, 3, 1, 0, 1361, kw_129},
    {"freeform", 8, 0, 1, 0, 1369}
}

GuiKeyWord kw_131[2] [static]

Initial value:

= {
    {"dakota", 8, 0, 1, 1, 1337},
    {"surfpack", 8, 0, 1, 1, 1335}
}

GuiKeyWord kw_132[3] [static]

Initial value:

= {
    {"eval.id", 8, 0, 2, 0, 1349},
    {"header", 8, 0, 1, 0, 1347},
    {"interface.id", 8, 0, 3, 0, 1351}
}
GuiKeyWord kw_133[4] [static]

Initial value:

```cpp
= {
    "active_only", 8, 0, 2, 0, 1355,
    "annotated", 8, 0, 1, 0, 1343,
    "custom_annotated", 8, 3, 1, 0, 1345, kw_132,
    "freeform", 8, 0, 1, 0, 1353
}
```

GuiKeyWord kw_134[5] [static]

Initial value:

```cpp
= {
    "export_points_file", 11, 3, 4, 0, 1357, kw_130, 0., 0., 0., "{File name for exporting approximation-based samples from evaluating the GP} MethodCommands.html#MethodNonDGlobalIntervalEst"},
    "gaussian_process", 8, 2, 1, 0, 1333, kw_131, 0., 0., 0., "{EGO GP selection} MethodCommands.html#MethodNonDGlobalIntervalEst"),
    "import_points_file", 11, 3, 5, 0, 1341, kw_133, 0., 0., 0., "{File name for points to be imported as the basis for the initial GP} MethodCommands.html#MethodNonDGlobalIntervalEst"),
    "kriging", 8, 2, 1, 0, 1332, kw_131,
    "use_derivatives", 8, 0, 2, 0, 1339, 0., 0., 0., "{Derivative usage} MethodCommands.html#MethodNonDGlobalIntervalEst"}
```

GuiKeyWord kw_135[8] [static]

Initial value:

```cpp
= {
    "ea", 8, 0, 1, 0, 1371,
    "ego", 8, 5, 1, 0, 1331, kw_134,
    "lns", 8, 0, 1, 0, 1373,
    "model_pointer", 11, 0, 3, 0, 1905,
    "rng", 8, 2, 2, 0, 1375, kw_128, 0., 0., 0., "{Random seed generator} MethodCommands.html#MethodNonDGlobalIntervalEst"),
    "samples", 9, 0, 4, 0, 1645, 0., 0, 0., "{Number of samples} MethodCommands.html#MethodNonDMC" },
    "sbo", 8, 5, 1, 0, 1329, kw_134,
    "seed", 0x19, 0, 5, 0, 1647, 0., 0., 0., 0., "{Refinement seed} MethodCommands.html#MethodNonDLocalRel*"
```

GuiKeyWord kw_136[2] [static]

Initial value:

```cpp
= {
    "complementary", 8, 0, 1, 1, 1823,
    "cumulative", 8, 0, 1, 1, 1821
}
```

GuiKeyWord kw_137[1] [static]

Initial value:

```cpp
= {
    "num_gen.reliability_levels", 13, 0, 1, 0, 1831
}
```
GuiKeyWord kw_138[1]  [static]
Initial value:

```java
= {
    "num_probability_levels",13,0,1,0,1827
}
```

GuiKeyWord kw_139[3]  [static]
Initial value:

```java
= {
    "eval_id",8,0,2,0,1787,
    "header",8,0,1,0,1785,
    "interface_id",8,0,3,0,1789
}
```

GuiKeyWord kw_140[3]  [static]
Initial value:

```java
= {
    "annotated",8,0,1,0,1781,
    "custom_annotated",8,3,1,0,1783,kw_139,
    "freeform",8,0,1,0,1791
}
```

GuiKeyWord kw_141[3]  [static]
Initial value:

```java
= {
    "eval_id",8,0,2,0,1771,
    "header",8,0,1,0,1769,
    "interface_id",8,0,3,0,1773
}
```

GuiKeyWord kw_142[4]  [static]
Initial value:

```java
= {
    "active_only",8,0,2,0,1777,
    "annotated",8,0,1,0,1765,
    "custom_annotated",8,3,1,0,1767,kw_141,
    "freeform",8,0,1,0,1775
}
```

GuiKeyWord kw_143[2]  [static]
Initial value:

```java
= {
    "parallel",8,0,1,1,1817,
    "series",8,0,1,1,1815
}
```
GuiKeyWord kw_144[3] [static]
Initial value:
= {
    {"gen_reliabilities",8,0,1,1,1811},
    {"probabilities",8,0,1,1,1809},
    {"system",8,2,2,0,1813,kw_143}
}

GuiKeyWord kw_145[2] [static]
Initial value:
= {
    {"compute",8,3,2,0,1807,kw_144},
    {"num_response_levels",13,0,1,0,1805}
}

GuiKeyWord kw_146[2] [static]
Initial value:
= {
    {"mt19937",8,0,1,1,1799},
    {"rnum2",8,0,1,1,1801}
}

GuiKeyWord kw_147[16] [static]
Initial value:
= {
    {"dakota",8,0,2,0,1761},
    {"distribution",8,2,10,0,1819,kw_136},
    {"export_points_file",11,3,4,0,1779,kw_140.,0.,0.,0.,"[File name for exporting approximation-based samples from evaluating the GP] MethodCommands.html#MethodNonDGlobalRel"},
    {"gen_reliability_levels",14,1,12,0,1829,kw_137},
    {"import_points_file",11,4,3,0,1763,kw_142.,0.,0.,0.,"[File name for points to be imported as the basis for the initial GP] MethodCommands.html#MethodNonDGlobalRel"},
    {"model_pointer",11,0,0,0,1905},
    {"probability_levels",14,11,1,0,1825,kw_138},
    {"response_levels",14,2,8,0,1803,kw_145},
    {"rng",8,2,7,0,1797,kw_146},
    {"seed",8x19,0,6,0,1795,0,0.,0.,0.,0.,"[Random seed for initial GP construction] MethodCommands.html#MethodNonDGlobalRel"},
    {"surfpack",8,0,2,0,1759},
    {"u_gaussian_process",8,0,1,1,1757},
    {"u_kriging",0,0,1,1,1756},
    {"use_derivatives",8,0,5,0,1793,0,0.,0.,0.,0.,"[Derivative usage] MethodCommands.html#MethodNonDGlobalRel"},
    {"x_gaussian_process",8,0,1,1,1755},
    {"x_kriging",0,0,1,1,1754}
}

GuiKeyWord kw_148[2] [static]
Initial value:
= {
    {"master",8,0,1,1,178},
    {"peer",8,0,1,1,181}
}
GuiKeyWord kw_149[1]  [static]
Initial value:

```cpp
= {
    {"model_pointer_list", 11, 0, 1, 0, 143, 0, 0, 0, 0, 0, 0,"{List of model pointers} MethodCommands.html#MethodMetaHybrid"}
}
```

GuiKeyWord kw_150[2]  [static]
Initial value:

```cpp
= {
    {"method_name_list", 15, 1, 1, 141, kw_149, 0, 0, 0, 0,"{List of method names} MethodCommands.html#MethodMetaHybrid"},
    {"method_pointer_list", 15, 0, 1, 145, 0, 0, 0, 0,"{List of method pointers} MethodCommands.html#MethodMetaHybrid"}
}
```

GuiKeyWord kw_151[1]  [static]
Initial value:

```cpp
= {
    {"global_model_pointer", 11, 0, 1, 0, 127, 0, 0, 0, 0,"{Pointer to the global model specification} MethodCommands.html#MethodMetaHybrid"}
}
```

GuiKeyWord kw_152[1]  [static]
Initial value:

```cpp
= {
    {"local_model_pointer", 11, 0, 1, 0, 133, 0, 0, 0, 0,"{Pointer to the local model specification} MethodCommands.html#MethodMetaHybrid"}
}
```

GuiKeyWord kw_153[5]  [static]
Initial value:

```cpp
= {
    {"global_method_name", 11, 1, 1, 1, 125, kw_151, 0, 0, 0, 0,"{Name of the global method} MethodCommands.html#MethodMetaHybrid"},
    {"global_method_pointer", 11, 0, 1, 1, 129, 0, 0, 0, 0,"{Pointer to the global method specification} MethodCommands.html#MethodMetaHybrid"},
    {"local_method_name", 11, 1, 2, 131, kw_152, 0, 0, 0, 0,"{Name of the local method} MethodCommands.html#MethodMetaHybrid"},
    {"local_method_pointer", 11, 0, 2, 135, 0, 0, 0, 0,"{Pointer to the local method specification} MethodCommands.html#MethodMetaHybrid"},
    {"local_search_probability", 10, 0, 3, 0, 137, 0, 0, 0, 0,"{Probability of executing local searches} MethodCommands.html#MethodMetaHybrid"}
}
```
CHAPTER 12. NAMESPACE DOCUMENTATION

GuiKeyWord kw_154[1] [static]
Initial value:

= {
    "model_pointer_list",11,0,1,0,119,0,0,0,0,"[List of model pointers]
    MethodCommands.html#MethodMetaHybrid"
}

GuiKeyWord kw_155[2] [static]
Initial value:

= {
    "method_name_list",15,1,1,1,117,kw_154,0,0,0,0,"[List of method names]
    MethodCommands.html#MethodMetaHybrid",
    "method_pointer_list",15,0,1,121,0,0,0,0,0,"[List of method pointers]
    MethodCommands.html#MethodMetaHybrid"
}

GuiKeyWord kw_156[8] [static]
Initial value:

= {
    "collaborative",8,2,1,1,139,kw_155,0,0,0,0,"[Collaborative hybrid]
    MethodCommands.html#MethodMetaHybrid",
    "coupled",8,5,1,1,122,kw_153,
    "embedded",8,5,1,1,123,kw_153,0,0,0,0,"[Embedded hybrid]
    MethodCommands.html#MethodMetaHybrid",
    "iterator_scheduling",8,2,0,1,177,kw_148,0,0,0,0,"[Message passing configuration for
    scheduling of iterator jobs] MethodCommands.html#MethodMeta",
    "iterator_servers",0x19,0,2,0,175,0,0,0,0,0,"[Number of iterator servers]
    MethodCommands.html#MethodMeta",
    "processors_per_iterator",0x19,0,4,0,183,0,0,0,0,0,"[Number of processors per iterator
    server] MethodCommands.html#MethodMeta",
    "sequential",8,2,1,1,115,kw_155,0,0,0,0,"[Sequential hybrid]
    MethodCommands.html#MethodMetaHybrid",
    "uncoupled",8,2,1,1,114,kw_155}

GuiKeyWord kw_157[2] [static]
Initial value:

= {
    "parallel",8,0,1,1,1087,
    "series",8,0,1,1,1085
}

GuiKeyWord kw_158[3] [static]
Initial value:

= {
    "gen_reliabilities",8,0,1,1,1081,
    "probabilities",8,0,1,1,1079,
    "system",8,2,2,0,1083,kw_157}
GuiKeyWord kw_159[2]  [static]  
Initial value:  
= {  
    {"compute",8,3,2,0,1077,kw_158},  
    {"num_response_levels",13,0,1,0,1075}  
}  

GuiKeyWord kw_160[12]  [static]  
Initial value:  
= {  
    {"adapt_import",8,0,1,1,1067},  
    {"distribution",8,2,7,0,1307,kw_27,0,0,0,0,0,"{Distribution type} MethodCommands.html#MethodNonD"},  
    {"gen_reliability_levels",14,1,9,0,1317,kw_28,0,0,0,0,0,"{Generalized reliability levels} MethodCommands.html#MethodNonD"},  
    {"import",8,0,1,1,1065},  
    {"mm_adapt_import",8,0,1,1,1069},  
    {"model_pointer",11,0,4,0,1905},  
    {"probability_levels",14,1,8,0,1313,kw_29,0,0,0,0,0,"{Probability levels} MethodCommands.html#MethodNonD"},  
    {"refinement_samples",9,0,2,0,1071},  
    {"response_levels",14,2,3,0,1073,kw_158},  
    {"rng",8,2,10,0,1321,kw_36,0,0,0,0,0,"{Random number generator} MethodCommands.html#MethodNonDNC"},  
    {"samples",9,0,5,0,1645,0,0,0,0,0,0,"{Number of samples} MethodCommands.html#MethodNonDNC"},  
    {"seed",0x19,0,6,0,1647,0,0,0,0,0,"{Refinement seed} MethodCommands.html#MethodNonDLocalRel"}  
}  

GuiKeyWord kw_161[3]  [static]  
Initial value:  
= {  
    {"eval_id",8,0,2,0,1877},  
    {"header",8,0,1,0,1875},  
    {"interface_id",8,0,3,0,1879}  
}  

GuiKeyWord kw_162[4]  [static]  
Initial value:  
= {  
    {"active_only",8,0,2,0,1883},  
    {"annotated",8,0,1,0,1871},  
    {"custom_annotated",8,3,1,0,1873,kw_161},  
    {"freeform",8,0,1,0,1881}  
}  

GuiKeyWord kw_163[3]  [static]  
Initial value:  
= {  
    {"import_points_file",11,4,1,1,1869,kw_162},  
    {"list_of_points",14,0,1,1,1867,0,0,0,0,0,0,"{List of points to evaluate} MethodCommands.html#MethodPSLFS"},  
    {"model_pointer",11,0,2,0,1905}  
}
GuiKeyWord kw_164[2] [static]
Initial value:

= {
    {"complementary", 8, 0, 1, 1, 1683},
    {"cumulative", 8, 0, 1, 1, 1681}
}

GuiKeyWord kw_165[1] [static]
Initial value:

= {
    {"num_gen_reliability_levels", 13, 0, 1, 0, 1677}
}

GuiKeyWord kw_166[1] [static]
Initial value:

= {
    {"num_probability_levels", 13, 0, 1, 0, 1673}
}

GuiKeyWord kw_167[2] [static]
Initial value:

= {
    {"parallel", 8, 0, 1, 1, 1669},
    {"series", 8, 0, 1, 1, 1667}
}

GuiKeyWord kw_168[3] [static]
Initial value:

= {
    {"gen_reliabilities", 8, 0, 1, 1, 1663},
    {"probabilities", 8, 0, 1, 1, 1661},
    {"system", 8, 2, 2, 0, 1665, kw_167}
}

GuiKeyWord kw_169[2] [static]
Initial value:

= {
    {"compute", 8, 3, 2, 0, 1659, kw_168},
    {"num_response_levels", 13, 0, 1, 0, 1657}
}
12.1. DAKOTA NAMESPACE REFERENCE

GuiKeyWord kw_170[7]  [static]
Initial value:
= {
    {"distribution",8,2,5,0,1679,kw_166},
    {"gen_reliability_levels",14,1,4,0,1675,kw_165},
    {"model_pointer",11,0,6,0,1905},
    {"nip",8,0,1,0,1653},
    {"probability_levels",14,1,3,0,1671,kw_166},
    {"response_levels",14,2,2,0,1655,kw_169},
    {"sqp",8,0,1,0,1651}
}

GuiKeyWord kw_171[3]  [static]
Initial value:
= {
    {"model_pointer",11,0,2,0,1905},
    {"nip",8,0,1,0,1689},
    {"sqp",8,0,1,0,1687}
}

GuiKeyWord kw_172[5]  [static]
Initial value:
= {
    {"adapt_import",8,0,1,1,1723},
    {"import",8,0,1,1,1721},
    {"mm_adapt_import",8,0,1,1,1725},
    {"refinement_samples",9,0,2,0,1727},
    {"seed",0x19,0,3,0,1729,0,0,0,0,0,"{Random seed}
MethodCommands.html#MethodNonDBayesCalib"}
}

GuiKeyWord kw_173[4]  [static]
Initial value:
= {
    {"first_order",8,0,1,1,1715},
    {"probability_refinement",8,5,2,0,1719,kw_172},
    {"sample_refinement",0,5,2,0,1718,kw_172},
    {"second_order",8,0,1,1,1717}
}

GuiKeyWord kw_174[10]  [static]
Initial value:
= {
    {"integration",8,4,3,0,1713,kw_173,0,0,0,0,"{Integration method}
MethodCommands.html#MethodNonDLocalRel"},
    {"nip",8,0,2,0,1711},
    {"no_approx",8,0,1,1,1707},
    {"sqp",8,0,2,0,1709},
    {"u_taylor_mean",8,0,1,1,1697},
    {"u_taylor_mpp",8,0,1,1,1701},
    {"x_two_point",8,0,1,1,1705},
    {"x_taylor_mean",8,0,1,1,1699},
    {"x_taylor_mpp",8,0,1,1,1699},
    {"x_two_point",8,0,1,1,1703}
}
GuiKeyWord kw_175[1] [static]
Initial value:

```javascript
= {
    "num_reliability_levels", 13, 0, 1, 0, 1751
}
```

GuiKeyWord kw_176[2] [static]
Initial value:

```javascript
= {
    "parallel", 8, 0, 1, 1, 1747,
    "series", 8, 0, 1, 1, 1745
}
```

GuiKeyWord kw_177[4] [static]
Initial value:

```javascript
= {
    "gen_reliabilities", 8, 0, 1, 1, 1741,
    "probabilities", 8, 0, 1, 1, 1737,
    "reliabilities", 8, 0, 1, 1, 1739,
    "system", 8, 2, 2, 0, 1743, kw_176
}
```

GuiKeyWord kw_178[2] [static]
Initial value:

```javascript
= {
    "compute", 8, 2, 0, 1735, kw_177,
    "num_response_levels", 13, 0, 1, 0, 1733
}
```

GuiKeyWord kw_179[7] [static]
Initial value:

```javascript
= {
    "distribution", 8, 2, 5, 0, 1819, kw_136,
    "gen_reliability_levels", 14, 1, 7, 0, 1829, kw_137,
    "model_pointer", 11, 0, 4, 0, 1905,
    "mpp_search", 8, 10, 0, 0, 1693, kw_174, 0, 0, 0, 0, "MPP search type MethodCommands.html#MethodNonDLocalRel",
    "probability_levels", 14, 1, 6, 0, 1825, kw_138,
    "reliability_levels", 14, 1, 5, 0, 1749, kw_175,
    "response_levels", 14, 2, 2, 0, 1731, kw_176
}
```
GuiKeyWord kw_180[17]  [static]
Initial value:

```cpp
= {
    {"display_all_evaluations",8,0,7,0,399,0,0,0,0,"[Display NOMAD evaluations] MethodCommands.html#MethodNOMADDC"},
    {"display_format",11,0,4,0,393},
    {"function_precision",10,0,1,0,387,0,0,0,0,"[Function Evaluation Precision] MethodCommands.html#MethodNOMADDC"},
    {"history_file",11,0,3,0,391,0,0,0,0,"[NOMAD History File] MethodCommands.html#MethodNOMADDC"},
    {"linear_equality_constraint_matrix",14,0,14,0,533,0,0,0,0,"[Linear equality coefficient matrix] MethodCommands.html#MethodMin"},
    {"linear_equality_scale_types",15,0,16,0,537,0,0,0,0,"[Linear equality scaling types] MethodCommands.html#MethodMin"},
    {"linear_equality_scales",14,0,17,0,539,0,0,0,0,"[Linear equality scales] MethodCommands.html#MethodMin"},
    {"linear_equality_targets",14,0,15,0,535,0,0,0,0,"[Linear equality targets] MethodCommands.html#MethodMin"},
    {"linear_inequality_constraint_matrix",14,0,9,0,523,0,0,0,0,"[Linear inequality coefficient matrix] MethodCommands.html#MethodMin"},
    {"linear_inequality_lower_bounds",14,0,10,0,525,0,0,0,0,"[Linear inequality lower bounds] MethodCommands.html#MethodMin"},
    {"linear_inequality_scale_types",15,0,12,0,529,0,0,0,0,"[Linear inequality scaling types] MethodCommands.html#MethodMin"},
    {"linear_inequality_scale_types",15,0,13,0,531,0,0,0,0,"[Linear inequality scaling types] MethodCommands.html#MethodMin"},
    {"linear_inequality_upper_bounds",14,0,11,0,527,0,0,0,0,"[Linear inequality upper bounds] MethodCommands.html#MethodMin"},
    {"model_pointer",11,0,8,0,1905},
    {"neighbor_order",0x19,0,6,0,397},
    {"seed",0x19,0,2,0,389,0,0,0,0,"[Random Seed] MethodCommands.html#MethodNOMADDC"},
    {"variable_neighborhood_search",10,0,5,0,395}
}
```

GuiKeyWord kw_181[2]  [static]
Initial value:

```cpp
= {
    {"num_offspring",0x19,0,2,0,501,0,0,0,0,"[Number of offspring in random shuffle crossover] MethodCommands.html#MethodJEGADC"},
    {"num_parents",0x19,0,1,0,499,0,0,0,0,"[Number of parents in random shuffle crossover] MethodCommands.html#MethodJEGADC"}
}
```

GuiKeyWord kw_182[5]  [static]
Initial value:

```cpp
= {
    {"crossover_rate",10,0,2,0,503,0,0,0,0,"[Crossover rate] MethodCommands.html#MethodJEGADC"},
    {"multi_point_binary",9,0,1,1,491,0,0,0,0,"[Multi point binary crossover] MethodCommands.html#MethodJEGADC"},
    {"multi_point_parameterized_binary",9,0,1,1,493,0,0,0,0,"[Multi point parameterized binary crossover] MethodCommands.html#MethodJEGADC"},
    {"multi_point_real",9,0,1,1,495,0,0,0,0,"[Multi point real crossover] MethodCommands.html#MethodJEGADC"},
    {"shuffle_random",6,2,1,1,497,kw_181,0,0,0,0,"[Random shuffle crossover] MethodCommands.html#MethodJEGADC"}
}
```
GuiKeyWord kw_183[3] [static]
Initial value:

```
= {
    "flat_file",1,0,1,1,487,
    "simple_random",8,0,1,1,483,
    "unique_random",8,0,1,1,485
}
```

GuiKeyWord kw_184[1] [static]
Initial value:

```
= {
    "mutation_scale",10,0,1,0,517,0,0,,0,,0,,0,"{Mutation scale}
    MethodCommands.html#MethodJEGADC"
}
```

GuiKeyWord kw_185[6] [static]
Initial value:

```
= {
    "bit_random",8,0,1,1,507,
    "mutation_rate",10,0,2,0,519,0,0,,0,,0,,0,"{Mutation rate}
    MethodCommands.html#MethodJEGADCM",
    "offset_cauchy",8,1,1,1,513,kw_184,
    "offset_normal",8,1,1,1,511,kw_184,
    "offset_uniform",8,1,1,1,515,kw_184,
    "replace_uniform",8,0,1,1,509
}
```

GuiKeyWord kw_186[3] [static]
Initial value:

```
= {
    "metric_tracker",8,0,1,1,433,0,0,,0,,0,,0,"{Convergence type}
    MethodCommands.html#MethodJEGAMOGA",
    "num_generations",0x29,0,3,0,437,0,0,,0,,0,,0,*{Number generations for metric_tracker
    converger} MethodCommands.html#MethodJEGAMOGA",
    "percent_change",10,0,2,0,435,0,0,,0,,0,,0,*{Percent change limit for metric_tracker
    converger} MethodCommands.html#MethodJEGAMOGA"
}
```

GuiKeyWord kw_187[2] [static]
Initial value:

```
= {
    "domination_count",8,0,1,1,407,
    "layer_rank",8,0,1,1,495
}
```
GuiKeyWord kw_188[1] [static]
Initial value:

= {
    {"num_designs",0x29,0,1,0,429,0,2.,0.,0.,0,*{Number designs to keep for max_designs nicher
MethodCommands.html#MethodJEGAMOGA"}
}

GuiKeyWord kw_189[3] [static]
Initial value:

= {
    {"distance",14,0,1,1,425},
    {"max_designs",14,1,1,1,427,kw_188},
    {"radial",14,0,1,1,423}  
}

GuiKeyWord kw_190[1] [static]
Initial value:

= {
    {"orthogonal_distance",14,0,1,1,441,0,0.,0.,0.,0,*[Post_processor distance
MethodCommands.html#MethodJEGAMOGA"}
}

GuiKeyWord kw_191[2] [static]
Initial value:

= {
    {"shrinkage_fraction",10,0,1,0,419},
    {"shrinkage_percentage",2,0,1,0,418}  
}

GuiKeyWord kw_192[4] [static]
Initial value:

= {
    {"below_limit",10,2,1,1,417,kw_191,0,,0,,0,*{Below limit selection
MethodCommands.html#MethodJEGADIC"},
    {"elitist",8,0,1,1,411},
    {"roulette_wheel",8,0,1,1,413},
    {"unique_roulette_wheel",8,0,1,1,415}  
}
GuiKeyWord kw_193[22] [static]
Initial value:

```c
= {
   {"convergence_type",8,3,4,0,431,kw_186},
   {"crossover_type",8,5,20,0,489,kw_182,0,0,0,0,0,"{Crossover type
MethodCommands.html#MethodJEGADC"},
   {"fitness_type",8,2,1,0,403,kw_187,0,0,0,0,0,"{Fitness type
MethodCommands.html#MethodJEGAMOGA"},
   {"initialization_type",8,3,19,0,481,kw_183,0,0,0,0,0,"{Initialization type
MethodCommands.html#MethodJEGADC"},
   {"linear_equality_constraint_matrix",14,0,12,0,533,0,0,0,0,0,"{Linear equality
coefficient matrix} MethodCommands.html#MethodMin"},
   {"linear_equality_scale_types",15,0,14,0,537,0,0,0,0,0,"{Linear equality scaling types
MethodCommands.html#MethodJEGADC"},
   {"linear_equality_scales",14,0,15,0,539,0,0,0,0,0,"{Linear equality scales
MethodCommands.html#MethodMin"},
   {"strategies",14,0,13,0,535,0,0,0,0,0,"{Linear equality targets
MethodCommands.html#MethodMin"},
   {"linear_inequality_constraint_matrix",14,0,7,0,523,0,0,0,0,0,"{Linear inequality
coefficient matrix} MethodCommands.html#MethodMin"},
   {"linear_inequality_lower_bounds",14,0,8,0,525,0,0,0,0,0,"{Linear inequality lower
bounds} MethodCommands.html#MethodMin"},
   {"linear_inequality_scaling_types",15,0,10,0,529,0,0,0,0,0,"{Linear inequality scaling
types} MethodCommands.html#MethodMin"},
   {"linear_inequality_scales",14,0,11,0,531,0,0,0,0,0,"{Linear inequality scales
MethodCommands.html#MethodMin"},
   {"linear_inequality_upper_bounds",14,0,9,0,527,0,0,0,0,0,"{Linear inequality upper
bounds} MethodCommands.html#MethodMin"},
   {"log_file",11,0,17,0,477,0,0,0,0,0,"{Log file} MethodCommands.html#MethodJEGADC"},
   {"model_pointer",11,0,6,0,1905},
   {"mutation_type",8,6,31,0,505,kw_185,0,0,0,0,0,"{Mutation type
MethodCommands.html#MethodJEGADC"},
   {"niching_type",8,3,3,0,421,kw_189,0,0,0,0,0,"{Niche pressure type
MethodCommands.html#MethodJEGAMOGA"},
   {"population_size",6x29,0,16,0,475,0,0,0,0,0,"{Number of population members
MethodCommands.html#MethodJEGADC"},
   {"postprocessor_type",8,1,5,0,439,kw_190,0,0,0,0,0,"{Postprocessor type
MethodCommands.html#MethodJEGAMOGA"},
   {"print_each_pop",8,0,18,0,479,0,0,0,0,0,"{Population output
MethodCommands.html#MethodJEGADC"},
   {"replacement_type",8,4,2,0,403,kw_192,0,0,0,0,0,"{Replacement type
MethodCommands.html#MethodJEGAMOGA"},
   {"seed",0x19,0,22,0,521,0,0,0,0,0,"{Random seed} MethodCommands.html#MethodJEGADC"}
}
```

GuiKeyWord kw_194[1] [static]
Initial value:

```c
= {
   {"model_pointer",11,0,1,0,151,0,0,0,0,0,0,"{Model pointer} MethodCommands.html#MethodMeta"}
}
```

GuiKeyWord kw_195[1] [static]
Initial value:

```c
= {
   {"seed",9,0,1,0,157,0,0,0,0,0,0,"{Seed for random starting points
MethodCommands.html#MethodMetaMultiStart"}
}
```
12.1. DAKOTA NAMESPACE REFERENCE

GuiKeyWord kw_196[7]  [static]

Initial value:

= {
     "iterator_scheduling",8,2,5,0,177,kw_148,0.,0.,0.,0.,"{Message passing configuration for scheduling of iterator jobs} MethodCommands.html#MethodMeta",
     "iterator_servers",0x19,0,4,0,175,0,0.,0.,0.,"{Number of iterator servers} MethodCommands.html#MethodMeta",
     "method_name",11,1,1,1,149,kw_194,0.,0.,0.,0.,"{Identification of a sub-method by name (no separate specification block)} MethodCommands.html#MethodMeta",
     "method_pointer",11,0,1,1,153,0,0.,0.,0.,"{Identification of a sub-method by pointer to a separate specification block} MethodCommands.html#MethodMeta",
     "processors_per_iterator",0x19,0,6,0,183,0,0.,0.,0.,"{Number of processors per iterator server} MethodCommands.html#MethodMeta",
     "random_starts",8,1,2,0,155,kw_195,0.,0.,0.,0.,"{Number of random starting points} MethodCommands.html#MethodMetaMultiStart",
     "starting_points",14,0,3,0,159,0,0.,0.,0.,0.,"{List of user-specified starting points} MethodCommands.html#MethodMetaMultiStart",
}

GuiKeyWord kw_197[2]  [static]

Initial value:

= {
     "model_pointer",11,0,2,0,1905,
     "partitions",13,0,11,1,1893,0,0.,0.,0.,"{Partitions per variable} MethodCommands.html#MethodPSMPS",
}

GuiKeyWord kw_198[5]  [static]

Initial value:

= {
     "min_boxsize_limit",10,0,2,0,709,0,0.,0.,0.,0.,"{Min boxsize limit} MethodCommands.html#MethodNCSUDC",
     "model_pointer",11,0,4,0,1905,
     "solution_accuracy",2,0,1,1,706,
     "solution_target",10,0,1,0,707,0,0.,0.,0.,0.,"{Solution Target} MethodCommands.html#MethodNCSUDC",
     "volume_boxsize_limit",10,0,3,0,711,0,0.,0.,0.,0.,"{Volume boxsize limit} MethodCommands.html#MethodNCSUDC",
}

GuiKeyWord kw_199[10]  [static]

Initial value:

= {
     "absolute_conv_tol",10,0,2,0,685,0,0.,0.,0.,0.,"{Absolute function convergence tolerance} MethodCommands.html#MethodLSNL2SOL",
     "covariance",9,0,8,0,697,0,0.,0.,0.,0.,"{Covariance post-processing} MethodCommands.html#MethodLSNL2SOL",
     "false_conv_tol",10,0,6,0,693,0,0.,0.,0.,0.,"{False convergence tolerance} MethodCommands.html#MethodLSNL2SOL",
     "function_precision",10,0,1,0,683,0,0.,0.,0.,0.,"{Relative precision in least squares terms} MethodCommands.html#MethodLSNL2SOL",
     "initial_trust_radius",10,0,7,0,695,0,0.,0.,0.,0.,"{Initial trust region radius} MethodCommands.html#MethodLSNL2SOL",
     "model_pointer",11,0,10,0,1905,
     "regression_diagnostics",8,0,9,0,699,0,0.,0.,0.,0.,"{Regression diagnostics post-processing} MethodCommands.html#MethodLSNL2SOL",}
GUIKeyword kw_200[2] [static]
Initial value:
= {
    "global", 8, 0, 1, 1, 1217,
    "local", 8, 0, 1, 1, 1215
}

GUIKeyword kw_201[2] [static]
Initial value:
= {
    "parallel", 8, 0, 1, 1, 1235,
    "series", 8, 0, 1, 1, 1233
}

GUIKeyword kw_202[3] [static]
Initial value:
= {
    "gen_reliabilities", 8, 0, 1, 1, 1229,
    "probabilities", 8, 0, 1, 1, 1227,
    "system", 8, 2, 2, 0, 1231, kw_201
}

GUIKeyword kw_203[2] [static]
Initial value:
= {
    "compute", 8, 3, 2, 0, 1225, kw_202,
    "num_response_levels", 13, 0, 1, 0, 1223
}

GUIKeyword kw_204[10] [static]
Initial value:
= {
    "distribution", 8, 2, 7, 0, 1307, kw_20, 0, 0, 0, 0,"{Distribution type}
    MethodCommands.html#MethodNonD",
    "emulator_samples", 9, 0, 2, 0, 1219,
    "gen_reliability_levels", 14, 1, 9, 0, 1317, kw_28, 0, 0, 0, 0,"{Generalized reliability levels}
    MethodCommands.html#MethodNonD",
    "lipschitz", 8, 2, 1, 0, 1213, kw_200,
    "model_pointer", 11, 0, 4, 0, 1905,
    "probability_levels", 14, 1, 8, 0, 1313, kw_29, 0, 0, 0, 0,"{Probability levels}
    MethodCommands.html#MethodNonD",
    "singular_conv_tol", 10, 0, 3, 0, 0, 0, 0, 0, 0,"{Singular convergence tolerance}
    MethodCommands.html#MethodLSNL2SOL",
    "singular_radius", 10, 0, 5, 0, 0, 0, 0, 0, 0,"{Step limit for sctol}
    MethodCommands.html#MethodLSNL2SOL",
    "x_conv_tol", 10, 0, 1, 0, 0, 0, 0, 0, 0,"{Convergence tolerance for change in parameter vector}
    MethodCommands.html#MethodLSNL2SOL"}
GuiKeyword kw_205[1]  [static]
Initial value:
   = {
      {"num_reliability_levels",13,0,1,0,1043,0,0.,0.,0.,0.,0.,0,"Number of reliability levels"
          MethodCommands.html#MethodNonD"}
      {"seed",0x19,0,6,0,1647,0,0.,0.,0.,0.,0.,0,"Refinement seed"
          MethodCommands.html#MethodNonDLocalReli"}
   }

GuiKeyword kw_206[2]  [static]
Initial value:
   = {
      {"parallel",8,0,1,1,1061},
      {"series",8,0,1,1,1059}
   }

GuiKeyword kw_207[4]  [static]
Initial value:
   = {
      {"gen_reliabilities",8,0,1,1,1055},
      {"probabilities",8,0,1,1,1051},
      {"reliabilities",8,0,1,1,1053},
      {"system",8,2,2,0,1057,kw_206}\}

GuiKeyword kw_208[2]  [static]
Initial value:
   = {
      {"compute",8,4,2,0,1049,kw_207,0.,0.,0.,0.,0.,0."Target statistics for response levels"
          MethodCommands.html#MethodNonD"},
      {"num_response_levels",13,0,1,0,1047,0,0.,0.,0.,0.,0.,0."Number of response levels"
          MethodCommands.html#MethodNonD"}
   }

GuiKeyword kw_209[3]  [static]
Initial value:
   = {
      {"eval_id",8,0,2,0,875},
      {"header",8,0,1,0,873},
      {"interface_id",8,0,3,0,877}
   }
GuiKeyWord kw_210[4] [static]
Initial value:

```java
= {
    "active_only", 8, 0, 2, 0, 881,
    "annotated", 8, 0, 1, 0, 869,
    "custom.annotated", 8, 3, 1, 0, 871, kw_209,
    "freeform", 8, 0, 1, 0, 879
}
```

GuiKeyWord kw_211[2] [static]
Initial value:

```java
= {
    "advancements", 9, 0, 1, 0, 807,
    "soft.convergence.limit", 9, 0, 2, 0, 809
}
```

GuiKeyWord kw_212[3] [static]
Initial value:

```java
= {
    "adapted", 8, 2, 1, 1, 805, kw_211,
    "tensor.product", 8, 0, 1, 1, 801,
    "total.order", 8, 0, 1, 1, 803
}
```

GuiKeyWord kw_213[1] [static]
Initial value:

```java
= {
    "noise.tolerance", 14, 0, 1, 0, 831
}
```

GuiKeyWord kw_214[1] [static]
Initial value:

```java
= {
    "noise.tolerance", 14, 0, 1, 0, 835
}
```

GuiKeyWord kw_215[2] [static]
Initial value:

```java
= {
    "l2.penalty", 10, 0, 2, 0, 841, 0, 0, 0, 0, 0,
    "noise.tolerance", 14, 0, 1, 0, 839
}
```
12.1. DAKOTA NAMESPACE REFERENCE

GuiKeyWord kw_216[2]  [static]
Initial value:

```cpp
= {
    {
        "equality constrained",8,0,1,0,821,
        "svd",8,0,1,0,819
    }
}
```

GuiKeyWord kw_217[1]  [static]
Initial value:

```cpp
= {
    {
        "noise_tolerance",14,0,1,0,825
    }
}
```

GuiKeyWord kw_218[17]  [static]
Initial value:

```cpp
= {
    {
        "basis_pursuit",8,0,2,0,827,0,0,0,0,0,"(L1 minimization via Basis Pursuit (BP)) MethodCommands.html#MethodNonDPCE",
        "basis_pursuit_de-noising",8,1,2,0,829,kw_213,0,0,0,0,"(L1 minimization via Basis Pursuit De-noising (BPDO)) MethodCommands.html#MethodNonDPCE",
        "bp",0,0,2,0,826,
        "bpdn",0,1,2,0,828,kw_213,
        "cross_validation",8,0,3,0,843,0,0,0,0,0,"(Specify whether to use cross validation) MethodCommands.html#MethodNonDPCE",
        "lars",0,1,2,0,832,kw_214,
        "lasso",0,2,2,0,836,kw_215,
        "least_absolute_shrinkage",8,2,2,0,837,kw_215,0,0,0,0,0,"(L1 minimization via Least Absolute Shrinkage Operator (LASSO)) MethodCommands.html#MethodNonDPCE",
        "least_angle_regression",8,1,2,0,833,kw_214,0,0,0,0,0,"(L1 minimization via Least Angle Regression (LARS)) MethodCommands.html#MethodNonDPCE",
        "least_squares",8,2,2,0,817,kw_216,0,0,0,0,0,"(Least squares regression) MethodCommands.html#MethodNonDPCE",
        "omp",0,1,2,0,822,kw_217,
        "orthogonal_matching_pursuit",8,1,2,0,823,kw_217,0,0,0,0,0,"(L1 minimization via Orthogonal Matching Pursuit (OMP)) MethodCommands.html#MethodNonDPCE",
        "ratio_order",10,0,1,0,815,0,0,0,0,0,"(Order of collocation oversampling relationship) MethodCommands.html#MethodNonDPCE",
        "use_derivatives",8,0,4,0,845
    }
}
```

GuiKeyWord kw_219[3]  [static]
Initial value:

```cpp
= {
    {
        "incremental_lhs",8,0,2,0,855,0,0,0,0,0,"(Use incremental LHS for expansion_samples) MethodCommands.html#MethodNonDPCE",
        "reuse_points",8,0,1,0,853,
        "reuse_samples",0,0,1,0,852
    }
}
```
GuiKeyWord kw_220[6] [static]
Initial value:
= {
    "basis_type", 8, 3, 2, 0, 799, kw_212,
    "collocation_points", 13, 17, 3, 1, 811, kw_218, 0., 0., 0., 0., "Number collocation points to estimate coeffs MethodCommands.html#MethodNonDPCE",
    "collocation_ratio", 10, 17, 3, 1, 813, kw_218, 0., 0., 0., 0., "Collocation point oversampling ratio to estimate coeffs MethodCommands.html#MethodNonDPCE",
    "dimension_preference", 14, 0, 1, 0, 797,
    "expansion_samples", 13, 3, 1, 1, 851, kw_219, 0., 0., 0., 0., "Number simulation samples to estimate coeffs MethodCommands.html#MethodNonDPCE",
    "import_points_filename", 11, 4, 4, 0, 867, kw_210, 0., 0., 0., 0., "File name for points to be imported for forming a PCE (unstructured grid assumed) MethodCommands.html#MethodNonDPCE"
}

GuiKeyWord kw_221[3] [static]
Initial value:
= {
    "eval_id", 8, 0, 2, 0, 921,
    "header", 8, 0, 1, 0, 919,
    "interface_id", 8, 0, 3, 0, 923
}

GuiKeyWord kw_222[3] [static]
Initial value:
= {
    "annotated", 8, 0, 1, 0, 915,
    "custom_annotated", 8, 3, 1, 0, 917, kw_221,
    "freeform", 8, 0, 1, 0, 925
}

GuiKeyWord kw_223[6] [static]
Initial value:
= {
    "collocation_points", 13, 0, 1, 1, 859,
    "cross_validation", 8, 0, 2, 0, 861,
    "import_points_filename", 11, 4, 5, 0, 867, kw_210, 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, 865,
    "reuse_samples", 0, 0, 4, 0, 864,
    "tensor_grid", 13, 0, 3, 0, 863
}

GuiKeyWord kw_224[3] [static]
Initial value:
= {
    "decay", 8, 0, 1, 1, 771,
    "generalized", 8, 0, 1, 1, 773,
    "sobol", 8, 0, 1, 1, 769"}
GuiKeyWord kw_225[2] [static]
Initial value:

= 
  {
    "dimension_adaptive",8,3,1,1,767,kw_224
  ,
    "uniform",8,0,1,1,765
  }

GuiKeyWord kw_226[4] [static]
Initial value:

= 
  {
    "adapt_import",8,0,1,1,903
  ,
    "import",8,0,1,1,905
  ,
    "mm_adapt_import",8,0,1,1,909
  ,
    "refinement_samples",9,0,2,0,911,0,0,0,0,"{Refinement samples}
    MethodCommands.html#MethodNonDLocalRel"
  }

GuiKeyWord kw_227[3] [static]
Initial value:

= 
  {
    "dimension_preference",14,0,1,0,787,0,0,0,0,"{Dimension preference for anisotropic
tensor and sparse grids} MethodCommands.html#MethodNonDPCE"
  ,
    "nested",8,0,2,0,789
  ,
    "non_nested",8,0,2,0,791
  }

GuiKeyWord kw_228[2] [static]
Initial value:

= 
  {
    "lhs",8,0,1,1,899
  ,
    "random",8,0,1,1,901
  }

GuiKeyWord kw_229[5] [static]
Initial value:

= 
  {
    "dimension_preference",14,0,2,0,787,0,0,0,0,"{Dimension preference for anisotropic
tensor and sparse grids} MethodCommands.html#MethodNonDPCE"
  ,
    "nested",8,0,3,0,789
  ,
    "non_nested",8,0,3,0,791
  ,
    "restricted",8,0,1,0,783
  ,
    "unrestricted",8,0,1,0,785
  }
GuiKeyWord kw_230[2]  [static]
Initial value:
= {
    "drop_tolerance",10,0,2,0,889,0,0,0,0,0,0,"{VBD tolerance for omitting small indices} MethodCommands.html#MethodNonDMC",
    "interaction_order",8x15,0,1,0,887,0,0,0,0,0,0,"{Restriction of order of VBD iterations} MethodCommands.html#MethodNonDPCF"
}

GuiKeyWord kw_232[1]  [static]
Initial value:
= {
    "previous_samples",9,0,1,1,1031,0,0,0,0,0,0,"{Previous samples for incremental approaches} MethodCommands.html#MethodNonDMC"
}

GuiKeyWord kw_233[4]  [static]
Initial value:
= {
    "incremental_lhs",8,1,1,1027,kw_232",
    "incremental_random",8,1,1,1029,kw_232",
    "lhs",8,0,1,1,1025",
    "random",8,0,1,1,1023"
}

GuiKeyWord kw_234[1]  [static]
Initial value:
= {
    "drop_tolerance",10,0,1,0,1035
}

GuiKeyWord kw_235[13]  [static]
Initial value:
= {
    "backfill",8,0,3,0,1037",
    "distribution",8,2,7,0,1307,kw_235",
    "distribution",8,2,7,0,1307,kw_235",
    "fixed_seed",8,0,15,0,1039,0,0,0,0,0,0,"{Fixed seed flag} MethodCommands.html#MethodNonDMC",
    "gen_reliability_levels",14,1,9,0,1317,kw_235,1041,0,0,0,0,0,"{Generalized reliability levels} MethodCommands.html#MethodNonD",
    "gen_reliability_levels",14,1,11,0,1317,kw_235,1041,0,0,0,0,0,"{Generalized reliability levels} MethodCommands.html#MethodNonD",
    "model_pointer",11,0,4,0,1905",
    "probability_levels",14,1,8,0,1313,kw_235,0,0,0,0,0,"{Probability levels} MethodCommands.html#MethodNonD",
    "reliability_levels",14,1,11,0,1317,kw_235,1041,0,0,0,0,0,"{Reliability levels} MethodCommands.html#MethodNonD",
    "reliability_levels",14,1,11,0,1317,kw_235,1041,0,0,0,0,0,"{Reliability levels} MethodCommands.html#MethodNonD",
    "response_levels",14,2,12,0,1045,kw_235,0,0,0,0,0,"{Response levels} MethodCommands.html#MethodNonD",
    "rng",8,2,10,0,1321,kw_235,1041,0,0,0,0,0,"{Random number generator} MethodCommands.html#MethodNonDMC",
    "sample_type",8,4,1,3,1021,kw_235",
    "samples",9,5,5,0,1645,0,0,0,0,0,"{Number of samples} MethodCommands.html#MethodNonDMC"}
GuiKeyWord kw_236[3]  [static]
Initial value:

= {
   {"eval_id",8,0,2,0,1013},
   {"header",8,0,1,0,1011},
   {"interface_id",8,0,3,0,1015}
}

GuiKeyWord kw_237[3]  [static]
Initial value:

= {
   {"annotated",8,0,1,0,1007},
   {"custom_annotated",8,3,1,0,1009,kw_236},
   {"freeform",8,0,1,0,1017}
}

GuiKeyWord kw_238[2]  [static]
Initial value:

= {
   {"generalized",8,0,1,1,949},
   {"sobol",8,0,1,1,947}
}

GuiKeyWord kw_239[3]  [static]
Initial value:

= {
   {"dimension_adaptive",8,2,1,1,945,kw_238},
   {"local_adaptive",8,0,1,1,951},
   {"uniform",8,0,1,1,943}
}

GuiKeyWord kw_240[2]  [static]
Initial value:

= {
   {"generalized",8,0,1,1,939},
   {"sobol",8,0,1,1,937}
}
GuiKeyWord kw_241[2] [static]
Initial value:

= {
    {"dimension,adaptive",8,2,1,1,935,kw,240},
    {"uniform",8,0,1,1,933}
}

GuiKeyWord kw_242[4] [static]
Initial value:

= {
    {"adapt_import",8,0,1,1,999},
    {"import",8,0,1,1,997},
    {"mm adapt import",8,0,1,1,1001},
    {"refinement_samples",9,0,2,0,1003}
}

GuiKeyWord kw_243[2] [static]
Initial value:

= {
    {"lhs",8,0,1,1,991},
    {"random",8,0,1,1,993}
}

GuiKeyWord kw_244[4] [static]
Initial value:

= {
    {"hierarchical",8,0,2,0,969},
    {"nodal",8,0,2,0,967},
    {"restricted",8,0,1,0,963},
    {"unrestricted",8,0,1,0,965}
}

GuiKeyWord kw_245[2] [static]
Initial value:

= {
    {"drop.tolerance",10,0,2,0,983,0,0..0,0,0,0,"{VBD tolerance for omitting small indices}
MethodCommands.html#MethodNonDSC"},
    {"interaction.order",0x19,0,1,0,981,0,0..0,0,0,"{Restriction of order of VBD iterations}
MethodCommands.html#MethodNonDSC"}
}
GuiKeyWord kw_246[28] [static]

Initial value:

```
= {
  {*askey*, 8, 0, 2, 0, 955},
  {*diagonal_covariance*, 8, 0, 8, 0, 985},
  {*dimension_preference*, 14, 0, 4, 0, 971},
  {*distribution*, 8, 2, 15, 0, 1307},
  {*fixed_seed*, 8, 0, 21, 0, 1039},
  {*gen_reliability_levels*, 14, 1, 17, 0, 1317},
  {*h_refinement*, 8, 3, 1, 0, 941},
  {*nested*, 8, 0, 6, 0, 975},
  {*non_nested*, 8, 0, 6, 0, 977},
  {*p_refinement*, 8, 2, 1, 0, 931},
  {*quadrature_order*, 13, 0, 3, 1, 959},
  {*probability_levels*, 14, 1, 16, 0, 1313},
  {*seed*, 0x19, 0, 14, 0, 1647},
  {*sparse_grid_level*, 13, 4, 3, 1, 961},
  {*samples*, 9, 0, 13, 0, 1645},
  {*sample_type*, 8, 2, 9, 0, 989},
  {*use_derivatives*, 8, 0, 2, 0, 973},
  {*variance_based_decomp*, 8, 2, 7, 0, 979},
  {*wiener*, 8, 0, 2, 0, 957},
}
```

GuiKeyWord kw_247[2] [static]

Initial value:

```
= {
  {*function_precision*, 10, 0, 12, 0, 303},
  {*linear_equality_constraint_matrix*, 14, 0, 7, 0, 533},
}
```

GuiKeyWord kw_248[13] [static]

Initial value:

```
= {
  {*function_precision*, 10, 0, 12, 0, 303},
  {*linear_equality_constraint_matrix*, 14, 0, 7, 0, 533},
  }
```
GuiKeyWord kw_249[12] [static]

Initial value:

```c
= {
    "gradient_tolerance","10,0,12,0,343",
    "linear_equality_constraint_matrix","0,533,0,0,0,0,0,0,0",
    "linear_equality_scale_types","15,0,9,0,537,0,0,0,0,0,0",
    "linear_equality_scales","14,0,10,1,537,0,0,0,0,0,0",
    "linear_equality_targets","14,0,8,0,535,0,0,0,0,0,0",
    "inequality_lower_bounds","14,0,3,0,523,0,0,0,0,0,0",
    "inequality_upper_bounds","14,0,4,0,527,0,0,0,0,0,0",
    "model_pointer","11,0,1,0,1905",
    "verify_level","9,0,11,0,301,0,0,0,0,0,0",
    "Gradient verification level",
}
```

GuiKeyWord kw_250[11] [static]

Initial value:

```c
= {
    "linear_equality_constraint_matrix","0,533,0,0,0,0,0,0,0",
    "linear_equality_scale_types","15,0,10,1,537,0,0,0,0,0,0",
    "linear_equality_scales","14,0,11,0,539,0,0,0,0,0,0",
    "linear_equality_targets","14,0,9,0,535,0,0,0,0,0,0",
    "inequality_lower_bounds","14,0,4,0,523,0,0,0,0,0,0",
    "inequality_upper_bounds","14,0,5,0,529,0,0,0,0,0,0",
    "model_pointer","11,0,1,0,1905",
    "verify_level","9,0,11,0,301,0,0,0,0,0,0",
    "Gradient verification level",
}```
GuiKeyWord kw_251[3] [static]
Initial value:
= {
    {*args, *tapia*, 8, 0, 1, 1, 333},
    {*elbakry*, 8, 0, 1, 1, 333},
    {*van*shanno*, 8, 0, 1, 1, 335}
}

GuiKeyWord kw_252[4] [static]
Initial value:
= {
    {*gradient_based_line_search*, 8, 0, 1, 1, 323, 0, 0, 0, 0, 0, *CHOOSE line search type*},
    {*tr_gds*, 8, 0, 1, 1, 327},
    {*trust_region*, 8, 0, 1, 1, 325},
    {*value_based_line_search*, 8, 0, 1, 1, 321}
}

GuiKeyWord kw_253[16] [static]
Initial value:
= {
    {*centering_parameter*, 10, 0, 4, 0, 339},
    {*gradient_tolerance*, 10, 5, 16, 0, 343},
    {*linear_inequality_constraint_matrix*, 14, 0, 11, 0, 533, 0, 0, 0, 0, 0, 0, *CHOOSE
    linear equality coefficient matrix*},
    {*linear_inequality_scale_matrix*, 14, 0, 11, 0, 533, 0, 0, 0, 0, 0, 0, *CHOOSE
    linear equality scaling matrix*},
    {*linear_inequality_types*, 15, 0, 13, 0, 537, 0, 0, 0, 0, 0, 0, *CHOOSE
    linear inequality types*},
    {*linear_inequality_upper_bounds*, 14, 0, 10, 0, 531, 0, 0, 0, 0, 0, 0, *CHOOSE
    linear inequality upper bounds*},
    {*linear_inequality_upper_bounds*, 14, 0, 10, 0, 531, 0, 0, 0, 0, 0, 0, *CHOOSE
    linear inequality upper bounds*},
    {*merit_function*, 8, 3, 2, 0, 329, kw_251},
    {*model_pointer*, 11, 0, 5, 0, 1905},
    {*search_method*, 8, 4, 1, 0, 319, kw_252},
    {*steplength_to_boundary*, 10, 0, 5, 0, 337}
CHAPTER 12. NAMESPACE DOCUMENTATION

GuiKeyWord kw_254[5] [static]

Initial value:

= 

* {
  "debug", 8, 0, 1, 1, 89, 0, 0, 0, 0, 0, 0, 0,"[CHOOSE output level]",
  "normal", 8, 0, 1, 1, 93,
  "quiet", 8, 0, 1, 1, 95,
  "silent", 8, 0, 1, 1, 97,
  "verbose", 8, 0, 1, 1, 91
}*

GuiKeyWord kw_255[2] [static]

Initial value:

= 

* {
  "model_pointer", 11, 0, 1, 0, 165, 0, 0, 0, 0, 0, 0,"{identification of model by pointer}
  MethodCommands.html#MethodMetaMultiStart",
  "opt_model_pointer", 3, 0, 1, 0, 164
}*

GuiKeyWord kw_256[1] [static]

Initial value:

= 

* {
  "seed", 0, 0, 1, 0, 171, 0, 0, 0, 0, 0, 0,"{seed for random weighting sets}
  MethodCommands.html#MethodMetaParetoSet"
}*

GuiKeyWord kw_257[10] [static]

Initial value:

= 

* {
  "iterator_scheduling", 8, 2, 5, 0, 177, kw_148, 0, 0, 0, 0, 0,"{message passing configuration for scheduling of iterator jobs} MethodCommands.html#MethodMeta",
  "iterator_servers", 0x19, 0, 4, 0, 175, 0, 0, 0, 0, 0, 0,"{number of iterator servers}
  MethodCommands.html#MethodMeta",
  "method_name", 11, 2, 1, 1, 163, kw_255, 0, 0, 0, 0, 0,"{identification of sub-iterator by name}
  MethodCommands.html#MethodMetaMultiStart",
  "method_pointer", 11, 0, 1, 1, 167, 0, 0, 0, 0, 0, 0,"{identification of sub-iterator by pointer}
  MethodCommands.html#MethodMetaMultiStart",
  "multi_objective_weight_sets", 6, 0, 3, 0, 172,
  "opt_method_name", 3, 2, 1, 1, 162, kw_255,
  "opt_method_pointer", 3, 0, 1, 1, 166,
  "processors_per_iterator", 0x19, 0, 6, 0, 183, 0, 0, 0, 0, 0, 0,"{number of processors per iterator server} MethodCommands.html#MethodMeta",
  "random_weight_sets", 9, 1, 2, 0, 169, kw_256, 0, 0, 0, 0, 0,"{number of random weighting sets}
  MethodCommands.html#MethodMetaParetoSet",
  "weight_sets", 14, 0, 3, 0, 173, 0, 0, 0, 0, 0, 0,"{list of user-specified weighting sets}
  MethodCommands.html#MethodMetaParetoSet"
}*
GuiKeyWord kw_258[4] [static]
Initial value:

```cpp
= {
    {"model_pointer",11,0,2,0,1905},
    {"partitions",13,0,1,0,1643,0,0,,0,,0,,0,0,0,0,0,0,"{Number of partitions} MethodCommands.html#MethodPSUADE"},
    {"samples",9,0,3,0,1645,0,0,,0,,0,,0,0,0,0,0,0,"{Number of samples} MethodCommands.html#MethodNonDMC"},
    {"seed",0x19,0,4,0,1647,0,0,,0,,0,,0,"{Refinement seed} MethodCommands.html#MethodNonLocalRel"}
}
```

GuiKeyWord kw_259[5] [static]
Initial value:

```cpp
= {
    {"converge_order",8,0,1,1,1899},
    {"converge_qoi",8,0,1,1,1901},
    {"estimate_order",8,0,1,1,1897},
    {"model_pointer",11,0,3,0,1905},
    {"refinement_rate",10,0,2,0,1903,0,0,,0,,0,,0,"{Refinement rate} MethodCommands.html#MethodSolnRichardson"}
}
```

GuiKeyWord kw_260[2] [static]
Initial value:

```cpp
= {
    {"num_generations",0x29,0,2,0,473},
    {"percent_change",10,0,1,0,471}
}
```

GuiKeyWord kw_261[2] [static]
Initial value:

```cpp
= {
    {"num_generations",0x29,0,2,0,467,0,0,,0,,0,,0,0,"{Number of generations (for convergence test) } MethodCommands.html#MethodJEGASOGA"},
    {"percent_change",10,0,1,5,465,0,0,,0,,0,,0,"{Percent change in fitness} MethodCommands.html#MethodJEGASOGA"}
}
```

GuiKeyWord kw_262[2] [static]
Initial value:

```cpp
= {
    {"average_fitness_tracker",8,2,1,1,469,kw_260},
    {"best_fitness_tracker",8,2,1,1,463,kw_261}
}
```
GuiKeyWord kw_263[2] [static]
Initial value:

```cpp
= {
    {"constraint_penalty",10,0,2,0,449,0,0,0,0,0,0,"Constraint penalty in merit function"MethodCommands.html#MethodJEGASOGA"},
    {"merit_function",8,0,1,1,447}
}
```

GuiKeyWord kw_264[4] [static]
Initial value:

```cpp
= {
    {"elitist",8,0,1,1,451},
    {"favor_feasible",8,0,1,1,455},
    {"roulette_wheel",8,0,1,1,457},
    {"unique_roulette_wheel",8,0,1,1,459}
}
```

GuiKeyWord kw_265[20] [static]
Initial value:

```cpp
= {
    {"convergence_type",8,2,3,0,461,kw_262,0,0,0,0,0,0,"Convergence type"MethodCommands.html#MethodJEGASOGA"},
    {"crossover_type",8,5,18,0,489,kw_182,0,0,0,0,0,0,"Crossover type"MethodCommands.html#MethodJEGADC"},
    {"fitness_type",8,2,1,0,445,kw_263,0,0,0,0,0,0,"Fitness type"MethodCommands.html#MethodJEGASOGA"},
    {"initialization_type",8,3,17,0,481,kw_183,0,0,0,0,0,0,"Initialization type"MethodCommands.html#MethodJEGADC"},
    {"linear_equality_constraint_matrix",14,0,10,0,533,0,0,0,0,0,"Linear equality coefficient matrix"MethodCommands.html#MethodMin"},
    {"linear_equality_scale_types",15,0,12,0,537,0,0,0,0,0,0,"Linear equality scaling types"MethodCommands.html#MethodMin"},
    {"linear_equality_scales",14,0,13,0,539,0,0,0,0,0,0,"Linear equality scales"MethodCommands.html#MethodMin"},
    {"linear_equality_targets",14,0,11,0,535,0,0,0,0,0,0,"Linear equality targets"MethodCommands.html#MethodMin"},
    {"linear_inequality_constraint_matrix",14,0,5,0,523,0,0,0,0,0,0,"Linear inequality coefficient matrix"MethodCommands.html#MethodMin"},
    {"linear_inequality_lower_bounds",14,0,6,0,525,0,0,0,0,0,0,"Linear inequality lower bounds"MethodCommands.html#MethodMin"},
    {"linear_inequality_scale_types",15,0,9,0,529,0,0,0,0,0,0,"Linear inequality scaling types"MethodCommands.html#MethodMin"},
    {"linear_inequality_scales",14,0,9,0,531,0,0,0,0,0,0,"Linear inequality scales"MethodCommands.html#MethodMin"},
    {"linear_inequality_upper_bounds",14,0,7,0,527,0,0,0,0,0,0,"Linear inequality upper bounds"MethodCommands.html#MethodMin"},
    {"log_file",11,0,15,0,477,0,0,0,0,0,0,"Log file"MethodCommands.html#MethodJEGADC"},
    {"model_pointer",11,0,4,0,1905},
    {"mutation_type",8,6,19,0,505,kw_185,0,0,0,0,0,0,"Mutation type"MethodCommands.html#MethodJEGADC"},
    {"population_size",8,0,29,0,475,0,0,0,0,0,0,"Number of population members"MethodCommands.html#MethodJEGADC"},
    {"print_each_pop",8,0,16,0,479,0,0,0,0,0,0,"Population output"MethodCommands.html#MethodJEGADC"},
    {"replacement_type",8,4,2,0,451,kw_264,0,0,0,0,0,0,"Replacement type"MethodCommands.html#MethodJEGADC"},
    {"seed",0x19,0,26,0,221,0,0,0,0,0,0,"Random seed"MethodCommands.html#MethodJEGADC"}
}
```
GuiKeyWord kw_266[15] [static]
Initial value:

```
= {
    {"function_precision",10,0,13,0,303,0,0.,0.,0.,0.,"Function precision" MethodCommands.html#MethodNPSOLDC"},
    {"linear.equality.constraint_matrix",14,0,8,0,533,0,0.,0.,0.,0.,"Linear equality coefficient matrix" MethodCommands.html#MethodMin"},
    {"linear.equality_scale_types",15,0,10,0,537,0,0.,0.,0.,0.,"Linear equality scaling types" MethodCommands.html#MethodMin"},
    {"linear.equality.scales",14,0,11,0,539,0,0.,0.,0.,0.,"Linear equality scales" MethodCommands.html#MethodMin"},
    {"linear.equality_targets",14,0,9,0,535,0,0.,0.,0.,0.,"Linear equality targets" MethodCommands.html#MethodMin"},
    {"linear.inequality.constraint_matrix",14,0,3,0,523,0,0.,0.,0.,0.,"Linear inequality coefficient matrix" MethodCommands.html#MethodMin"},
    {"linear.inequality_lower.bounds",14,0,4,0,525,0,0.,0.,0.,0.,"Linear inequality lower bounds" MethodCommands.html#MethodMin"},
    {"linear.inequality_scale_types",15,0,6,0,529,0,0.,0.,0.,0.,"Linear inequality scaling types" MethodCommands.html#MethodMin"},
    {"linear.inequality.scales",14,0,7,0,531,0,0.,0.,0.,0.,"Linear inequality scales" MethodCommands.html#MethodMin"},
    {"linear.inequality.upper.bounds",14,0,5,0,527,0,0.,0.,0.,0.,"Linear inequality upper bounds" MethodCommands.html#MethodMin"},
    {"linesearch_tolerance",10,0,14,0,305,0,0.,0.,0.,0.,"Line search tolerance" MethodCommands.html#MethodNPSOLDC"},
    {"model_pointer",11,0,2,0,1905},
    {"nlxso1",8,0,1,1,298},
    {"npso1",8,0,1,1,297},
    {"verify.level",9,0,12,0,301,0,0.,0.,0.,0.,"Gradient verification level" MethodCommands.html#MethodNPSOLDC"}
}
```

GuiKeyWord kw_267[7] [static]
Initial value:

```
= {
    {"approx.method.name",3,0,1,1,250},
    {"approx.method.pointer",3,0,1,1,248},
    {"approx.model_pointer",3,0,2,2,252},
    {"method.name",11,0,1,1,251},
    {"method_pointer",11,0,1,1,249},
    {"model_pointer",11,0,2,2,253},
    {"replace.points",8,0,3,0,255,0,0.,0.,0.,0.,"Replace points used in surrogate construction with best points from previous iteration" MethodCommands.html#MethodSBG"}
}
```

GuiKeyWord kw_268[2] [static]
Initial value:

```
= {
    {"filter",8,0,1,1,241,0,0.,0.,0.,0.,"[CHOOSE acceptance logic]"},
    {"tr_ratio",8,0,1,1,239}
}
```

GuiKeyWord kw_269[7] [static]
Initial value:
= {
    "augmented.lagrangian.objective", 8, 0, 1, 217, 0, 0, 0, 0, 0, "[CHOOSE objective formulation]
    "lagrangian.objective", 8, 0, 1, 219,
    "linearized.constraints", 8, 0, 2, 223, 0, 0, 0, 0, 0, "[CHOOSE constraint formulation]
    "no.constraints", 8, 0, 2, 225,
    "original_constraints", 8, 0, 2, 221, 0, 0, 0, 0, 0, "@",
    "original_primary", 8, 0, 1, 213, 0, 0, 0, 0, 0, "@",
    "single_objective", 8, 0, 1, 215
}

GuiKeyWord kw_270[1] [static]
Initial value:
= {
    "homotopy", 8, 0, 1, 245
}

GuiKeyWord kw_271[4] [static]
Initial value:
= {
    "adaptive_penalty.merit", 8, 0, 1, 231, 0, 0, 0, 0, 0, "[CHOOSE merit function]
    "augmented.lagrangian.merit", 8, 0, 1, 235, 0, 0, 0, 0, 0, "@",
    "lagrangian.merit", 8, 0, 1, 233,
    "penalty.merit", 8, 0, 1, 229
}

GuiKeyWord kw_272[6] [static]
Initial value:
= {
    "contract_threshold", 10, 0, 4, 0, 203, 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, 207, 0, 0, 0, 0, 0, "[Trust region contraction factor]
    MethodCommands.html#MethodSBL",
    "expand_threshold", 10, 0, 4, 0, 205, 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, 209, 0, 0, 0, 0, 0, "[Trust region expansion factor]
    MethodCommands.html#MethodSBL",
    "initial_size", 10, 0, 1, 0, 199, 0, 0, 0, 0, 0, "[Trust region initial size (relative to bounds)]
    MethodCommands.html#MethodSBL",
    "minimum_size", 10, 0, 2, 0, 201, 0, 0, 0, 0, 0, "[Trust region minimum size]
    MethodCommands.html#MethodSBL"
}

GuiKeyWord kw_273[13] [static]
Initial value:
= {
    "acceptance_logic", 8, 2, 8, 0, 237, kw_268, 0, 0, 0, 0, 0, "[SBL iterate acceptance logic]
    MethodCommands.html#MethodSBL",
    "approx.method_name", 3, 0, 1, 188,
    "approx.method_pointer", 3, 0, 1, 186,
    "approx.model_pointer", 3, 0, 2, 190,
    "approx.subproblem", 8, 7, 6, 0, 231, kw_269, 0, 0, 0, 0, 0, "[Approximate subproblem formulation]
    MethodCommands.html#MethodSBL",
    "constraint.relax", 8, 1, 0, 243, kw_270, 0, 0, 0, 0, 0, "[SBL constraint relaxation method for
12.1. DAKOTA NAMESPACE REFERENCE

Methods:

- **Methods**: MethodCommands.html#MethodSBL
  - **merit function**: MethodCommands.html#MethodSBL
  - **method name**: MethodCommands.html#MethodMetaParetoSet
  - **method pointer**: MethodCommands.html#MethodMetaParetoSet
  - **model pointer**: MethodCommands.html#MethodMetaParetoSet
  - **soft convergence limit**: MethodCommands.html#MethodSBL
  - **trust region**: MethodCommands.html#MethodSBL
  - **truth surrogate bypass**: MethodCommands.html#MethodSBL

GuiKeyWord kw.274[4] [static]

Initial value:

```java
= {
  "final point", 14, 0, 1, 1, 1859, 0, 0, 0, 0, 0, 0, "[CHOOSE final pt or increment]";
  "model pointer", 11, 0, 3, 0, 1905,
  "num steps", 9, 0, 2, 2, 1863, 0, 0, 0, 0, 0, 0, "[Number of steps along vector]

GuiKeyWord kw.276[1] [static]

Initial value:

```java
= {
  "optional interface responses pointer", 11, 0, 1, 0, 2153, 0, 0, 0, 0, 0, 0, "[Responses pointer for nested model optional interfaces] ModelCommands.html#ModelNested"
}
```

GuiKeyWord kw.277[2] [static]

Initial value:

```java
= {
  "iterator scheduling", 8, 2, 2, 0, 2159, kw.277,
  "iterator servers", 8, 19, 0, 1, 0, 2157,
```

GuiKeyWord kw.278[7] [static]

Initial value:

```java
= {
  "primary response mapping", 14, 0, 6, 0, 2171, 0, 0, 0, 0, 0, 0, "[Primary response mappings for nested models] ModelCommands.html#ModelNested",
  "primary variable mapping", 15, 0, 4, 0, 2167, 0, 0, 0, 0, 0, 0, "[Primary variable mappings for nested models] ModelCommands.html#ModelNested",
  "processors per iterator", 8, 19, 0, 3, 0, 2165,
  "secondary response mapping", 14, 0, 7, 0, 2173, 0, 0, 0, 0, 0, 0, "[Secondary response mappings for nested models] ModelCommands.html#ModelNested",
  "secondary variable mapping", 15, 0, 5, 0, 2169, 0, 0, 0, 0, 0, 0, "[Secondary variable mappings for nested models] ModelCommands.html#ModelNested"
```
GuiKeyWord kw.279[2]  [static]
Initial value:

```c
= {
    "optional_interface_pointer", 11, 1, 0, 2151, kw.276, 0., 0., 0., 0.,
    "ModelCommands.html#ModelNested"},
    "sub_method_pointer", 11, 2, 1, 2155, kw.278, 0., 0., 0., 0.,
    "ModelCommands.html#ModelNested"
}
```

GuiKeyWord kw.280[1]  [static]
Initial value:

```c
= {
    "interface_pointer", 11, 0, 1, 0, 1919, 0., 0., 0., 0.,
    "ModelCommands.html#ModelSingle"
}
```

GuiKeyWord kw.281[3]  [static]
Initial value:

```c
= {
    "eval.id", 8, 0, 2, 0, 2111},
    "header", 8, 0, 1, 0, 2109},
    "interface.id", 8, 0, 3, 0, 2113
}
```

GuiKeyWord kw.282[4]  [static]
Initial value:

```c
= {
    "active_only", 8, 0, 2, 0, 2117},
    "annotated", 8, 0, 1, 0, 2105},
    "custom_annotated", 8, 3, 1, 0, 2107, kw.281},
    "freeform", 8, 0, 1, 0, 2115
}
```

GuiKeyWord kw.283[6]  [static]
Initial value:

```c
= {
    "additive", 8, 0, 2, 2, 2087, 0., 0., 0., 0.,
    "first.order", 8, 0, 1, 1, 2083, 0., 0., 0., 0.,
    "multiplicative", 8, 0, 2, 2, 2089],
    "second.order", 8, 0, 1, 1, 2085],
    "zeroth.order", 8, 0, 1, 1, 2081
}
```
GuiKeyWord kw_284[2]  [static]
Initial value:
    = {
        {"folds",0,0,1,0,2097,0,0,0,0,0,0,"{Number cross validation folds}
        ModelCommands.html(ModelSurrG)*},
        {"percent",10,0,1,0,2099,0,0,0,0,0,0,"{Percent points per CV fold}
        ModelCommands.html(ModelSurrG)*} }

GuiKeyWord kw_285[2]  [static]
Initial value:
    = {
        {"cross_validation",8,2,1,0,2095,kw_284[0,0,0,0,0,0,0,"{Perform cross validation}
        ModelCommands.html(ModelSurrG)*},
        {"press",8,0,2,0,2101,5,0,0,0,0,0,"{Perform PRESS cross validation}
        ModelCommands.html(ModelSurrG)*} }

GuiKeyWord kw_286[3]  [static]
Initial value:
    = {
        {"eval_id",8,0,2,0,2071},
        {"header",8,0,1,0,2069},
        {"interface_id",8,0,3,0,2073} }

GuiKeyWord kw_287[3]  [static]
Initial value:
    = {
        {"annotated",8,0,1,0,2065},
        {"custom.annotated",8,3,1,0,2067,kw_286},
        {"freeform",8,0,1,0,2075} }

GuiKeyWord kw_288[3]  [static]
Initial value:
    = {
        {"constant",8,0,1,1,1935},
        {"linear",8,0,1,1,1937},
        {"reduced_quadratic",8,0,1,1,1939} }

GuiKeyWord kw_289[2]  [static]
Initial value:
    = {
        {"point_selection",8,0,1,0,1931,0,0,0,0,0,0,"{GP point selection}
        ModelCommands.html(ModelSurrG)*},
        {"trend",8,3,2,0,1933,kw_288,0,0,0,0,0,"{GP trend function} ModelCommands.html(ModelSurrG)*} }
GuiKeyWord kw_290[4] [static]
Initial value:
= {
    "constant",8,0,1,1,1945},
    "linear",8,0,1,1,1947},
    "quadratic",8,0,1,1,1951},
    "reduced_quadratic",8,0,1,1,1949}
}

GuiKeyWord kw_291[7] [static]
Initial value:
= {
    "correlation_lengths",14,0,5,0,1961,0,0.,0.,0.,0,"{Surfpack GP correlation lengths} ModelCommands.html#ModelSurr"},
    "export_model_file",11,0,6,0,1963},
    "find_nugget",9,0,4,0,1959,0,0.,0.,0.,0,"{Surfpack finds the optimal nugget } Model Commands.html#ModelSurr"},
    "max_trials",0x19,0,3,0,1955,0,0.,0.,0.,0,"{Surfpack GP maximum trials} ModelCommands.html#ModelSurr"},
    "nugget",0x1a,0,4,0,1957,0,0.,0.,0.,0,"{Surfpack user-specified nugget } ModelCommands.html#ModelSurr"},
    "optimization_method",11,0,2,0,1953,0,0.,0.,0.,0,"{Surfpack GP optimization method} ModelCommands.html#ModelSurr"},
    "trend",8,4,1,0,1943,kw_290,0.,0.,0.,0,"{Surfpack GP trend function} ModelCommands.html#ModelSurr"}
)

GuiKeyWord kw_292[2] [static]
Initial value:
= {
    "dakota",8,2,1,1,1929,kw_289},
    "surfpack",8,7,1,1,1941,kw_291}
)

GuiKeyWord kw_293[3] [static]
Initial value:
= {
    "eval_id",8,0,2,0,2055},
    "header",8,0,1,0,2053},
    "interface_id",8,0,3,0,2057}
)

GuiKeyWord kw_294[4] [static]
Initial value:
= {
    "active_only",8,0,2,0,2061},
    "annotated",8,0,1,0,2049,0,0.,0.,0.,0,"{Challenge file in annotated format} ModelCommands.html#ModelSurr"},
    "custom_annotated",8,3,1,0,2051,kw_293},
    "freeform",8,0,1,0,2059,0,0.,0.,0.,0,"{Challenge file in freeform format} ModelCommands.html#ModelSurr"}
)
GuiKeyWord kw_295[2] [static]
Initial value:

= {
    {"cubic",8,0,1,1,1973},
    {"linear",8,0,1,1,1971}
}

GuiKeyWord kw_296[3] [static]
Initial value:

= {
    {"export_model_file",11,0,3,0,1975},
    {"interpolation",8,2,2,0,1969,kw_295,0,,0,,0,"{MARS interpolation} ModelCommands.html#ModelSurfG"},
    {"max_bases",9,0,1,0,1967,0,,0,,0,"{MARS maximum bases} ModelCommands.html#ModelSurfG"}
}

GuiKeyWord kw_297[3] [static]
Initial value:

= {
    {"export_model_file",11,0,3,0,1983},
    {"poly_order",9,0,1,0,1979,0,,0,,0,"{MLS polynomial order} ModelCommands.html#ModelSurfG"},
    {"weight_function",9,0,2,0,1981,0,,0,,0,"{MLS weight function} ModelCommands.html#ModelSurfG"}
}

GuiKeyWord kw_298[5] [static]
Initial value:

= {
    {"export_model_file",11,0,4,0,1993},
    {"max_nodes",9,0,1,0,1987},
    {"nodes",1,0,1,0,1996},
    {"random_weight",9,0,3,0,1991,0,,0,,0,"{ANN random weight} ModelCommands.html#ModelSurfG"},
    {"range",10,0,2,0,1989,0,,0,,0,"{ANN range} ModelCommands.html#ModelSurfG"}
}

GuiKeyWord kw_299[2] [static]
Initial value:

= {
    {"gradient_threshold",10,0,1,1,2027},
    {"jump_threshold",10,0,1,1,2025}
GuiKeyWord kw_300[3] [static]
Initial value:
= {
    "cell_type",11,0,1,0,2019,
    "discontinuity_detection",8,2,3,0,2023,kw_299,
    "support_layers",9,0,2,0,2021
}

GuiKeyWord kw_301[4] [static]
Initial value:
= {
    "cubic",8,0,1,1,2013,0,0,0,0,0,"[CHOOSE polynomial order]",
    "export_model_file",11,0,2,0,2015,
    "linear",8,0,1,1,2009,
    "quadratic",8,0,1,1,2011
}

GuiKeyWord kw_302[5] [static]
Initial value:
= {
    "bases",9,0,1,0,1997,0,0,0,0,0,0,"[RBF number of bases] ModelCommands.html#ModelSurrG",
    "export_model_file",11,0,5,0,2005,
    "max_bts",9,0,2,0,1995,0,0,0,0,0,0,"[RBF maximum points] ModelCommands.html#ModelSurrG",
    "max_subsets",9,0,4,0,2003,
    "min_partition",9,0,3,0,2001,0,0,0,0,0,0,"[RBF minimum partitions] ModelCommands.html#ModelSurrG"
}

GuiKeyWord kw_303[3] [static]
Initial value:
= {
    "all",8,0,1,1,2041,
    "none",8,0,1,1,2045,
    "region",8,0,1,1,2043
}

GuiKeyWord kw_304[23] [static]
Initial value:
= {
    "actual_model_pointer",11,0,4,0,2037,0,0,0,0,0,0,"[Pointer to the truth model specification] ModelCommands.html#ModelSurrG",
    "challenge_points_file",11,4,11,0,2103,kw_282,0,0,0,0,"[Challenge file for surrogate metrics] ModelCommands.html#ModelSurrG",
    "correction",8,6,9,0,2079,kw_283,0,0,0,0,"[Surrogate correction approach] ModelCommands.html#ModelSurrG",
    "dace_method_pointer",11,0,4,0,2035,0,0,0,0,0,0,"[Design of experiments method pointer] ModelCommands.html#ModelSurrG",
    "diagnostics",11,2,10,0,2092,kw_285,
    "export_points_file",11,3,7,0,2063,kw_287,0,0,0,0,0,"[File export of global approximation-based sample results] ModelCommands.html#ModelSurrG",
    "gaussian_process",8,2,1,1,1927,kw_292,0,0,0,0,0,"[CHOOSE surrogate type] Dakota Gaussian process ModelCommands.html#ModelSurrG"}
12.1. DAKOTA NAMESPACE REFERENCE

{import_points_file";11,4,6,0,2047,kw,294,0.,0.,0.,0,"{File import of samples for global approximation builds}
ModelCommands.html#ModelSurrG"},
{"kriging","0,2,1,1,1926,kw,294,0.,0.,0.,0","{Multivariate adaptive regression splines}
ModelCommands.html#ModelSurrG"},
{"mars","8,3,1,1,1977,kw,297,0.,0.,0.,0","{Moving least squares}
ModelCommands.html#ModelSurrG"},
{"metrics","15,2,10,0,2093,kw,285,0.,0.,0.,0","{Compute surrogate diagnostics}
ModelCommands.html#ModelSurrG"},
{"minpoints","8,0,3,0,2031"},
{"moving_least_squares","8,3,1,1,1977,kw,297,0.,0.,0.,0","{Moving least squares}
ModelCommands.html#ModelSurrG"},
{"neural_network","8,5,1,1,1985,kw,298,0.,0.,0.,0","{Artificial neural network}
ModelCommands.html#ModelSurrG"},
{"piecewise_decomposition","8,3,2,0,2017,kw,300"},
{"polynomial","8,4,1,1,2007,kw,296,0.,0.,0.,0","{Polynomial}
ModelCommands.html#ModelSurrG"},
{"radial basis","8,5,1,1,1995,kw,302"},
{"recommended_points","8,0,3,0,2033"},
{"reuse_points","8,3,5,0,2039,kw,303"},
{"reuse_samples","0,3,5,0,2038,kw,303"},
{"samples_file","3,4,6,0,2046,kw,294"},
{"total_points","9,0,3,0,2029"},
{"use_derivatives","8,0,6,0,2077,0.,0.,0.,0","{Surpack GP gradient enhancement}
ModelCommands.html#ModelSurrG"}]

GuiKeyword kw_305[6] [static]

Initial value:

= {
    "additive","8,0,2,2,2143,0.,0.,0.,0.,0","{Choose correction type}"
    "combined","8,0,2,2,2147"
    "first_order","8,0,1,1,2139,0.,0.,0.,0.,0","{Choose correction order}"
    "multiplicative","8,0,2,2,2145"
    "second_order","8,0,1,1,2141"
    "zeroth_order","8,0,1,1,2137"}

GuiKeyword kw_306[3] [static]

Initial value:

= {
    "correction","8,6,3,3,2135,kw,305,0.,0.,0.,0.,0","{Surrogate correction approach}
ModelCommands.html#ModelSurrH"},
    "high_fidelity_model_pointer","11,0,2,2,2133,0.,0.,0.,0.,0","{Pointer to the high fidelity model specification}
ModelCommands.html#ModelSurrH"},
    "low_fidelity_model_pointer","11,0,1,1,2131,0.,0.,0.,0.,0","{Pointer to the low fidelity model specification}
ModelCommands.html#ModelSurrH"}

GuiKeyword kw_307[2] [static]

Initial value:

= {
    "actual_model_pointer","11,0,2,2,2127,0.,0.,0.,0.,0","{Pointer to the truth model specification}
ModelCommands.html#ModelSurrL"},
    "taylor_series","8,0,1,1,2125,0.,0.,0.,0.,0","{Taylor series local approximation}
ModelCommands.html#ModelSurrL"}
GuiKeyWord kw_308[2] [static]
Initial value:
= {
    {"actual_model_pointer",11,0,2,2,2127,0,0,0,0,0,0,0,0,"[Pointer to the truth model specification] ModelCommands.html#ModelSurrl"},
    {"tana",8,0,1,1,2121,0,0,0,0,0,0,0,0,"[Two-point adaptive nonlinear approximation ] ModelCommands.html#ModelSurrMP"}
}

GuiKeyWord kw_309[5] [static]
Initial value:
= {
    {"global",8,23,2,1,1925,kw_304,0,0,0,0,0,"[CHOOSE surrogate category] ModelCommands.html#ModelSurrG"},
    {"hierarchical",8,3,2,1,2129,kw_306,0,0,0,0,0,"[Hierarchical approximation ] ModelCommands.html#ModelSurrMP"},
    {"id_surrogates",13,0,1,0,1923,0,0,0,0,0,0,"[Surrogate response ids ModelCommands.html#ModelSurrogate"},
    {"local",8,2,2,1,2123,kw_307,0,0,0,0,0,"[Local approximation ] ModelCommands.html#ModelSurrL"},
    {"multipoint",8,2,2,1,2119,kw_308,0,0,0,0,0,"[Multipoint approximation ] ModelCommands.html#ModelSurrMP"}
}

GuiKeyWord kw_310[7] [static]
Initial value:
= {
    {"hierarchical_tagging",8,0,4,0,1915,0,0,0,0,0,"[Hierarchical evaluation tags] ModelCommands.html#ModelIndControl"},
    {"id_model",11,0,1,0,1909,0,0,0,0,0,0,"[Model set identifier] Model Commands.html#ModelIndControl"},
    {"nested",8,2,5,1,2149,kw_309,0,0,0,0,0,"[CHOOSE model type]"},
    {"responses_pointer",11,0,3,0,1913,0,0,0,0,0,0,"[Responses set pointer] ModelCommands.html#ModelIndControl"},
    {"single",8,1,5,1,1917,kw_388,0,0,0,0,0,"@"},
    {"surrogate",8,5,5,1,1921,kw_309},
    {"variables_pointer",11,0,2,0,1911,0,0,0,0,0,0,"[Variables set pointer] ModelCommands.html#ModelIndControl"}
}

GuiKeyWord kw_311[2] [static]
Initial value:
= {
    {"exp_id",8,0,2,0,2817},
    {"header",8,0,1,0,2815}
}

GuiKeyWord kw_312[3] [static]
Initial value:
= {
    {"annotated",8,0,1,0,2811,0,0,0,0,0,0,0,0,"[Data file in annotated format ] RespCommands.html#RespFnLS"},
    {"custom_annotated",8,2,1,0,2813,kw_311},
    {"freeform",8,0,1,0,2819,0,0,0,0,0,0,0,0,"[Data file in freeform format ] RespCommands.html#RespFnLS"}
}
GuiKeyWord kw_313[6] [static]
Initial value:
= {
    "interpolate", 8, 0, 5, 0, 2821,
    "num_config_variables", 0x29, 0, 2, 0, 2805, 0, 0, 0, 0, 0, 0,"{Configuration variable columns in file} RespCommands.html#RespFnLS",
    "num_experiments", 0x29, 0, 1, 0, 2803, 0, 0, 0, 0, 0, 0,"{Experiments in file} RespCommands.html#RespFnLS",
    "freefield_coordinates", 8, 0, 6, 0, 2823,
    "scalar_data_file", 11, 3, 4, 0, 2809, kw_312,
    "variance_type", 0x80f, 0, 3, 0, 2807, 0, 0, 0, 0, 0, 0, 0,"field.calibration_terms"
}

GuiKeyWord kw_314[2] [static]
Initial value:
= {
    "exp_id", 8, 0, 2, 0, 2833,
    "header", 8, 0, 1, 0, 2831
}

GuiKeyWord kw_315[6] [static]
Initial value:
= {
    "annotated", 8, 0, 1, 0, 2827,
    "custom_annotated", 8, 2, 1, 0, 2829, kw_314,
    "freeform", 8, 0, 1, 0, 2835,
    "num_config_variables", 0x29, 0, 3, 0, 2839,
    "num_experiments", 0x29, 0, 2, 0, 2837,
    "variance_type", 0x80f, 0, 4, 0, 2841, 0, 0, 0, 0, 0, 0, 0,"calibration_terms"
}

GuiKeyWord kw_316[4] [static]
Initial value:
= {
    "coordinate_data_file", 11, 0, 3, 0, 2793,
    "coordinate_list", 14, 0, 3, 0, 2791,
    "lengths", 13, 0, 1, 1, 2787, 0, 0, 0, 0, 0, 0, 0,"calibration_terms",
    "num_coordinates_per_field", 13, 0, 2, 0, 2789
}

GuiKeyWord kw_317[6] [static]
Initial value:
= {
    "nonlinear_equality_scale_types", 0x807, 0, 2, 0, 2856, 0, 0, 0, 0, 0, 0, 0,"nonlinear_equality_constraints",
    "nonlinear_equality_scales", 0x806, 0, 3, 0, 2858, 0, 0, 0, 0, 0, 0,"nonlinear_equality_constraints",
    "nonlinear_equality_targets", 6, 0, 1, 0, 2854, 0, 0, 0, 0, 0, 0,"nonlinear_equality_constraints",
    "scales", 0x80e, 0, 3, 0, 2859, 0, 0, 0, 0, 0, 0,"nonlinear_equality_constraints",
    "targets", 14, 0, 1, 0, 2855, 0, 0, 0, 0, 0,"{Nonlinear equality targets} RespCommands.html#RespFnLS",
    "nonlinear_equality_constraints"
}
GuiKeyWord kw_318[8]  [static]

Initial value:

```
= {
    {"lower_bounds",14,0,1,0,2845,0,0.,0.,0.,0.,0.,0.},{"Nonlinear inequality lower bounds"}
    RespCommands.html#RespFnLS",0,"nonlinear_inequality_constraints"},
    {"nonlinear_inequality_lower_bounds",6,0,1,0,2844,0,0.,0.,0.,0.,0.,0.,0.},{"Nonlinear inequality lower bounds"}
    nonlinear.inequality_constraints"},
    {"nonlinear_inequality_scale_types",0x807,0,3,0,2848,0,0.,0.,0.,0.,0.,0.},{"Nonlinear inequality scale types"}
    nonlinear_inequality_constraints"},
    {"nonlinear_inequality_scales",0x806,0,4,0,2851,0,0.,0.,0.,0.,0.,0.},{"Nonlinear inequality scales"}
    nonlinear_inequality_constraints"},
    {"nonlinear_inequality_upper_bounds",6,0,2,0,2846,0,0.,0.,0.,0.,0.,0.},{"Nonlinear inequality upper bounds"}
    RespCommands.html#RespFnLS",0,"nonlinear_inequality_constraints"}
}
```

GuiKeyWord kw_319[18]  [static]

Initial value:

```
= {
    {"calibration_data",8,6,6,0,2801,kw_313},{"calibration data"}
    RespCommands.html#RespFnLS"},
    {"calibration_data_file",11,6,6,0,2825,kw_315,0.,0.,0.,0.},{"calibration data file name"}
    RespCommands.html#RespFnLS"},
    {"calibration_term_scale_types",0x807,0,3,0,2796,0,0.,0.,0.,0.,0.,0.},{"calibration term scale types"}
    RespCommands.html#RespFnLS"},
    {"calibration_term_scales",0x806,0,4,0,2798,0,0.,0.,0.,0.,0.,0.},{"calibration term scales"}
    RespCommands.html#RespFnLS"},
    {"calibration_weights",6,0,5,0,2799,0,0.,0.,0.,0.,0.,0.},{"calibration weights"}
    RespCommands.html#RespFnLS"},
    {"field_calibration_terms",0x29,4,2,0,2785,kw_316},{"field calibration terms"}
    RespCommands.html#RespFnLS"},
    {"least_squares_term_scale_types",0x807,0,3,0,2794,0,0.,0.,0.,0.,0.,0.},{"least squares term scale types"}
    RespCommands.html#RespFnLS"},
    {"least_squares_term_scales",0x806,0,4,0,2796,0,0.,0.,0.,0.,0.,0.},{"least squares term scales"}
    RespCommands.html#RespFnLS"},
    {"least_squares_weights",6,0,5,0,2798,0,0.,0.,0.,0.,0.,0.},{"least squares weights"}
    RespCommands.html#RespFnLS"},
    {"num_nonlinear_inequality_constraints",0x21,6,8,0,2853,kw_317,0.,0.,0.},{"Number of nonlinear inequality constraints"}
    RespCommands.html#RespFnLS"},
    {"num_nonlinear_inequality_constraints",0x21,8,7,0,2843,kw_318,0.,0.,0.},{"Number of nonlinear inequality constraints"}
    RespCommands.html#RespFnLS"},
    {"primary_scale_types",0x807,0,3,0,2795,0,0.,0.,0.,0.,0.,0.},{"primary scale types"}
    RespCommands.html#RespFnLS"},
    {"primary_scales",0x806,0,4,0,2797,0,0.,0.,0.,0.,0.,0.},{"primary scales"}
    RespCommands.html#RespFnLS"},
    {"scalar_calibration_terms",0x29,9,0,1,0,2783},{"Scalar calibration terms"}
    RespCommands.html#RespFnLS"},
    {"weights",14,0,5,0,2799,0,0.,0.,0.,0.},{"weights"}
    RespCommands.html#RespFnLS"},
}
```

GuiKeyWord kw_320[4]  [static]

Initial value:

```
= {
    {"absolute",8,0,2,0,2895},{"absolute"}
    RespCommands.html#RespGradMixed"},
    {"bounds",8,0,2,0,2897},{"variable bounds"}
    RespCommands.html#RespGradMixed"},
    {"ignore_bounds",8,0,1,0,2891,0,0.,0.,0.,0.},{"Ignore variable bounds"}
    RespCommands.html#RespGradMixed"},
    {"relative",8,0,2,0,2893},{"relative"}
    RespCommands.html#RespGradMixed"},
}
```
GuiKeyWord kw_321[10] [static]
Initial value:

= {
  {
    "central", 8, 0, 6, 0, 2905, 0, 0, 0, 0, 0, 0, "[CHOOSE difference interval]"},
  {
    "dakota", 8, 4, 4, 0, 2889, kw_320, 0, 0, 0, 0, 0, "[CHOOSE gradient source]"},
  {
    "fd_gradient_step_size", 6, 0, 7, 0, 2906},
  {
    "fd_step_size", 14, 0, 7, 0, 2907, 0, 0, 0, 0, 0, 0, "[Finite difference step size]"},
  {
    "id_analytic_gradients", 13, 0, 2, 2, 2883, 0, 0, 0, 0, 0, 0, "[Analytic derivatives function list]"},
  {
    "id_numerical_gradients", 13, 0, 1, 1, 2881, 0, 0, 0, 0, 0, 0, "[Numerical derivatives function list]"},
  {
    "interval_type", 8, 0, 5, 0, 2901, 0, 0, 0, 0, 0, 0, "[Interval type]"},
  {
    "method_source", 8, 0, 3, 0, 2887, 0, 0, 0, 0, 0, 0, "[Method source]"},
  {
    "vendor", 8, 0, 4, 0, 2899}
}

GuiKeyWord kw_322[2] [static]
Initial value:

= {
  {
    "fd_hessian_step_size", 6, 0, 1, 0, 2938},
  {
    "fd_step_size", 14, 0, 1, 0, 2939, 0, 0, 0, 0, 0, 0, "[Finite difference step size]"},
  {
    "interval_type", 8, 0, 5, 0, 2901, 0, 0, 0, 0, 0, 0, "[Interval type]"},
  {
    "method_source", 8, 0, 3, 0, 2887, 0, 0, 0, 0, 0, 0, "[Method source]"},
  {
    "vendor", 8, 0, 4, 0, 2899}
}

GuiKeyWord kw_323[1] [static]
Initial value:

= {
  {
    "damped", 8, 0, 1, 0, 2955, 0, 0, 0, 0, 0, 0, "[Numerical safeguarding of BFGS update]"},
  {
    "interval_type", 8, 0, 5, 0, 2901, 0, 0, 0, 0, 0, 0, "[Interval type]"}
}

GuiKeyWord kw_324[2] [static]
Initial value:

= {
  {
    "bfgs", 8, 1, 1, 2953},
  {
    "sr1", 8, 0, 1, 1, 2957}
}

GuiKeyWord kw_325[8] [static]
Initial value:

= {
  {
    "absolute", 8, 0, 2, 0, 2943},
  {
    "bounds", 8, 0, 2, 0, 2945},
  {
    "central", 8, 0, 3, 0, 2949, 0, 0, 0, 0, 0, 0, "[CHOOSE difference interval]"},
  {
    "forward", 8, 0, 3, 0, 2947, 0, 0, 0, 0, 0, 0, "[CHOOSE difference interval]"}
  {
    "id_analytic_hessians", 13, 0, 5, 0, 2959, 0, 0, 0, 0, 0, 0, "[Analytic Hessians function list]"},
  {
    "interval_type", 8, 0, 5, 0, 2901, 0, 0, 0, 0, 0, 0, "[Interval type]"},
  {
    "method_source", 8, 0, 3, 0, 2887, 0, 0, 0, 0, 0, 0, "[Method source]"},
  {
    "vendor", 8, 0, 4, 0, 2899}
CHAPTER 12. NAMESPACE DOCUMENTATION

{"id:numerical_hessians","13,2,1,0,2937,kw,322,0,0,0,0,0,0","{Numerical Hessians function list} RespCommands.html#RespHessMixed"},
{"id:quasi_hessians","13,2,4,0,2951,kw,324,0,0,0,0,0,0,"[Quasi Hessians function list] RespCommands.html#RespHessMixed"},
{"relative",8,0,2,0,2941}

GuiKeyWord kw_326[4] [static]
Initial value:

= {
    {"coordinate_data_file",11,0,3,0,2779},
    {"coordinate_list",14,0,3,0,2777},
    {"lengths",13,0,1,1,2773},
    {"num_coordinates_per_field",13,0,2,0,2775}
}

GuiKeyWord kw_327[6] [static]
Initial value:

= {
    {"nonlinear_equality_scale_types",0x807,0,2,0,2764,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"nonlinear_equality_scales",0x806,0,3,0,2766,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"nonlinear_equality_targets",6,0,1,0,2762,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"nonlinear_equality_scales",0x806,0,3,0,2767,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"nonlinear_equality_upper_bounds",6,0,2,0,2763,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"targets",14,0,1,0,2761,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"nonlinear_equality_scale_types",0x807,0,3,0,2756,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"nonlinear_equality_scales",0x806,0,4,0,2758,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"nonlinear_equality_upper_bounds",6,0,2,0,2754,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"scale_types",0x80f,0,2,0,2752,0,0,0,0,0,0,"nonlinear_equality_constraints"},
    {"targets",14,0,1,0,2761,0,0,0,0,0,0,"nonlinear_equality_constraints"}
}

GuiKeyWord kw_328[8] [static]
Initial value:

= {
    {"lower_bounds",14,0,1,0,2753,0,0,0,0,0,0,"{Nonlinear inequality constraint lower bounds} RespCommands.html#RespFnOpt"},
    {"nonlinear_inequality_lower_bounds",6,0,1,0,2752,0,0,0,0,0,0,"nonlinear_inequality_constraints"},
    {"nonlinear_inequality_scale_types",0x807,0,3,0,2756,0,0,0,0,0,0,"nonlinear_inequality_constraints"},
    {"nonlinear_inequality_scales",0x806,0,4,0,2758,0,0,0,0,0,0,"nonlinear_inequality_constraints"},
    {"nonlinear_inequality_upper_bounds",6,0,2,0,2754,0,0,0,0,0,0,"nonlinear_inequality_constraints"},
    {"scale_types",0x80f,0,3,0,2757,0,0,0,0,0,0,"{Nonlinear constraint scaling types (for inequalities or equalities) RespCommands.html#RespFnOpt"},
    {"scales",0x80e,0,4,0,2759,0,0,0,0,0,0,"{Nonlinear constraint scales (for inequalities or equalities) RespCommands.html#RespFnOpt"},
    {"upper_bounds",14,0,2,0,2755,0,0,0,0,0,0,"{Nonlinear inequality constraint upper bounds} RespCommands.html#RespFnOpt"},
    {"nonlinear_inequality_constraints"}
GuiKeyWord kw_329[15] [static]
Initial value:
= {
  {"field_objectives", 0x29, 4, 8, 0, 2771, kw_326},
  {"multi_objective_weights", 0x29, 4, 0, 2748, 0, 0, 0, 0, 0, 0, 0, 0,"objective_functions"},
  {"nonlinear_inequality_constraints", 0x29, 4, 0, 2761, kw_327, 0, 0, 0, 0, 0, 0, 0, 0, 0,"{Number of nonlinear
inequality constraints} RespCommands.html#RespFnOpt"},
  {"num_field_objectives", 0x21, 4, 8, 0, 2770, kw_326},
  {"num_nonlinear_inequality_constraints", 0x21, 6, 6, 0, 2760, kw_327},
  {"num_nonlinear_inequality_constraints", 0x21, 8, 5, 0, 2750, kw_328},
  {"number_of_nonlinear_inequality_constraints", 0x21, 0, 7, 0, 2768},
  {"objective_function_scale_types", 0x807, 0, 2, 0, 2744, 0, 0, 0, 0, 0, 0, 0, 0, 0,"objective functions"},
  {"objective_function_scales", 0x806, 0, 3, 0, 2746, 0, 0, 0, 0, 0, 0, 0, 0, 0,"objective functions"},
  {"primary_scale_types", 0x80f, 0, 2, 0, 2745, 0, 0, 0, 0, 0, 0, 0, 0, 0,"{Objective function scaling types} RespCommands.html#RespFnOpt"},
  {"primary_scales", 0x80e, 0, 3, 0, 2747, 0, 0, 0, 0, 0, 0, 0, 0, 0,"{Objective function scales} RespCommands.html#RespFnOpt"},
  {"weights", 14, 0, 4, 0, 2749, 0, 0, 0, 0, 0,"{Multi-objective weightings} RespCommands.html#RespFnOpt"}
}

GuiKeyWord kw_330[4] [static]
Initial value:
= {
  {"coordinate_data_file", 11, 0, 3, 0, 2873},
  {"coordinate_list", 14, 0, 3, 0, 2871},
  {"lengths", 13, 0, 1, 1, 2867},
  {"num_coordinates_per_field", 13, 0, 2, 0, 2869}
}

GuiKeyWord kw_331[4] [static]
Initial value:
= {
  {"field_responses", 0x29, 4, 2, 0, 2865, kw_330},
  {"num_field_responses", 0x29, 4, 2, 0, 2864, kw_330},
  {"num_scalar_responses", 0x29, 0, 1, 0, 2862},
  {"scalar_responses", 0x29, 5, 1, 0, 2863}
}

GuiKeyWord kw_332[8] [static]
Initial value:
= {
  {"central", 8, 0, 6, 0, 2905, 0, 0, 0, 0, 0, 0,"{CHOOSE difference interval}"},
  {"dakota", 8, 4, 0, 2889, kw_329, 0, 0, 0, 0, 0, 0,"{CHOOSE gradient source}"},
  {"fd_gradient_step_size", 6, 0, 7, 0, 2906},
  {"fd_step_size", 14, 0, 7, 0, 2907, 0, 0, 0, 0, 0, 0,"{Finite difference step size} RespCommands.html#RespGradNum"},
  {"forward", 8, 0, 6, 0, 2903, 0, 0, 0, 0, 0, 0,"{CHOOSE gradient source}"},
  {"interval_type", 8, 0, 5, 0, 2901, 0, 0, 0, 0, 0, 0,"{Interval type} RespCommands.html#RespGradNum"},
  {"method_source", 8, 0, 3, 0, 2887, 0, 0, 0, 0, 0, 0,"{Method source} RespCommands.html#RespGradNum"},
  {"vendor", 8, 0, 4, 0, 2899}
}
GUIKeyword kw_333[7] [static]
Initial value:
= {
    {"absolute",8,0,2,0,2017},
    {"bounds",8,0,2,0,2919},
    {"central",8,0,3,0,2923,0,0,0,0,0,0,"[CHOOSE difference interval]"},
    {"fd_hessian_step_size",6,0,1,0,2912},
    {"fd_step_size",14,0,1,0,2913,0,0,0,0,0,0,"[Finite difference step size]
RespCommands.html#RespHessNum"},
    {"forward",8,0,3,0,2921,0,0,0,0,0,0,"@"},
    {"relative",8,0,2,0,2915}
}

GUIKeyword kw_334[1] [static]
Initial value:
= {
    {"damped",8,0,1,0,2929,0,0,0,0,0,0,"{Numerical safeguarding of BFGS update}
RespCommands.html#RespHessQuasi"}}

GUIKeyword kw_335[2] [static]
Initial value:
= {
    {"bfgs",8,1,1,2927,kw_334,0,0,0,0,0,"[CHOOSE Hessian approx.]"},
    {"sr1",8,0,1,0,2931}}

GUIKeyword kw_336[19] [static]
Initial value:
= {
    {"analytic_gradients",8,0,4,2,2877,0,0,0,0,0,0,"[CHOOSE gradient type]"},
    {"analytic_hessians",8,0,5,3,2933,0,0,0,0,0,0,"[CHOOSE Hessian type]"},
    {"calibration_terms",0x29,18,3,1,2781,kw_319,0,0,0,0,0,0,"{Calibration (Least squares) Number of calibration terms} RespCommands.html#RespFnLS"},
    {"descriptors",15,0,2,0,2739,0,0,0,0,0,0,"{Response descriptors} RespCommands.html#RespLabels"},
    {"id_responses",11,0,1,0,2737,0,0,0,0,0,0,"{Responses set identifier} RespCommands.html#RespSetId"},
    {"least_squares_terms",0x21,18,3,1,2780,kw_319},
    {"mixed_gradients",8,10,4,2,2879,kw_321,0,0,0,0,0,0,"[Mixed gradients]
RespCommands.html#RespGradMixed"},
    {"mixed_hessians",8,8,5,3,2935,kw_325,0,0,0,0,0,0,"[Mixed Hessians]
RespCommands.html#RespGradMixed"},
    {"no_gradients",8,0,4,2,2875,0,0,0,0,0,0,"@"},
    {"no_hessians",8,0,5,3,2909,0,0,0,0,0,0,"@"},
    {"num_least_squares_terms",0x21,18,3,1,2780,kw_319},
    {"num_objective_functions",0x21,15,3,1,2740,kw_329},
    {"num_response_functions",0x21,4,3,1,2860,kw_331},
    {"numerical_gradients",8,8,4,2,2885,kw_332,0,0,0,0,0,0,"[Numerical gradients]
RespCommands.html#RespGradNum"},
    {"numerical_hessians",8,7,5,3,2911,kw_333,0,0,0,0,0,0,"[Numerical Hessians]
RespCommands.html#RespHessNum"},
    {"objective_functions",0x29,15,3,1,2741,kw_329,0,0,0,0,0,0,"{Optimization} Number of objective functions} RespCommands.html#RespFnOpt"},
    {"quasi_hessians",8,2,5,3,2925,kw_335,0,0,0,0,0,0,"[Quasi Hessians]
RespCommands.html#RespHessQuasi"},
    {"response_descriptors",7,0,2,0,2738},
    {"response_functions",0x29,4,3,1,2861,kw_331,0,0,0,0,0,0,"{Generic responses} Number of response functions} RespCommands.html#RespFnGen"}
GuiKeyWord kw_337[6] [static]
Initial value:

= {
    {"aleatory",8,0,1,1,2187},
    {"all",8,0,1,1,2181},
    {"design",8,0,1,1,2183},
    {"epistemic",8,0,1,1,2189},
    {"state",8,0,1,1,2191},
    {"uncertain",8,0,1,1,2185}
}

GuiKeyWord kw_338[11] [static]
Initial value:

= {
    {"alphas",14,0,1,2339,0,0,0,0,0,"{beta uncertain alphas} VarCommands.html#VarCAUV_Beta"},
    {"betas",14,0,2,2341,0,0,0,0,0,"{beta uncertain betas} VarCommands.html#VarCAUV_Beta"},
    {"buv_alphas",6,0,1,2338,0,0,0,0,0,"{beta uncertain buv alphas}"},
    {"buv_betas",6,0,2,2340,0,0,0,0,0,"{beta uncertain buv betas}"},
    {"buv_descriptors",7,0,6,2348,0,0,0,0,0,"{beta uncertain buv descriptors}"},
    {"buv_lower_bounds",6,0,3,2342,0,0,0,0,0,"{beta uncertain buv lower bounds}"},
    {"buv_upper_bounds",6,0,4,2344,0,0,0,0,0,"{beta uncertain buv upper bounds}"},
    {"descriptors",15,0,6,2349,0,0,0,0,0,"{Descriptors} VarCommands.html#VarCAUV_Gamma"},
    {"initial_point",14,0,5,2347,0,0,0,0,0,"{beta uncertain initial point}"},
    {"lower_bounds",14,0,3,2343,0,0,0,0,0,"{Distribution lower bounds} VarCommands.html#VarCAUV_Beta"},
    {"upper_bounds",14,0,4,2345,0,0,0,0,0,"{Distribution upper bounds} VarCommands.html#VarCAUV_Beta"},
}

GuiKeyWord kw_339[5] [static]
Initial value:

= {
    {"descriptors",15,0,4,2421,0,0,0,0,0,"{Descriptors} VarCommands.html#VarDAUV_Negative_Binomial"},
    {"initial_point",13,0,3,2419,0,0,0,0,0,"{binomial uncertain initial point}"},
    {"num_trials",13,0,2,2417,0,0,0,0,0,"{binomial uncertain num trials} VarCommands.html#VarDAUV_Binomial"},
    {"probability_per_trial",14,0,1,1,2415,0,0,0,0,0,"{probability per trial} VarCommands.html#VarDAUV_Beta"},
}

GuiKeyWord kw_340[12] [static]
Initial value:

= {
    {"cdv_descriptors",7,0,6,5,2208,0,0,0,0,0,0,0,0,"{continuous design} VarCommands.html#VarCDV"},
    {"cdv_initial_point",6,0,1,0,2198,0,0,0,0,0,0,0,0,"{continuous design}"},
    {"cdv_lower_bounds",6,0,2,0,2200,0,0,0,0,0,0,0,0,"{continuous design}"},
    {"cdv_scale_types",0x807,0,4,0,2204,0,0,0,0,0,0,0,0,"{continuous design}"},
    {"cdv_scales",0x4806,0,5,0,2206,0,0,0,0,0,0,0,0,"{continuous design}"},
    {"cdv_upper_bounds",6,0,3,0,2202,0,0,0,0,0,0,0,0,"{continuous design}"},
    {"descriptors",15,0,6,0,2209,0,0,0,0,0,"{Descriptors} VarCommands.html#VarCDV"},
    {"initial_point",14,0,1,0,2199,0,0,0,0,0,"{initial point} VarCommands.html#VarCDV"},
}
CHAPTER 12. NAMESPACE DOCUMENTATION

continuous\_design",
    {"lower\_bounds",14,0,2,0,2201,0,0.,0.,0.,0.,0.,0.,0.,0.},"{lower bounds} VarCommands.html#VarCDV",0,*
continuous\_design",
    {"scale\_types",0x80f,0,4,0,2205,0,0.,0.,0.,0.,0.,0.,0.},"{Scaling types} VarCommands.html#VarCDV",0,*
continuous\_design",
    {"scales",0x80e,0,5,0,2207,0,0.,0.,0.,0.,0.,0.,0.,0.},"{Scales} VarCommands.html#VarCDV",0,*
continuous\_design",
    {"upper\_bounds",14,0,3,0,2203,0,0.,0.,0.,0.,0.,0.,0.,0.},"{Upper bounds} VarCommands.html#VarCDV",0,*
continuous\_design"
}

GuiKeyWord kw\_341[10] [static]

Initial value:

```
= {
    {"descriptors",15,0,6,0,2505,0,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.},"{Descriptors} VarCommands.html#VarDUSRV",0,*
    {"interval\_uncertain"},
    {"interval\_probabilities"},
    {"interval\_probs"},
    {"iuv\_descriptors"},
    {"iuv\_interval\_probs"},
    {"iuv\_num\_intervals"},
    {"lower\_bounds"},
    {"num\_intervals"},
    {"upper\_bounds"},
}
```

GuiKeyWord kw\_342[8] [static]

Initial value:

```
= {
    {"csv\_descriptors",7,0,4,0,2570,0,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.},"{Descriptors} VarCommands.html#VarCSV",0,*
    {"csv\_initial\_state"},
    {"csv\_lower\_bounds"},
    {"csv\_upper\_bounds"},
    {"descriptors",15,0,4,0,2571,0,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.},"{Descriptors} VarCommands.html#VarCSV",0,*
    {"initial\_point"},
    {"lower\_bounds"},
    {"num\_intervals"},
    {"upper\_bounds"},
}
```

GuiKeyWord kw\_343[8] [static]

Initial value:

```
= {
    {"ddv\_descriptors",7,0,4,0,2218,0,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.},"{Descriptors} VarCommands.html#VarDDRIV",0,*
    {"ddv\_initial\_point"},
    {"ddv\_lower\_bounds"},
    {"ddv\_upper\_bounds"},
    {"descriptors",15,0,4,0,2219,0,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.},"{Descriptors} VarCommands.html#VarDDRIV",0,*
    {"initial\_point"},
    {"lower\_bounds"},
    {"num\_intervals"},
    {"upper\_bounds"},
```
GuiKeyWord kw_344[1] [static]

Initial value:

```cpp
= {
    {"adjacency_matrix",13,0,1,0,2231}
}
```

GuiKeyWord kw_345[7] [static]

Initial value:

```cpp
= {
    {"categorical",15,1,3,0,2229,kw_344,0,0,0,0,0,0,0,0,0,"integer"},
    {"descriptors",15,0,5,0,2235,0,0,0,0,0,0,0,0,0,0,"{Descriptors} VarCommands.html#VarDDSIV",0,"integer"},
    {"elements",13,0,2,1,2227},
    {"elements_per_variable",0x80d,0,1,0,2225,0,0,0,0,0,0,0,0,0,"integer"},
    {"initial_point",13,0,4,0,2233,0,0,0,0,0,0,0,0,0,"{Initial point} VarCommands.html#VarDDSIV",0,"integer"},
    {"num_set_values",0x805,0,1,0,2224,0,0,0,0,0,0,0,0,0,"integer"},
    {"set_values",5,0,2,1,2226}
}
```

GuiKeyWord kw_346[1] [static]

Initial value:

```cpp
= {
    {"adjacency_matrix",13,0,1,0,2257}
}
```

GuiKeyWord kw_347[7] [static]

Initial value:

```cpp
= {
    {"categorical",15,1,3,0,2255,kw_346,0,0,0,0,0,0,0,0,0,"integer"},
    {"descriptors",15,0,5,0,2261,0,0,0,0,0,0,0,0,0,0,"{Descriptors} VarCommands.html#VarCAUVNormal",0,"real"},
    {"elements",14,0,2,1,2253},
    {"elements_per_variable",0x80d,0,1,0,2251,0,0,0,0,0,0,0,0,0,"real"},
    {"initial_point",14,0,4,0,2259,0,0,0,0,0,0,0,0,0,"real"},
    {"num_set_values",0x805,0,1,0,2258,0,0,0,0,0,0,0,0,0,"real"},
    {"set_values",6,0,2,1,2252}
}
```
GuiKeyWord kw_348[7] [static]
Initial value:

```json
{"adjacency_matrix",13,0,3,0,2243},
{"descriptors",15,0,5,0,2247,0,0,0,0,0,"[Descriptors] VarCommands.html#VarDDSRV",0,"string"},
{"elements",15,0,2,1,2241},
{"elements_per_variable",0x80d,0,1,0,2239,0,0,0,0,0,"string"},
{"initial_point",15,0,4,0,2245,0,0,0,0,0,"[initial point] VarCommands.html#VarDDSRV",0,"string"},
{"num_set_values",0x805,0,1,0,2238,0,0,0,0,0,0,0,"string"},
{"set_values",7,0,2,1,2240}
```

GuiKeyWord kw_349[3] [static]
Initial value:

```json
{"integer",0x19,7,1,0,2223,kw_345,"discrete_interval_uncertain"},
{"real",0x19,7,3,0,2249,kw_347,"discrete_interval_uncertain"},
{"string",0x19,7,2,0,2237,kw_348,"discrete_interval_uncertain"}
```

GuiKeyWord kw_350[9] [static]
Initial value:

```json
{"descriptors",15,0,6,0,2519,0,0,0,0,0,"[Descriptors] VarCommands.html#VarCSV",0,"discrete_interval_uncertain"},
{"intervalprobabilities",14,0,2,0,2511,0,0,0,0,0,"[interval probabilities] VarCommands.html#VarDIUV"},
{"intervalprobs",6,0,2,0,2510},
{"lower_bounds",13,0,3,1,2513,0,0,0,0,0,0,"[Lower bounds] VarCommands.html#VarDIUV"},
{"num_intervals",13,0,1,0,2509,0,0,0,0,0,0,"[Number of intervals defined for each interval variable] VarCommands.html#VarDIUV"},
{"range_probabilities",6,0,2,0,2510},
{"range_probs",6,0,2,0,2510},
{"upper_bounds",13,0,4,2,2515,0,0,0,0,0,0,"[Upper bounds] VarCommands.html#VarDIUV"}
```

GuiKeyWord kw_351[8] [static]
Initial value:

```json
{"descriptors",15,0,4,0,2581,0,0,0,0,0,0,"[descriptors] VarCommands.html#VarDSRIV",0,"discrete_state_range"},
{"dsv_descriptors",7,0,4,0,2580,0,0,0,0,0,0,"[dsv_descriptors] VarCommands.html#VarDSRIV",0,"discrete_state_range"},
{"dsv_initial_state",5,0,1,0,2574,0,0,0,0,0,0,"[dsv_initial_state] VarCommands.html#VarDSRIV",0,"discrete_state_range"},
{"dsv_lower_bounds",5,0,2,0,2576,0,0,0,0,0,0,"[dsv_lower_bounds] VarCommands.html#VarDSRIV",0,"discrete_state_range"},
{"dsv_upper_bounds",5,0,3,0,2578,0,0,0,0,0,0,"[dsv_upper_bounds] VarCommands.html#VarDSRIV",0,"discrete_state_range"},
{"initial_state",13,0,1,0,2575,0,0,0,0,0,0,"[initial states] VarCommands.html#VarDSRIV",0,"discrete_state_range"},
{"lower_bounds",13,0,2,0,2577,0,0,0,0,0,0,"[Lower bounds] VarCommands.html#VarDSRIV",0,"discrete_state_range"},
{"upper_bounds",13,0,3,0,2579,0,0,0,0,0,0,"[Upper bounds] VarCommands.html#VarDSRIV",0,"discrete_state_range"}
```
### GuiKeyWord kw.352[7] [static]

Initial value:

```json
= {
    {"categorical",15,0,3,0,2591,0,0,0,0,0,0,0,"integer"},
    {"descriptors",15,0,5,0,2595,0,0,0,0,0,0,0,"integer"},
    {"elements",13,0,2,1,2589},
    {"elements_per_variable",0x80d,0,1,0,2587,0,0,0,0,0,0,0,"integer"},
    {"initial_state",13,0,4,0,2593,0,0,0,0,0,0,0,"integer"},
    {"num_set_values",0x805,0,1,0,2586,0,0,0,0,0,0,0,"integer"},
    {"set_values",5,0,2,1,2588}
}
```

### GuiKeyWord kw.353[7] [static]

Initial value:

```json
= {
    {"categorical",15,0,3,0,2613,0,0,0,0,0,0,0,"integer"},
    {"descriptors",15,0,5,0,2617,0,0,0,0,0,0,0,"real"},
    {"elements",14,0,2,1,2611},
    {"elements_per_variable",0x80d,0,1,0,2609,0,0,0,0,0,0,0,"real"},
    {"initial_state",14,0,4,0,2615,0,0,0,0,0,0,0,"real"},
    {"num_set_values",0x805,0,1,0,2608,0,0,0,0,0,0,0,"real"},
    {"set_values",6,0,2,1,2615}
}
```

### GuiKeyWord kw.354[6] [static]

Initial value:

```json
= {
    {"descriptors",15,0,4,0,2605,0,0,0,0,0,0,0,"string"},
    {"elements",15,0,2,1,2601},
    {"elements_per_variable",0x80d,0,1,0,2599,0,0,0,0,0,0,0,"string"},
    {"initial_state",15,0,3,0,2603,0,0,0,0,0,0,0,"string"},
    {"num_set_values",0x805,0,1,0,2598,0,0,0,0,0,0,0,"string"},
    {"set_values",7,0,2,1,2600}
}
```

### GuiKeyWord kw.355[3] [static]

Initial value:

```json
= {
    {"integer",0x19,7,1,0,2585,kw.352},
    {"real",0x19,7,3,0,2607,kw.353},
    {"string",0x19,6,2,0,2597,kw.354}
}
```

### GuiKeyWord kw.356[9] [static]

Initial value:
CHAPTER 12. NAMESPACE DOCUMENTATION

= {
    { "categorical",15,0,4,0,2531,0,0,0,0,0,0,0,0,"integer"},
    { "descriptors",15,0,6,0,2533,0,0,0,0,0,0,0,0,"{descriptors} VarCommands.html#VarDSRIV",0,"integer"},
    { "elements",13,0,2,1,2527},
    { "elements_per_variable",13,0,1,0,2525,0,0,0,0,0,0,0,0,"integer"},
    { "initial_point",13,0,5,0,2533,0,0,0,0,0,0,0,0,"integer"},
    { "num_set_values",5,0,1,0,2524,0,0,0,0,0,0,0,0,"integer"},
    { "set_probabilities",14,0,3,0,2529,0,0,0,0,0,0,0,0,"{Probabilities for each set member} VarCommands.html#VarDUSIV"},
    { "set_probs",6,0,3,0,2528},
    { "set_values",5,0,2,1,2526}
}

GuiKeyWord kw_357[9] [static]
Initial value:

= {
    { "categorical",15,0,4,0,2557,0,0,0,0,0,0,0,0,"real"},
    { "descriptors",15,0,6,0,2561,0,0,0,0,0,0,0,0,"{descriptors} VarCommands.html#VarDSSRV",0,"real"},
    { "elements",14,0,2,1,2553},
    { "elements_per_variable",13,0,1,0,2551,0,0,0,0,0,0,0,0,"real"},
    { "initial_point",14,0,5,0,2559,0,0,0,0,0,0,0,0,"real"},
    { "num_set_values",5,0,1,0,2550,0,0,0,0,0,0,0,0,"real"},
    { "set_probabilities",14,0,3,0,2555},
    { "set_probs",6,0,3,0,2554},
    { "set_values",6,0,2,1,2552}
}

GuiKeyWord kw_358[8] [static]
Initial value:

= {
    { "descriptors",15,0,5,0,2547,0,0,0,0,0,0,0,0,"string"},
    { "elements",15,0,2,1,2541},
    { "elements_per_variable",13,0,1,0,2539,0,0,0,0,0,0,0,0,"string"},
    { "initial_point",15,0,4,0,2545,0,0,0,0,0,0,0,0,"string"},
    { "num_set_values",5,0,1,0,2538,0,0,0,0,0,0,0,0,"string"},
    { "set_probabilities",14,0,3,0,2543,0,0,0,0,0,0,0,0,"{Probabilities for each set member} VarCommands.html#VarDUSRV"},
    { "set_probs",6,0,3,0,2542},
    { "set_values",7,0,2,1,2540}
}

GuiKeyWord kw_359[3] [static]
Initial value:

= {
    { "integer",0x19,9,1,0,2523,kw_356},
    { "real",0x19,9,3,0,2549,kw_357},
    { "string",0x19,9,2,0,2537,kw_358}
}
GuiKeyWord kw_360[5]  [static]
Initial value:

```cpp
= {
    {"betas",14,0,1,1,2331,0,0,0,0,0,0,"exponential uncertain betas"},
    {"descriptors",15,0,3,0,2335,0,0,0,0,0,0,"{Descriptors}\ VarCommands.html#VarCAUV_Beta",0,*
        exponential_uncertain"},
    {"euv_betas",7,0,3,0,2334,0,0,0,0,0,0,"exponential uncertain"},
    {"initial_point",14,0,2,0,2333,0,0,0,0,0,0,"exponential uncertain"}
}
```

GuiKeyWord kw_361[7]  [static]
Initial value:

```cpp
= {
    {"alphas",14,0,1,1,2373,0,0,0,0,0,0,"frechet uncertain alphas"},
    {"betas",14,0,2,2,2375,0,0,0,0,0,0,"{frechet uncertain betas}\ VarCommands.html#VarCAUV_Frechet",0,*
        frechet_uncertain"},
    {"fuv_alphas",6,0,1,1,2372,0,0,0,0,0,0,"frechet uncertain"},
    {"fuv_betas",6,0,2,2,2374,0,0,0,0,0,0,"frechet uncertain"},
    {"fuv_descriptors",7,0,4,0,2378,0,0,0,0,0,0,"frechet uncertain"},
    {"initial_point",14,0,3,0,2371,0,0,0,0,0,0,"frechet uncertain"}
}
```

GuiKeyWord kw_362[7]  [static]
Initial value:

```cpp
= {
    {"alphas",14,0,1,1,2353,0,0,0,0,0,0,"gamma uncertain alphas"},
    {"betas",14,0,2,2,2355,0,0,0,0,0,0,"{gamma uncertain betas}\ VarCommands.html#VarCAUV_Gamma",0,*
        gamma_uncertain"},
    {"gauv_alphas",6,0,1,1,2352,0,0,0,0,0,0,"gamma uncertain"},
    {"gauv_betas",6,0,2,2,2354,0,0,0,0,0,0,"gamma uncertain"},
    {"gauv_descriptors",7,0,4,0,2358,0,0,0,0,0,0,"gamma uncertain"},
    {"initial_point",14,0,3,0,2357,0,0,0,0,0,0,"gamma uncertain"}
}
```

GuiKeyWord kw_363[4]  [static]
Initial value:

```cpp
= {
    {"descriptors",15,0,3,0,2439,0,0,0,0,0,0,"{Descriptors}\ VarCommands.html#VarDAUV_Hypergeometric",0,"geometric uncertain"},
    {"initial_point",13,0,2,2,2437,0,0,0,0,0,0,"geometric uncertain"},
    {"prob_per_trial",6,0,1,1,2434,0,0,0,0,0,0,"geometric uncertain"},
    {"probability_per_trial",14,0,1,1,2435,0,0,0,0,0,0,"geometric uncertain"}
}
```
GuiKeyWord kw_364[7] [static]
Initial value:
= {
    {"alphas",14,0,1,1,2363,0,0,0,0,0,0,"gumbel uncertain alphas"
        VarCommands.html#VarCAUV_Gumbel",0,"gumbel uncertain"},
    {"betas",14,0,2,2,2365,0,0,0,0,0,"gumbel uncertain betas"
        VarCommands.html#VarCAUV_Gumbel",0,"gumbel uncertain"},
    {"descriptors",15,0,4,0,2369,0,0,0,0,0,"Descriptors"
        VarCommands.html#VarCAUV_Frechet"},
    {"guuv alphas",6,0,1,1,2362,0,0,0,0,0,"gumbel uncertain"},
    {"guuv betas",6,0,2,2,2364,0,0,0,0,0,"gumbel uncertain"},
    {"initial point",14,0,3,0,2367,0,0,0,0,0,0,"gumbel uncertain"}

GuiKeyWord kw_365[11] [static]
Initial value:
= {
    {"abscissas",14,0,2,1,2395,0,0,0,0,0,"sets of abscissas for bin-based histogram"
        VarCommands.html#VarCAUV_Bin_histogram"},
    {"counts",14,0,3,2,2399,0,0,0,0,0,"sets of counts for bin-based histogram variables"
        VarCommands.html#VarCAUV_Bin_histogram"},
    {"descriptors",15,0,5,0,2403,0,0,0,0,0,"Descriptors"
        VarCommands.html#VarCEUVInterval"},
    {"huv bin abscissas",6,0,2,1,2394},
    {"huv bin counts",6,0,3,2,2398},
    {"huv bin descriptors",7,0,5,0,2402,0,0,0,0,0,0,"histogram_bin_uncertain"},
    {"initial point",14,0,4,0,2401,0,0,0,0,0,0,"histogram_bin_uncertain"},
    {"num pairs",5,0,1,0,2392,0,0,0,0,0,0,"histogram_bin_uncertain"},
    {"ordinates",14,0,2,1,2397,0,0,0,0,0,0,"sets of ordinates for bin-based histogram"
        VarCommands.html#VarCAUV_Bin_histogram"},
    {"pairs per variable",13,0,1,0,2393,0,0,0,0,0,0,"histogram_bin_uncertain"}

GuiKeyWord kw_366[6] [static]
Initial value:
= {
    {"abscissas",13,0,2,1,2459,0,0,0,0,0,"sets of abscissas for point-based histogram"
        VarCommands.html#VarDAUV_Point_histogram"},
    {"counts",14,0,3,2,2461,0,0,0,0,0,"sets of counts for point-based histogram variables"
        VarCommands.html#VarDAUV_Point_histogram"},
    {"descriptors",15,0,5,0,2465,0,0,0,0,0,"integer"
        VarCommands.html#VarCEUVInterval"},
    {"initial point",13,0,4,0,2463,0,0,0,0,0,0,"integer"},
    {"num pairs",5,0,1,0,2456,0,0,0,0,0,0,"integer"},
    {"pairs per variable",13,0,1,0,2457,0,0,0,0,0,0,"integer"}
}

GuiKeyWord kw_367[6] [static]
Initial value:
= {
    {"abscissas",14,0,2,1,2483},
    {"counts",14,0,3,2,2485},
    {"descriptors",15,0,5,0,2489,0,0,0,0,0,"real"},
    {"initial point",14,0,4,0,2487,0,0,0,0,0,0,"real"},
    {"num pairs",5,0,1,0,2480,0,0,0,0,0,0,"real"},
    {"pairs per variable",13,0,1,0,2481,0,0,0,0,0,0,"real"}
GuiKeyWord kw_368[6]  [static]
Initial value:
= {
    {*abscessas*,15,0,2,1,2471},
    {*counts*,14,0,3,2,2473},
    {*descriptors*,15,0,5,0,2477,0,0,0,0,0,"{Descriptors} VarCommands.html#VarDIUV",0,*
string*},
    {*initial_point*,15,0,4,0,2475,0,0,0,0,0,0,"string"},
    {*num_pairs*,5,0,1,0,2469,0,0,0,0,0,0,"string"},
    {*pairs_per_variable*,13,0,1,0,2469,0,0,0,0,0,0,"string"}
}

GuiKeyWord kw_369[3]  [static]
Initial value:
= {
    {*integer*,0x19,6,1,0,2455,kw_366},
    {*real*,0x19,6,3,0,2479,kw_367},
    {*string*,0x19,6,2,0,2467,kw_368}
}

GuiKeyWord kw_370[5]  [static]
Initial value:
= {
    {*descriptors*,15,0,5,0,2451,0,0,0,0,0,"{Descriptors} VarCommands.html#VarDAUV,Point Histogram",0,"hypergeometric_uncertain"},
    {*initial_point*,13,0,4,0,2449,0,0,0,0,0,0,"hypergeometric_uncertain"},
    {*num_drawn*,13,0,3,3,2447,0,0,0,0,0,0,"hypergeometric uncertain num drawn" VarCommands.html#VarDAUV,Hypergeometric",0,"hypergeometric uncertain"},
    {*selected_population*,13,0,2,2,2445,0,0,0,0,0,0,"hypergeometric uncertain selected population" VarCommands.html#VarDAUV,Hypergeometric",0,"hypergeometric uncertain"},
    {*total_population*,13,0,1,1,2443,0,0,0,0,0,0,"hypergeometric uncertain total population" VarCommands.html#VarDAUV,Hypergeometric",0,"hypergeometric uncertain"}
}

GuiKeyWord kw_371[2]  [static]
Initial value:
= {
    {*lnnu_zetas*,6,0,1,1,2289,0,0,0,0,0,0,"lognormal_uncertain"},
    {*zetas*,14,0,1,1,2281,0,0,0,0,0,0,"lognormal uncertain zetas" VarCommands.html#VarCAUV,Lognormal",0,"lognormal uncertain"}
}

GuiKeyWord kw_372[4]  [static]
Initial value:
= {
    {*error_factors*,14,0,1,1,2287,0,0,0,0,0,0,"{CHOOS variance spec.}{lognormal uncertain error factors} VarCommands.html#VarCAUV,Lognormal",0,"lognormal uncertain"},
    {*lnnu_std_deviations*,6,0,1,1,2286,0,0,0,0,0,0,"lognormal uncertain"},
    {*std_deviations*,14,0,1,1,2285,0,0,0,0,0,0,"{lognormal uncertain standard deviations} VarCommands.html#VarCAUV,Lognormal",0,"lognormal uncertain"}
}
GuiKeyWord kw_373[11] [static]

Initial value:

```c++
= {
    "descriptors", 15, 0, 5, 0, 2295, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Normal uncertain",
    "initial_point", 14, 0, 4, 0, 2293, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Normal uncertain",
    "lower_bounds", 14, 0, 2, 0, 2298, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Normal uncertain",
    "means", 14, 0, 3, 0, 2290, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Normal uncertain",
    "upper_bounds", 14, 0, 2, 0, 2294, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Normal uncertain",
    "upper_bounds", 14, 0, 2, 2, 2311, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Normal uncertain"
}
```

GuiKeyWord kw_374[7] [static]

Initial value:

```c++
= {
    "descriptors", 15, 0, 4, 0, 2315, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Triangular uncertain",
    "initial_point", 14, 0, 3, 0, 2313, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Triangular uncertain",
    "lower_bounds", 14, 0, 1, 1, 2309, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Triangular uncertain",
    "luuv_descriptors", 7, 0, 4, 0, 2314, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Triangular uncertain",
    "luuv_lower_bounds", 6, 0, 1, 1, 2308, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Triangular uncertain",
    "luuv_upper_bounds", 6, 0, 2, 2, 2310, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Triangular uncertain",
    "probability_per_trial", 14, 0, 1, 1, 2425, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Triangular uncertain"
}
```

GuiKeyWord kw_375[5] [static]

Initial value:

```c++
= {
    "descriptors", 15, 0, 4, 0, 2431, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Geometric uncertain",
    "num_trials", 13, 0, 3, 0, 2429, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Geometric uncertain",
    "probability_per_trial", 14, 0, 1, 1, 2424, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Geometric uncertain",
    "probability_per_trial", 14, 0, 1, 1, 2425, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Geometric uncertain"
}
```

GuiKeyWord kw_376[11] [static]

Initial value:

```c++
= {
    "descriptors", 15, 0, 6, 0, 2275, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Lognormal uncertain",
    "lower_bounds", 14, 0, 3, 0, 2269, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Lognormal uncertain",
    "upper_bounds", 14, 0, 2, 2, 2311, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "Lognormal uncertain"
}
```
12.1. Dakota Namespace Reference

```json
{"means":14.0,1,1,2265,0,0,0,0,0,"{normal uncertain means}
VarCommands.html#VarCAUV_Normal",0,"normal uncertain"},
{"nuv_lower_bounds":6,0,3,0,2268,0,0,0,0,0,0,"normal uncertain"},
{"nuv_means":6,0,1,1,2264,0,0,0,0,0,0,"normal uncertain"},
{"nuv_stddeviations":6,0,2,2,2266,0,0,0,0,0,0,"normal uncertain"},
{"nuv_upper_bounds":6,0,4,0,2270,0,0,0,0,0,0,"normal uncertain"},
{"stddeviations":14.0,2,2,2267,0,0,0,0,0,"{normal uncertain standard deviations}
VarCommands.html#VarCAUV_Normal",0,"normal uncertain"},
{"upper_bounds":14.0,4,0,2271,0,0,0,0,0,"{Distribution upper bounds}
VarCommands.html#VarCAUV_Normal",0,"normal uncertain"}
}

GuiKeyWord kw.377[3] [static]
Initial value:

=`
{"descriptors":15.0,3,0,2411,0,0,0,0,0,"{Descriptors} VarCommands.html#VarDAUV_Binomial* ,0,"poisson uncertain"},
{"initial_point":13.0,2,0,2409,0,0,0,0,0,0,"poisson uncertain"},
{"lambdas":14.0,1,1,2407,0,0,0,0,0,0,"poisson uncertain lambdas
VarCommands.html#VarDAUV_Poisson",0,"poisson uncertain"}
```

GuiKeyWord kw.378[9] [static]
Initial value:

=`
{"descriptors":15.0,5,0,2327,0,0,0,0,0,"{Descriptors} VarCommands.html#VarDAUV_Exponential* ,0,"triangular uncertain"},
{"lower_bounds":14.0,2,2,2321,0,0,0,0,0,"{Distribution lower bounds
VarCommands.html#VarDAUV_Triangular",0,"triangular uncertain"},
{"modes":14.0,1,1,2319,0,0,0,0,0,"{triangular uncertain modes
VarCommands.html#VarDAUV_Triangular",0,"triangular uncertain"},
{"tuv_descriptors":7,0,5,0,2326,0,0,0,0,0,0,"triangular uncertain modes
VarCommands.html#VarDAUV_Triangular",0,"triangular uncertain"},
{"tuv_lower_bounds":6,0,2,2,2320,0,0,0,0,0,0,"triangular uncertain"},
{"tuv_modes":6,0,1,1,2318,0,0,0,0,0,0,"triangular uncertain"},
{"tuv_upper_bounds":6,0,3,3,2322,0,0,0,0,0,0,"triangular uncertain"},
{"upper_bounds":14.0,3,3,2323,0,0,0,0,0,"{Distribution upper bounds
VarCommands.html#VarDAUV_Triangular",0,"triangular uncertain"}
```

GuiKeyWord kw.379[7] [static]
Initial value:

=`
{"descriptors":15.0,4,0,2305,0,0,0,0,0,"{Descriptors} VarCommands.html#VarDAUV_Loguniform* ,0,"uniform uncertain"},
{"initial_point":14.0,3,0,2303,0,0,0,0,0,0,"uniform uncertain"},
{"lower_bounds":14.0,1,1,2299,0,0,0,0,0,"{Distribution lower bounds
VarCommands.html#VarDAUV_Uniform",0,"uniform uncertain"},
{"upper_bounds":14.0,2,2,2301,0,0,0,0,0,"{Distribution upper bounds
VarCommands.html#VarDAUV_Uniform",0,"uniform uncertain"},
{"uuv_descriptors":7,0,4,0,2304,0,0,0,0,0,0,"uniform uncertain"},
{"uuv_lower Bounds":6,0,1,1,2298,0,0,0,0,0,0,"uniform uncertain"},
{"uuv_upper Bounds":6,0,2,2,2300,0,0,0,0,0,0,"uniform uncertain"}
```
GuiKeyWord kw_380[7] [static]
Initial value:

```{"alphas",14,0,1,1,2383,0,0,0,0,0,0,0,0,0,"weibull uncertain alphas"}
VarCommands.html#VarCAUV
{"betas",14,0,2,2,2385,0,0,0,0,0,0,0,0,0,"weibull uncertain betas"}
VarCommands.html#VarCAUV
{"descriptors",15,0,4,0,2389,0,0,0,0,0,0,0,0,0,"weibull uncertain"}
VarCommands.html#VarCAUV
{"initial_point",14,0,3,0,2387,0,0,0,0,0,0,0,0,0,"weibull uncertain"},
{"wuv_alphas",6,0,1,1,2382,0,0,0,0,0,0,0,0,0,"weibull uncertain"},
{"wuv betas",6,0,2,2,2384,0,0,0,0,0,0,0,0,0,"weibull uncertain"},
{"wuv descriptors",7,0,4,0,2388,0,0,0,0,0,0,0,0,0,"weibull uncertain"}
```

GuiKeyWord kw_382[6] [static]
Initial value:

```{"environment",0x108,15,1,1,kw_12,0,0,0,0,0,"Environment"}
EnvCommands.html","{"interface",0x308,9,5,5,2619,kw_26,0,0,0,0,"Interface"}
InterfCommands.html","{"method",0x308,90,2,2,83,kw_275,0,0,0,0,"Method"}
MethodCommands.html","{"model",8,7,3,1,1907,kw_310,0,0,0,0,"Model"}
ModelCommands.html","{"responses",0x308,19,6,6,2735,kw_336,0,0,0,0,"Responses"}
RespCommands.html","{"variables",0x308,34,4,4,2175,kw_381,0,0,0,0,"Variables"}
VarCommands.html"}
```

KeyWord kw_1[3] [static]
Initial value:

```{"eval_id",8,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"eval id"}
{"header",8,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"header"}
{"interface_id",8,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"interface id"}
```

1480 distinct keywords (plus 205 aliases)

KeyWord kw_2[3] [static]
Initial value:

```{"annotated",8,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"annotated"}
{"custom_annotated",8,3,1,0,kw_1,0,0,0,0,0,0,0,0,0,0,0,0,"custom annotated"}
{"freeform",8,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"freeform"}
```

1480 distinct keywords (plus 205 aliases)
KeyWord kw.3[2] [static]
Initial value:

= {
    "input", 11, 3, 1, 0, kw.2, 0, 0, 0, 0, N{str, postRunInput},
    "output", 11, 0, 2, 0, 0, 0, 0, 0, N{str, postRunOutput}
}

KeyWord kw.4[3] [static]
Initial value:

= {
    "eval_id", 8, 0, 2, 0, 0, 0, 0, 0, N{str, preRunOutputFormat_TABULAR_EVAL_ID},
    "header", 8, 0, 1, 0, 0, 0, 0, 0, N{str, preRunOutputFormat_TABULAR_HEADER},
    "interface_id", 8, 0, 3, 0, 0, 0, 0, 0, N{str, preRunOutputFormat_TABULAR_IFACE_ID}
}

KeyWord kw.5[3] [static]
Initial value:

= {
    "annotated", 8, 0, 1, 0, 0, 0, 0, 0, N{str, preRunOutputFormat_TABULAR_ANNOTATED},
    "custom.annotated", 8, 3, 1, 0, kw.4, 0, 0, 0, N{str, preRunOutputFormat_TABULAR_NONE},
    "freeform", 8, 0, 1, 0, 0, 0, 0, 0, N{str, preRunOutputFormat_TABULAR_NONE}
}

KeyWord kw.6[2] [static]
Initial value:

= {
    "input", 11, 0, 1, 0, 0, 0, 0, 0, N{str, preRunInput},
    "output", 11, 3, 2, 0, kw.5, 0, 0, 0, N{str, preRunOutput}
}

KeyWord kw.7[1] [static]
Initial value:

= {
    "stop restart", 0x29, 0, 1, 0, 0, 0, 0, 0, N{int, stopRestart}
}

KeyWord kw.8[1] [static]
Initial value:

= {
    "results_output_file", 11, 0, 1, 0, 0, 0, 0, 0, N{str, resultsOutputFile}
}
CHAPTER 12. NAMESPACE DOCUMENTATION

KeyWord kw_9[2]  [static]
Initial value:
= {
  "input",11,0,1,0,0,0..0,N_stm(str,runInput)},
  "output",11,0,2,0,0,0..0,N_stm(str,runOutput)}

KeyWord kw_10[3]  [static]
Initial value:
= {
  "eval_id",8,0,2,0,0,0..0,N_stm(augment_utype,tabularFormat_TABULAR_EVAL_ID)},
  "header",8,0,1,0,0,0..0,N_stm(augment_utype,tabularFormat_TABULAR_HEADER)},
  "interface_id",8,0,3,0,0,0..0,N_stm(augment_utype,tabularFormat_TABULAR_IFACE_ID)}

KeyWord kw_11[5]  [static]
Initial value:
= {
  "annotated",8,0,2,0,0,0..0,N_stm(utype,tabularFormat_TABULAR_ANNOTATED)},
  "custom_annotated",8,3,2,0,kw_10,0,0,0..0,N_stm(utype,tabularFormat_TABULAR_NONE)},
  "freeform",8,0,2,0,0,0..0,N_stm(utype,tabularFormat_TABULAR_NONE)},
  "tabular_data_file",11,0,1,0,0,0..0,N_stm(str,tabularDataFile)},
  "tabular_graphics_file",3,0,1,0,0,0..0,-1,N_stm(str,tabularDataFile)}

KeyWord kw_12[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,errorMessage)},
  "graphics",8,0,0,0,0,0..0,N_stm(true,graphicsFlag)},
  "post_run",8,0,8,0,kw_10,0,0,0,N_stm(true,postRunFlag)},
  "pre_run",8,2,6,0,kw_10,0,0,0,N_stm(true,preRunFlag)},
  "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,ouputPrecision)}

KeyWord kw_13[1]  [static]
Initial value:
= {
  "cache_tolerance",10,0,1,0,0,0..0,0,N_ilm(Real,nearbyEvalCacheTol)}
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_14[4] [static]
Initial value:

```plaintext
= {
    "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, N_ifm(false, restartFileFlag),
    "strict_cache_equality", 8, 1, 3, 0, kw_13, 0, 0, 0, N_ifm(true, nearbyEvalCacheFlag)
}
```

KeyWord kw_15[1] [static]
Initial value:

```plaintext
= {
    "processors_per_analysis", 0x19, 0, 1, 0, 0, 0, 0, N_ifm(pint, procsPerAnalysis)
}
```

KeyWord kw_16[4] [static]
Initial value:

```plaintext
= {
    "abort", 8, 0, 1, 0, 0, 0, 0, N_ifm(lit, failAction_abort),
    "continuation", 8, 0, 1, 0, 0, 0, 0, N_ifm(lit, failAction_continuation),
    "recover", 14, 0, 1, 0, 0, 0, 0, N_ifm(Rlit, TYPE_DATA, failAction_recover),
    "retry", 9, 0, 1, 0, 0, 0, 0, N_ifm(lit, TYPE_DATA, failAction_retry)
}
```

KeyWord kw_17[1] [static]
Initial value:

```plaintext
= {
    "numpy", 8, 0, 1, 0, 0, 0, 0, N_ifm(true, numpyFlag)
}
```

KeyWord kw_18[8] [static]
Initial value:

```plaintext
= {
    "copy_files", 15, 0, 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", 8, 0, 2, 0, 0, 0, 0, 0, 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, N_ifm(strL, linkFiles),
    "named", 11, 0, 1, 0, 0, 0, 0, N_ifm(str, workDir),
    "replace", 8, 0, 6, 0, 0, 0, 0, 0, N_ifm(true, templateReplace)
}
```
KeyWord kw_19[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, 0, 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, 0, kw_18, 0, 0, 0, 0, N_ifm(true, useWorkdir)}
}

KeyWord kw_20[12]  [static]
Initial value:
= {
  {"analysis_components", 15, 0, 1, 0, 0, 0, 0, 0, N_ifm(str2D, analysisComponents)},
  {"deactivate", 8, 4, 6, 0, kw_14},
  {"direct", 8, 1, 4, 1, kw_15, 0, 0, 0, 0, N_ifm(type, interfaceType, TEST_INTERFACE)},
  {"failure_capture", 8, 4, 5, 0, kw_16},
  {"fork", 8, 9, 4, 1, kw_19, 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_17, 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_19, 0, 0, 0, N_ifm(type, interfaceType, SYSTEM_INTERFACE)}
}

KeyWord kw_21[2]  [static]
Initial value:
= {
  {"master", 8, 0, 1, 1, 0, 0, 0, 0, 0, N_ifm(type, analysisScheduling, MASTER_SCHEDULING)},
  {"peer", 8, 0, 1, 1, 0, 0, 0, 0, 0, N_ifm(type, analysisScheduling, PEER_SCHEDULING)}
}

KeyWord kw_22[2]  [static]
Initial value:
= {
  {"dynamic", 8, 0, 1, 1, 0, 0, 0, 0, 0, N_ifm(type, asynchLocalEvalScheduling, DYNAMIC_SCHEDULING)},
  {"static", 8, 0, 1, 1, 0, 0, 0, 0, 0, N_ifm(type, asynchLocalEvalScheduling, STATIC_SCHEDULING)}
}

KeyWord kw_23[3]  [static]
Initial value:
= {
  {"analysis_concurrency", 0x19, 0, 3, 0, 0, 0, 0, 0, 0, N_ifm(pint, asynchLocalAnalysisConcurrency)},
  {"evaluation_concurrency", 0x19, 0, 1, 0, 0, 0, 0, 0, 0, N_ifm(pint, asynchLocalEvalConcurrency)},
  {"local_evaluation_scheduling", 8, 2, 2, 0, kw_22}
}
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_24[2] [static]

Initial value:
= {
  {"dynamic", 8, 0, 1, 1, 0, 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_25[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_24}
}

KeyWord kw_26[9] [static]

Initial value:
= {
  {"algebraic_mappings", 11, 0, 0, 0, 0, 0, 0, 0, N_ifm(str, algebraicMappings)},
  {"analysis_drivers", 15, 0, 3, 0, kw_20, 0, 0, 0, N_ifm(strL, analysisDrivers)},
  {"analysis_scheduling", 8, 2, 1, 0, kw_21},
  {"analysis_servers", 0x19, 0, 0, 0, 0, 0, 0, 0, N_ifm(pint, analysisServers)},
  {"asynchronous", 8, 3, 0, kw_23, 0, 0, 0, 0, N_ifm(type, interfaceSynchronization, ASYNCHRONOUS_INTERFACE)},
  {"evaluation_scheduling", 8, 2, 0, 0, 0, kw_25},
  {"evaluation_servers", 0x19, 0, 0, 0, 0, 0, 0, 0, N_ifm(pint, evalServers)},
  {"id_interface", 11, 0, 0, 0, 0, 0, 0, 0, N_ifm(str, idInterface)},
  {"processors_per_evaluation", 0x19, 0, 0, 0, 0, 0, 0, 0, N_ifm(pint, procsPerEval)}
}

KeyWord kw_27[1] [static]

Initial value:
= {
  {"model_pointer", 11, 0, 0, 0, 0, 0, 0, 0, N_mdm(str, modelPointer)}
}

KeyWord kw_28[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_29[2] [static]

Initial value:
= {
  {"complementary", 8, 0, 1, 1, 0, 0, 0, 0, 0, N_mdm(type, distributionType, COMPLEMENTARY)},
  {"cumulative", 8, 0, 1, 1, 0, 0, 0, 0, 0, N_mdm(type, distributionType, CUMULATIVE)}
}
KeyWord kw_30 [static]
Initial value:
= {
    "num_gen_reliability_levels", 13, 0, 1, 0, 0, 0, 0, 0, N_mdm(num_resplevs, genReliabilityLevels)
}

KeyWord kw_31 [static]
Initial value:
= {
    "num_probability_levels", 13, 0, 1, 0, 0, 0, 0, 0, N_mdm(num_resplevs, probabilityLevels)
}

KeyWord kw_32 [static]
Initial value:
= {
    "mt19937", 8, 0, 1, 0, 0, 0, 0, N_mdm(lit, rngName_mt19937),
    "rnum2", 8, 0, 1, 0, 0, 0, 0, N_mdm(lit, rngName_rnum2)
}

KeyWord kw_33 [static]
Initial value:
= {
    "distribution", 8, 2, 1, 0, kw_29,
    "gen_reliability_levels", 14, 1, 3, 0, kw_30, 0, 0, 0, N_mdm(resplevs, genReliabilityLevels),
    "probability_levels", 14, 1, 2, 0, kw_31, 0, 0, 0, N_mdm(resplevs01, probabilityLevels),
    "rng", 8, 2, 4, 0, kw_32
}

KeyWord kw_34 [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_35 [static]
Initial value:
= {
    "eval_id", 8, 0, 2, 0, 0, 0, 0, 0, N_mdm(augment_utype, approxExportFormat_TABULAR_EVAL_ID),
    "header", 8, 0, 1, 0, 0, 0, 0, 0, N_mdm(augment_utype, approxExportFormat_TABULAR_HEADER),
    "interface_id", 8, 0, 3, 0, 0, 0, 0, 0, N_mdm(augment_utype, approxExportFormat_TABULAR_IFACE_ID)
}
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_36[3]  [static]
Initial value:
= {
  {"annotated",8,0,1,0,0,0,0,0,N,mxm(utype,approxExportFormat_TABULAR_ANNOTATED)},
  {"custom_annotated",8,3,1,0,kw_35,0,0,0,N,mxm(utype,approxExportFormat_TABULAR_ANNOTATED)},
  {"freeform",8,0,1,0,0,0,0,0,N,mxm(utype,approxExportFormat_TABULAR_NONE)}
}

KeyWord kw_37[3]  [static]
Initial value:
= {
  {"distance",8,0,1,0,0,0,0,0,N,mxm(lit,fitnessMetricType_distance)},
  {"gradient",8,0,1,0,0,0,0,0,N,mxm(lit,fitnessMetricType_gradient)},
  {"predicted_variance",8,0,1,0,0,0,0,0,N,mxm(lit,fitnessMetricType_predicted_variance)}
}

KeyWord kw_38[3]  [static]
Initial value:
= {
  {"eval_id",8,0,1,0,0,0,0,0,N,mxm(augment_type,approxImportFormat_TABULAR_EVAL_ID)},
  {"header",8,0,1,0,0,0,0,0,N,mxm(augment_type,approxImportFormat_TABULAR_HEADER)},
  {"interface_id",8,0,1,0,0,0,0,0,N,mxm(augment_type,approxImportFormat_TABULAR_IFACE_ID)}
}

KeyWord kw_39[4]  [static]
Initial value:
= {
  {"active_only",8,0,2,0,0,0,0,0,N,mxm(true,approxImportActive)},
  {"annotated",8,0,1,0,0,0,0,0,N,mxm(utype,approxImportFormat_TABULAR_ANNOTATED)},
  {"custom_annotated",8,3,1,0,kw_38,0,0,0,N,mxm(utype,approxImportFormat_TABULAR_NONE)},
  {"freeform",8,0,1,0,0,0,0,0,N,mxm(utype,approxImportFormat_TABULAR_NONE)}
}

KeyWord kw_40[2]  [static]
Initial value:
= {
  {"parallel",8,0,1,1,0,0,0,0,N,mxm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
  {"series",8,0,1,1,0,0,0,0,N,mxm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
}

KeyWord kw_41[3]  [static]
Initial value:
= {
  {"gen_reliabilities",8,0,1,1,0,0,0,0,N,mxm(type,responseLevelTarget_GEN_RELIABILITIES)},
  {"probabilities",8,0,1,1,0,0,0,0,N,mxm(type,responseLevelTarget_PROBABILITIES)},
  {"system",8,0,2,0,0,0,0,N,mxm(type,responseLevelTarget_SYSTEM)}
}
CHAPTER 12. NAMESPACE DOCUMENTATION

KeyWord kw_42[2] [static]

Initial value:
= {
    "compute", 8, 3, 2, 0, kw_41,
    "num_response_levels", 13, 0, 1, 0, 0, 0, 0, N_mdm(num_resplevs, responseLevels)
}

KeyWord kw_43[11] [static]

Initial value:
= {
    0, 0, 1, 0, 0, kw_27,
    0, 0, 2, 0, 0, kw_28,
    0, 0, 4, 0, 0, kw_33,
    "batch_selection", 8, 4, 3, 0, kw_34,
    "batch_size", 9, 0, 0, 0, 0, 0, N_mdm(int, batchSize),
    "emulator_samples", 9, 0, 1, 0, 0, 0, 0, N_mdm(int, emulatorSamples),
    "export_points_file", 11, 3, 6, 0, kw_36,
    "fitness_metric", 8, 3, 2, 0, kw_37,
    "import_points_file", 11, 4, 5, 0, kw_39,
    "misc_options", 15, 0, 8, 0, 0, 0, 0, N_mdm(str, miscOptions),
    "response_levels", 14, 2, 7, 0, kw_42,
}

KeyWord kw_44[9] [static]

Initial value:
= {
    "linear_equality_constraint_matrix", 14, 0, 6, 0, 0, 0, 0, N_mdm(RealDL, linearEqConstraintCoeffs),
    "linear_equality_scale_types", 15, 0, 8, 0, 0, 0, 0, N_mdm(str, linearEqScaleTypes),
    "linear_equality_scales", 14, 0, 9, 0, 0, 0, 0, N_mdm(RealDL, linearEqScales),
    "linear_equality_targets", 14, 0, 7, 0, 0, 0, 0, N_mdm(RealDL, linearEqTargets),
    "linear_inequality_constraint_matrix", 14, 0, 1, 0, 0, 0, 0, N_mdm(RealDL, linearIneqConstraintCoeffs),
    "linear_inequality_lower_bounds", 14, 0, 2, 0, 0, 0, 0, N_mdm(RealDL, linearIneqLowerBnds),
    "linear_inequality_scale_types", 15, 0, 4, 0, 0, 0, 0, N_mdm(str, linearIneqScaleTypes),
    "linear_inequality_scales", 14, 0, 5, 0, 0, 0, 0, N_mdm(RealDL, linearIneqScales),
    "linear_inequality_upper_bounds", 14, 0, 3, 0, 0, 0, 0, N_mdm(RealDL, linearIneqUpperBnds)
}

KeyWord kw_45[7] [static]

Initial value:
= {
    "merit1", 8, 0, 1, 1, 0, 0, 0, N_mdm(lit, meritFunction_merit1),
    "merit1_smooth", 8, 0, 1, 1, 0, 0, 0, N_mdm(lit, meritFunction_merit1_smooth),
    "merit2", 8, 0, 1, 1, 0, 0, 0, N_mdm(lit, meritFunction_merit2),
    "merit2_smooth", 8, 0, 1, 1, 0, 0, 0, N_mdm(lit, meritFunction_merit2_smooth),
    "merit2_squared", 8, 0, 1, 1, 0, 0, 0, N_mdm(lit, meritFunction_merit2_squared),
    "merit_max", 8, 0, 1, 1, 0, 0, 0, N_mdm(lit, meritFunction_merit_max),
    "merit_max_smooth", 8, 0, 1, 1, 0, 0, 0, N_mdm(lit, meritFunction_merit_max_smooth)
}
KeyWord `kw_46[2]`  [static]
Initial value:

```csharp
= {
    {"blocking",8,0,1,1,0,0.,0.,0,Ν_mdmm(lit,evalSynchronize,blocking)},
    {"nonblocking",8,0,1,1,0,0.,0.,0,Ν_mdmm(lit,evalSynchronize,nonblocking)}
}
```

KeyWord `kw_47[11]`  [static]
Initial value:

```csharp
= {
    {0,0,1,0,0,kw_27},
    {0,0,9,0,kw_44},
    {"constraint_penalty",10,0,7,0,0,0.,0.,0,Ν_mdmm(Real,constrPenalty)},
    {"contraction_factor",10,0,2,0,0,0.,0.,0,Ν_mdmm(Real,contractStepLength)},
    {"initial_delta",10,0,1,0,0,0.,0.,0,Ν_mdmm(Real,initStepLength)},
    {"merit_function",8,1,6,0,kw_45},
    {"smoothing_factor",10,0,8,0,0,0.,0.,0,Ν_mdmm(Real,smoothFactor)},
    {"solution_accuracy",2,0,4,0,0,0.,0.,0,1,Ν_mdmm(Real,solnTarget)},
    {"solution_target",10,0,4,0,0,0,0,0,0,Ν_mdmm(Real,solnTarget)},
    {"threshold_delta",10,0,3,0,0,0.,0.,0,Ν_mdmm(Real,threshStepLength)}
}
```

KeyWord `kw_48[3]`  [static]
Initial value:

```csharp
= {
    {"eval_id",8,0,2,0,0,0,0,0,Ν_mdmm(augment_utype,approxExportFormat_TABULAR_EVAL_ID)},
    {"header",8,0,1,0,0,0,0,0,Ν_mdmm(augment_utype,approxExportFormat_TABULAR_HEADER)},
    {"interface_id",8,0,3,0,0,0,0,0,Ν_mdmm(augment_utype,approxExportFormat_TABULAR_IFACE_ID)}
}
```

KeyWord `kw_49[3]`  [static]
Initial value:

```csharp
= {
    {"annotated",8,0,1,0,0,0,0,0,Ν_mdmm(utype,approxExportFormat_TABULAR_ANNOTATED)},
    {"custom_annotated",8,3,1,0,kw_48,0,0,0,Ν_mdmm(utype,approxExportFormat_TABULAR_NONE)},
    {"freeform",8,0,1,0,0,0,0,0,Ν_mdmm(utype,approxExportFormat_TABULAR_NONE)}
}
```

KeyWord `kw_50[3]`  [static]
Initial value:

```csharp
= {
    {"eval_id",8,0,2,0,0,0,0,0,Ν_mdmm(augment_utype,approxImportFormat_TABULAR_EVAL_ID)},
    {"header",8,0,1,0,0,0,0,0,Ν_mdmm(augment_utype,approxImportFormat_TABULAR_HEADER)},
    {"interface_id",8,0,3,0,0,0,0,0,Ν_mdmm(augment_utype,approxImportFormat_TABULAR_IFACE_ID)}
}
```
KeyWord kw_51[4] [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(utype, approxImportFormat_TABULAR_ANNOTATED)},
  {"freeform", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(utype, approxImportFormat_TABULAR_NONE)}
}

KeyWord kw_52[6] [static]
Initial value:
= {
  {"dakota", 8, 0, 1, 0, 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, 3, 5, 0, kw_50, 0, 0, 0, N, mdm(str, approxExportFile)},
  {"import_points_file", 11, 4, 4, 0, kw_51, 0, 0, 0, N, mdm(str, approxImportFile)},
  {"posterior_adaptive", 8, 0, 3, 0, 0, 0, 0, 0, N, mdm(true, adaptPosteriorRefine)},
  {"surfpack", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(type, emulatorType_KRIGING_EMULATOR)}
}

KeyWord kw_53[2] [static]
Initial value:
= {
  {"collocation_ratio", 10, 0, 1, 1, 0, 0, 0, 0, N, mdm(Realp, collocationRatio)},
  {"posterior_adaptive", 8, 0, 2, 0, 0, 0, 0, 0, N, mdm(true, adaptPosteriorRefine)}
}

KeyWord kw_54[2] [static]
Initial value:
= {
  {"expansion_order", 13, 2, 1, 1, kw_53, 0, 0, 0, N, mdm(usharray, expansionOrder)},
  {"sparse_grid_level", 13, 0, 1, 1, 0, 0, 0, 0, N, mdm(usharray, sparseGridLevel)}
}

KeyWord kw_55[1] [static]
Initial value:
= {
  {"sparse_grid_level", 13, 0, 1, 1, 0, 0, 0, 0, N, mdm(usharray, sparseGridLevel)}
}

KeyWord kw_56[5] [static]
Initial value:
= {
  {"gaussian_process", 8, 6, 1, 1, kw_52},
  {"kriging", 6, 0, 1, 1, kw_52, 0, 0, 0, N, mdm(type, emulatorType_PCE_EMULATOR)},
  {"pce", 8, 2, 1, 1, kw_54, 0, 0, 0, N, mdm(type, emulatorType_PCE_EMULATOR)},
  {"use_derivatives", 8, 0, 2, 0, 0, 0, 0, 0, N, mdm(true, methodUseDerivsFlag)}
}
KeyWord kw_57[6] [static]
Initial value:
= {
  "chains", 0x29, 0, 1, 0, 0, 3, 0, 0, N, Nmdm(int, numChains),
  "crossover_chain_pairs", 0x29, 0, 3, 0, 0, 0, 0, N, Nmdm(int, crossoverChainPairs),
  "emulator", 8, 5, 6, 0, kw_56,
  "gr_threshold", 0x1a, 0, 4, 0, 0, 0, 0, N, Nmdm(Real, grThreshold),
  "jump_step", 0x29, 0, 5, 0, 0, 0, 0, N, Nmdm(int, jumpStep),
  "num_cr", 0x29, 0, 2, 0, 1, 0, 0, N, Nmdm(int, numCR)
}

KeyWord kw_58[1] [static]
Initial value:
= {
  "proposal_updates", 9, 0, 1, 0, 0, 0, 0, N, Nmdm(int, proposalCovUpdates),
}

KeyWord kw_59[2] [static]
Initial value:
= {
  "diagonal", 8, 0, 1, 1, 0, 0, 0, N, Nmdm(lit, proposalCovInputType_diagonal),
  "matrix", 8, 0, 1, 1, 0, 0, 0, N, Nmdm(lit, proposalCovInputType_matrix)
}

KeyWord kw_60[2] [static]
Initial value:
= {
  "diagonal", 8, 0, 1, 1, 0, 0, 0, N, Nmdm(lit, proposalCovInputType_diagonal),
  "matrix", 8, 0, 1, 1, 0, 0, 0, N, Nmdm(lit, proposalCovInputType_matrix)
}

KeyWord kw_61[4] [static]
Initial value:
= {
  "derivatives", 8, 1, 1, 1, kw_58, 0, 0, 0, N, Nmdm(lit, proposalCovType_derivatives),
  "filename", 11, 2, 1, 1, kw_59, 0, 0, 0, N, Nmdm(str, proposalCovFile),
  "prior", 8, 0, 1, 1, 0, 0, 0, N, Nmdm(lit, proposalCovType_prior),
  "values", 14, 2, 1, 1, kw_60, 0, 0, 0, N, Nmdm(RealDL, proposalCovData)
}

KeyWord kw_62[2] [static]
Initial value:
= {
  "mt19937", 8, 0, 1, 1, 0, 0, 0, 0, N, Nmdm(lit, rngName_mt19937),
  "rnum2", 8, 0, 1, 1, 0, 0, 0, 0, N, Nmdm(lit, rngName_rnum2)
}
KeyWord kw.63[7] [static]
Initial value:
= {
    "adaptive_metropolis",8,0,1,0,0,0,0,0,N,mdm(lit,mcmcType_adaptive_metropolis),
    "delayed_rejection",8,0,1,0,0,0,0,0,N,mdm(lit,mcmcType_delayed_rejection),
    "metropolis_hastings",8,0,1,0,0,0,0,0,N,mdm(lit,mcmcType_metropolis_hastings),
    "multilevel",8,0,1,0,0,0,0,0,N,mdm(lit,mcmcType_multilevel),
    "proposal_covariance",8,3,0,kw.61,0,0,0,0,N,mdm(lit,proposalCovType_user),
    "rng",8,2,2,0,kw.62
}

KeyWord kw.64[3] [static]
Initial value:
= {
    "eval_id",8,0,2,0,0,0,0,0,0,N,mdm(augment_utype,approxExportFormat_TABULAR_EVAL_ID),
    "header",8,0,1,0,0,0,0,0,0,N,mdm(augment_utype,approxExportFormat_TABULAR_HEADER),
    "interface_id",8,0,3,0,0,0,0,0,0,N,mdm(augment_utype,approxExportFormat_TABULAR_IFACE_ID)
}

KeyWord kw.65[3] [static]
Initial value:
= {
    "annotated",8,0,1,0,0,0,0,0,0,N,mdm(utype,approxExportFormat_TABULAR_ANNOTATED),
    "freeform",8,0,1,0,0,0,0,0,0,N,mdm(utype,approxExportFormat_TABULAR_NONE)
}

KeyWord kw.66[3] [static]
Initial value:
= {
    "eval_id",8,0,2,0,0,0,0,0,0,N,mdm(augment_utype,approxImportFormat_TABULAR_EVAL_ID),
    "header",8,0,1,0,0,0,0,0,0,N,mdm(augment_utype,approxImportFormat_TABULAR_HEADER),
    "interface_id",8,0,3,0,0,0,0,0,0,N,mdm(augment_utype,approxImportFormat_TABULAR_IFACE_ID)
}

KeyWord kw.67[4] [static]
Initial value:
= {
    "active_only",8,0,2,0,0,0,0,0,0,N,mdm(true,approxImportActive),
    "annotated",8,0,1,0,0,0,0,0,0,N,mdm(utype,approxImportFormat_TABULAR_ANNOTATED),
    "custom_annotated",8,3,1,0,kw.61,0,0,0,0,N,mdm(utype,approxImportFormat_TABULAR_NONE),
    "freeform",8,0,1,0,0,0,0,0,0,N,mdm(utype,approxImportFormat_TABULAR_NONE)
}
KeyWord kw_68[4] [static]
Initial value:
= {
  
  "emulator_samples", 0,0,1,1,0,0,0,0,0,0,N_mdmb(int, emulatorSamples),
  "export_points_file", 11,3,3,0,kw,65,0,0,0,0,N_mdmb(str, approxExportFile),
  "import_points_file", 11,4,2,0,kw,67,0,0,0,0,N_mdmb(str, approxImportFile)
}

KeyWord kw_69[3] [static]
Initial value:
= {
  
  "eval_id", 8,0,2,0,0,0,0,0,0,N_mdmb(augment_utype, approxExportFormat_TABULAR_EVAL_ID),
  "header", 8,0,1,0,0,0,0,0,0,N_mdmb(augment_utype, approxExportFormat_TABULAR_HEADER),
  "interface_id", 8,0,3,0,0,0,0,0,0,N_mdmb(augment_utype, approxExportFormat_TABULAR_IFACE_ID)
}

KeyWord kw_70[3] [static]
Initial value:
= {
  
  "annotated", 8,0,1,0,0,0,0,0,0,N_mdmb(utype, approxExportFormat_TABULAR_ANNOTATED),
  "custom_annotated", 8,3,1,0,kw,69,0,0,0,0,N_mdmb(utype, approxExportFormat_TABULAR_NONE),
  "freeform", 8,0,1,0,0,0,0,0,0,N_mdmb(utype, approxExportFormat_TABULAR_NONE)
}

KeyWord kw_71[3] [static]
Initial value:
= {
  
  "eval_id", 8,0,2,0,0,0,0,0,0,N_mdmb(augment_utype, approxImportFormat_TABULAR_EVAL_ID),
  "header", 8,0,1,0,0,0,0,0,0,N_mdmb(augment_utype, approxImportFormat_TABULAR_HEADER),
  "interface_id", 8,0,3,0,0,0,0,0,0,N_mdmb(augment_utype, approxImportFormat_TABULAR_IFACE_ID)
}

KeyWord kw_72[4] [static]
Initial value:
= {
  
  "active_only", 8,0,2,0,0,0,0,0,0,N_mdmb(true, approxImportActive),
  "annotated", 8,0,1,0,0,0,0,0,0,N_mdmb(utype, approxImportFormat_TABULAR_ANNOTATED),
  "custom_annotated", 8,3,1,0,kw,71,0,0,0,0,N_mdmb(utype, approxImportFormat_TABULAR_NONE),
  "freeform", 8,0,1,0,0,0,0,0,0,N_mdmb(utype, approxImportFormat_TABULAR_NONE)
}

KeyWord kw_73[6] [static]
Initial value:
= {
  
  "dakota", 8,0,1,1,0,0,0,0,0,N_mdmb(type, emulatorType_GP_EMULATOR),
  "emulator_samples", 9,0,2,0,0,0,0,0,0,N_mdmb(int, emulatorSamples),
  "export_points_file", 11,3,5,0,kw,70,0,0,0,0,N_mdmb(str, approxExportFile),
  "import_points_file", 11,4,4,0,kw,72,0,0,0,0,N_mdmb(str, approxImportFile),
  "posterior_adaptive", 8,0,3,0,0,0,0,0,0,N_mdmb(true, adaptPosteriorRefine),
  "surfpack", 8,0,1,1,0,0,0,0,0,N_mdmb(type, emulatorType_KRIGING_EMULATOR)
}
KeyWord kw_74[2]  [static]
Initial value:
= {
    {"collocation\_ratio",10,1,1,0,0,0,0,0,N,mmd(Realp, collocationRatio)},
    {"posterior\_adaptive",8,0,2,0,0,0,0,0,N,mmd(true, adaptPosteriorRefine)}
}

KeyWord kw_75[2]  [static]
Initial value:
= {
    {"expansion\_order",13,2,1,1,kw_74,0,0,0,0,N,mmd(usharray, expansionOrder)},
    {"sparse\_grid\_level",13,0,1,1,0,0,0,0,N,mmd(usharray, sparseGridLevel)}
}

KeyWord kw_76[1]  [static]
Initial value:
= {
    {"sparse\_grid\_level",13,0,1,1,0,0,0,0,N,mmd(usharray, sparseGridLevel)}
}

KeyWord kw_77[5]  [static]
Initial value:
= {
    {"gaussian\_process",8,6,1,1,kw_73},
    {"kriging",0,6,1,1,kw_73,0,0,-1},
    {"pce",8,2,1,1,kw_75,0,0,0,N,mmd(type, emulatorType_PCE\_EMULATOR)},
    {"sc",8,1,1,kw_76,0,0,0,N,mmd(type, emulatorType_SC\_EMULATOR)},
    {"use\_derivatives",8,0,2,0,0,0,0,0,N,mmd(true, methodUseDerivsFlag)}
}

KeyWord kw_78[3]  [static]
Initial value:
= {
    {0,0,7,0,0,kw_63},
    {"emulator",8,5,1,0,kw_77},
    {"logit\_transform",8,0,2,0,0,0,0,0,N,mmd(true, logitTransform)}
}

KeyWord kw_79[8]  [static]
Initial value:
= {
    {0,0,1,0,0,kw_27},
    {0,0,2,0,0,kw_28},
    {"calibrate\_sigma",8,0,4,0,0,0,0,0,N,mmd(true, calibrateSigmaFlag)},
    {"dream",8,6,1,1,kw_57,0,0,0,N,mmd(utype, subMethod\_SUBMETHOD\_DREAM)},
    {"ogensa",8,3,1,1,kw_68,0,0,0,N,mmd(utype, subMethod\_SUBMETHOD\_GENSEA)},
    {"likelihood\_scale",10,0,3,0,0,0,0,0,N,mmd(Real, likelihoodScale)},
    {"queso",8,2,1,1,kw_78,0,0,0,N,mmd(utype, subMethod\_SUBMETHOD\_QUESO)},
    {"standardized\_space",8,0,2,0,0,0,0,0,N,mmd(true, standardizedSpace)}
}
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_80[4]  [static]
Initial value:
= {
  {0,0,1,0,0,kw,27},
  {0,0,2,2,0,0,0,0,N_mdm(ivec,stepsPerVariable)},
  "step_vector",14,0,1,0,0,0,0,N_mdm(Real,stepVector)},
  "steps_per_variable",13,0,2,2,0,0,0,0,N_mdm(ivec,stepsPerVariable)}

KeyWord kw_81[5]  [static]
Initial value:
= {
  {"misc_options",15,0,4,0,0,0,0,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_82[3]  [static]
Initial value:
= {
  {0,0,1,0,0,kw,27},
  {0,0,5,0,0,kw,81},
  "{beta_solver_name",11,0,1,0,0,0,0,0,N_mdm(str,betaSolverName)}
}

KeyWord kw_83[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_84[4]  [static]
Initial value:
= {
  {0,0,1,0,0,kw,27},
  {0,0,5,0,0,kw,81},
  {0,0,2,0,0,kw,83},
  "{"}
}

KeyWord kw_85[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_86[8]  [static]
Initial value:

= {
  {0,0,1,0,kw_27},
  {0,0,5,0,kw_81},
  {"constraint_penalty",10,0,6,0,0,0,0,0,N,mdm(Real,constraintPenalty)},
  {"division",8,2,1,kw_85},
  {"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_87[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_88[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_89[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_90[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_91[5] [static]
Initial value:

= {
    "non_adaptive", 8, 0, 2, 0, 0, 0, 0, 0, Nmdm(false, mutationAdaptive),
    "offset_cauchy", 8, 2, 1, 1, kw_90, 0, 0, 0, 0, Nmdm(lit, mutationType_offset_cauchy),
    "offset_normal", 8, 2, 1, 1, kw_90, 0, 0, 0, 0, Nmdm(lit, mutationType_offset_normal),
    "replace_uniform", 8, 0, 1, 0, 0, 0, 0, 0, Nmdm(lit, mutationType_replace_uniform)
}

KeyWord kw_92[4] [static]
Initial value:

= {
    "chc", 9, 0, 1, 1, 0, 0, 0, 0, Nmdm(ilit2, TYPE_DATA_replacementType_chc),
    "elitist", 9, 0, 1, 1, 0, 0, 0, 0, Nmdm(ilit2, TYPE_DATA_replacementType_elitist),
    "new_solutions_generated", 9, 0, 2, 0, 0, 0, 0, 0, Nmdm(int, newSolsGenerated),
    "random", 9, 0, 1, 1, 0, 0, 0, 0, Nmdm(ilit2, TYPE_DATA_replacementType_random)
}

KeyWord kw_93[11] [static]
Initial value:

= {
    0, 0, 1, 0, 0, kw_27,
    0, 0, 5, 0, 0, kw_81,
    "constraint_penalty", 10, 0, 9, 0, 0, 0, 0, 0, Nmdm(Real, constraintPenalty),
    "crossover_rate", 10, 0, 5, 0, 0, 0, 0, 0, Nmdm(Real, crossoverRate),
    "crossover_type", 8, 3, 2, 0, kw_87,
    "fitness_type", 8, 2, 3, 0, kw_88,
    "initialization_type", 8, 3, 2, 0, kw_89,
    "mutation_rate", 10, 0, 7, 0, 0, 0, 0, 0, Nmdm(Real, mutationRate),
    "mutation_type", 8, 5, 8, 0, kw_81,
    "population_size", 0x19, 0, 1, 0, 0, 0, 0, 0, Nmdm(pint, populationSize),
    "replacement_type", 8, 6, 4, 0, kw_92
}

KeyWord kw_94[2] [static]
Initial value:

= {
    "constraint_penalty", 10, 0, 2, 0, 0, 0, 0, 0, Nmdm(Real, constraintPenalty),
    "contraction_factor", 10, 0, 1, 0, 0, 0, 0, 0, Nmdm(Real, contractFactor)
}

KeyWord kw_95[3] [static]
Initial value:

= {
    "adaptive_pattern", 8, 0, 1, 1, 0, 0, 0, 0, Nmdm(lit, exploratoryMoves_adaptive),
    "basic_pattern", 8, 0, 1, 1, 0, 0, 0, 0, Nmdm(lit, exploratoryMoves_simple),
    "multi_step", 8, 0, 1, 1, 0, 0, 0, 0, Nmdm(lit, exploratoryMoves_multi_step)
}
KeyWord kw_96[2] [static]
Initial value:
= {
    {"coordinate", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(lit, patternBasis_coordinate)},
    {"simplex", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(lit, patternBasis_simplex)}
}

KeyWord kw_97[2] [static]
Initial value:
= {
    {"blocking", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(lit, evalSynchronize_blocking)},
    {"nonblocking", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(lit, evalSynchronize_nonblocking)}
}

KeyWord kw_98[12] [static]
Initial value:
= {
    {0, 0, 1, 0, 0, kw_27},
    {0, 0, 5, 0, 0, kw_81},
    {0, 0, 2, 0, 0, kw_83},
    {0, 0, 2, 0, 0, kw_96},
    {"constant_penalty", 8, 0, 1, 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_95},
    {"no_expansion", 8, 0, 2, 0, 0, 0, 0, 0, N, mdm(false, expansionFlag)},
    {"pattern_basli", 8, 2, 4, 0, kw_36},
    {"stochastic", 8, 0, 5, 0, 0, 0, 0, 0, N, mdm(true, randomizeOrderFlag)},
    {"synchronization", 8, 2, 8, 0, kw_37},
    {"total_pattern_size", 9, 0, 6, 0, 0, 4, 0, 0, N, mdm(int, totalPatternSize)}
}

KeyWord kw_99[8] [static]
Initial value:
= {
    {0, 0, 1, 0, 0, kw_27},
    {0, 0, 5, 0, 0, kw_81},
    {0, 0, 2, 0, 0, kw_83},
    {0, 0, 2, 0, 0, kw_96},
    {"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_100[4] [static]
Initial value:
= {
    {0, 0, 1, 0, 0, kw_27},
    {0, 0, 9, 0, 0, kw_24},
    {"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_101[3] [static]
Initial value:
= {
    {0,0,1,0,kw,27},
    {0,0,9,0,kw,44},
}

KeyWord kw_102[1] [static]
Initial value:
= {
    "drop_tolerance",10,0,1,0,0,0.,0.,0,N_mdm(Real,vbdDropTolerance) }

KeyWord kw_103[14] [static]
Initial value:
= {
    {0,0,1,0,0,kw,27},
    {0,0,2,0,0,kw,28},
    {"box_behnken",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_BOX_BEHNKEN)},
    {"central_composite",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_CENTRAL_COMPOSITE)},
    {"fixed_seed",8,0,5,0,0,0,0,0,0,N_mdm(true,fixedSeedFlag)},
    {"grid",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_GRID)},
    {"lhs",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_LHS)},
    {"main_effects",8,0,2,0,0,0,0,0,0,N_mdm(true,mainEffectFlag)},
    {"oa_lhs",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_OA_LHS)},
    {"oas",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_OAS)},
    {"quality_metrics",8,0,3,0,0,0,0,0,0,N_mdm(true,volQualityFlag)},
    {"random",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_RANDOM)},
    {"symbols",9,0,6,0,0,0,0,0,0,N_mdm(int,numSymbols)},
    {"variance_based_decomp",8,1,4,0,kw_102,0,0,0,0,N_mdm(true,vbdFlag)}
}

KeyWord kw_104[7] [static]
Initial value:
= {
    {0,0,1,0,0,kw,27},
    {0,0,9,0,0,kw,44},
    {"bfgs",8,0,1,1,0,0,0,0,0,N_mdm(utype,methodName_DOT_BFGS)},
    {"frcg",8,0,1,1,0,0,0,0,0,N_mdm(utype,methodName_DOT_FRCG)},
    {"mmfd",8,0,1,1,0,0,0,0,0,N_mdm(utype,methodName_DOT_MMFD)},
    {"slp",8,0,1,1,0,0,0,0,0,N_mdm(utype,methodName_DOT_SLP)},
    {"sqp",8,0,1,1,0,0,0,0,0,N_mdm(utype,methodName_DOT_SQP)}
}

KeyWord kw_105[3] [static]
Initial value:
= {
    {"eval_id",8,0,2,0,0,0,0,0,0,N_mdm(augment_utype,approxExportFormat_TABULAR_EVAL_ID)},
    {"header",8,0,1,0,0,0,0,0,0,N_mdm(augment_utype,approxExportFormat_TABULAR_HEADER)},
    {"interface_id",8,0,3,0,0,0,0,0,0,N_mdm(augment_utype,approxExportFormat_TABULAR_IFACE_ID)}
}
KeyWord kw_106[3]  [static]
Initial value:

```c
= {
    "annotated", 8, 0, 1, 0, 0, 0, 0, N_mdm(utype, approxExportFormat_TABULAR_ANNOTATED),
    "custom_annotated", 8, 3, 1, 0, kw_105, 0, 0, 0, N_mdm(utype, approxExportFormat_TABULAR_NONE),
    "freeform", 8, 0, 1, 0, 0, 0, 0, 0, N_mdm(utype, approxExportFormat_TABULAR_NONE)
}
```

KeyWord kw_107[2]  [static]
Initial value:

```c
= {
    "dakota", 8, 0, 1, 0, 0, 0, 0, N_mdm(type, emulatorType_GP_EMULATOR),
    "surfpack", 8, 0, 1, 0, 0, 0, 0, N_mdm(type, emulatorType_KRIGING_EMULATOR)
}
```

KeyWord kw_108[3]  [static]
Initial value:

```c
= {
    "eval_id", 8, 0, 2, 0, 0, 0, 0, 0, N_mdm(augment_utype, approxImportFormat_TABULAR_EVAL_ID),
    "header", 8, 0, 1, 0, 0, 0, 0, 0, N_mdm(augment_utype, approxImportFormat_TABULAR_HEADER),
    "interface_id", 8, 0, 3, 0, 0, 0, 0, 0, N_mdm(augment_utype, approxImportFormat_TABULAR_IFACE_ID)
}
```

KeyWord kw_109[4]  [static]
Initial value:

```c
= {
    "active_only", 8, 0, 2, 0, 0, 0, 0, 0, N_mdm(true, approxImportActive),
    "annotated", 8, 0, 1, 0, 0, 0, 0, N_mdm(utype, approxImportFormat_TABULAR_ANNOTATED),
    "custom_annotated", 8, 3, 1, 0, kw_108, 0, 0, 0, N_mdm(utype, approxImportFormat_TABULAR_NONE),
    "freeform", 8, 0, 1, 0, 0, 0, 0, 0, N_mdm(utype, approxImportFormat_TABULAR_NONE)
}
```

KeyWord kw_110[7]  [static]
Initial value:

```c
= {
    0, 0, 1, 0, 0, kw_27,
    "export_points_file", 11, 3, 4, 0, kw_106, 0, 0, 0, N_mdm(str, approxExportFile),
    "gaussian_process", 8, 2, 1, 0, kw_107],
    "import_points_file", 11, 4, 3, 0, kw_109, 0, 0, 0, N_mdm(str, approxImportFile),
    "kriging", 0, 2, 1, 0, kw_107, 0, 0, -2,
    "seed", 0x19, 0, 5, 0, 0, 0, 0, N_mdm(pint, randomSeed),
    "use_derivatives", 8, 0, 2, 0, 0, 0, 0, 0, N_mdm(true, methodUseDerivsFlag)
}
```
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_111[5] [static]
Initial value:

= 

{{0,0,1,0,0,kw_27},
 {0,0,2,0,0,kw_28},
 {0,0,4,0,0,kw_33},
 {*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_112[3] [static]
Initial value:

= 

{{"grid",8,0,1,0,0,0,0,0,N,mdm(lit,trialType.grid)},
 {*halton*,8,0,1,0,0,0,0,0,N,mdm(lit,trialType_halton)},
 {*random*,8,0,1,0,0,0,0,0,N,mdm(lit,trialType_random)}}

KeyWord kw_113[1] [static]
Initial value:

= 

{{"drop_tolerance",10,0,1,0,0,0,0,0,N,mdm(Real,vbdDropTolerance)}}

KeyWord kw_114[8] [static]
Initial value:

= 

{{0,0,1,0,0,kw_27},
 {0,0,2,0,0,kw_28},
 {*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_112},
 {*variance_based_decomp*,8,1,3,0,kw_113,0,0,0,0,N,mdm(true,vbdFlag)}}

KeyWord kw_115[1] [static]
Initial value:

= 

{{"drop_tolerance",10,0,1,0,0,0,0,0,N,mdm(Real,vbdDropTolerance)}}
KeyWord kw_116[11] [static]

Initial value:

```c
= {
    {0,0,1,0,0,kw_27},
    {"fixed_sequence",8,0,6,0,0,0,0,0,Nmdm(true, fixedSequenceFlag)},
    {"halton",8,0,1,0,0,0,0,0,Nmdm(utype, methodName_FSU_HALTON)},
    {"hammersley",8,0,1,0,0,0,0,0,Nmdm(utype, methodName_FSU_HAMMERSLEY)},
    {"latize",8,0,2,0,0,0,0,0,Nmdm(true, latizeFlag)},
    {"prime_base",13,0,9,0,0,0,0,0,Nmdm(ivec, primeBase)},
    {"quality_metrics",8,0,3,0,0,0,0,0,0,Nmdm(true, volQualityFlag)},
    {"samples",9,5,5,0,0,0,0,0,Nmdm(int, numSamples)},
    {"sequence_leap",13,0,8,0,0,0,0,0,0,Nmdm(ivec, sequenceLeap)},
    {"sequence_start",13,0,7,0,0,0,0,0,0,Nmdm(ivec, sequenceStart)},
    {"variance_based_decomp",8,1,4,0,kw_115,0.,0.,0,Nmdm(true, vbdFlag)}
}
```

KeyWord kw_117[3] [static]

Initial value:

```c
= {
    {"eval_id",8,0,2,0,0,0,0,0,0,Nmdm(augment_utype, approxExportFormat_TABULAR_EVAL_ID)},
    {"header",8,0,1,0,0,0,0,0,0,Nmdm(augment_utype, approxExportFormat_TABULAR_HEADER)},
    {"interface_id",8,0,3,0,0,0,0,0,0,Nmdm(augment_utype, approxExportFormat_TABULAR_IFACE_ID)}
}
```

KeyWord kw_118[3] [static]

Initial value:

```c
= {
    {"annotated",8,0,1,0,0,0,0,0,0,Nmdm(utype, approxExportFormat_TABULAR_ANNOTATED)},
    {"custom_annotated",8,3,1,0,kw_117,0.,0.,0,Nmdm(utype, approxExportFormat_TABULAR_NONE)},
    {"freeform",8,0,1,0,0,0,0,0,0,Nmdm(utype, approxExportFormat_TABULAR_NONE)}
}
```

KeyWord kw_119[3] [static]

Initial value:

```c
= {
    {"eval_id",8,0,2,0,0,0,0,0,0,Nmdm(augment_utype, approxImportFormat_TABULAR_EVAL_ID)},
    {"header",8,0,1,0,0,0,0,0,0,Nmdm(augment_utype, approxImportFormat_TABULAR_HEADER)},
    {"interface_id",8,0,3,0,0,0,0,0,0,Nmdm(augment_utype, approxImportFormat_TABULAR_IFACE_ID)}
}
```

KeyWord kw_120[4] [static]

Initial value:

```c
= {
    {"active_only",8,0,2,0,0,0,0,0,0,Nmdm(true, approxImportActive)},
    {"annotated",8,0,1,0,0,0,0,0,0,Nmdm(utype, approxImportFormat_TABULAR_ANNOTATED)},
    {"custom_annotated",8,3,1,0,kw_119,0.,0.,0,Nmdm(utype, approxImportFormat_TABULAR_NONE)},
    {"freeform",8,0,1,0,0,0,0,0,0,Nmdm(utype, approxImportFormat_TABULAR_NONE)}
}
```
KeyWord kw_121[2]  [static]
Initial value:
  = {
    {"parallel",8,0,1,0,0,0,0,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
    {"series",8,0,1,0,0,0,0,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
  }

KeyWord kw_122[3]  [static]
Initial value:
  = {
    {"gen_reliabilities",8,0,1,0,0,0,0,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES)},
    {"probabilities",8,0,1,0,0,0,0,0,N_mdm(type,responseLevelTarget_PROBABILITIES)},
    {"system",8,2,2,0,kw_121}
  }

KeyWord kw_123[2]  [static]
Initial value:
  = {
    {"compute",8,3,2,0,kw_122},
    {"num_response_levels",13,0,1,0,0,0,0,0,N_mdm(num_resplevs,responseLevels)}
  }

KeyWord kw_124[7]  [static]
Initial value:
  = {
    {0,0,1,0,0,kw_27},
    {0,0,2,0,0,kw_28},
    {0,0,4,0,0,kw_33},
    {"emulator_samples",9,0,1,0,0,0,0,0,N_mdm(int,emulatorSamples)},
    {"export_points_file",11,3,3,0,kw_118,0,0,0,N_mdm(str,approxExportFile)},
    {"import_points_file",11,4,2,0,kw_120,0,0,0,N_mdm(str,approxImportFile)},
    {"response_levels",14,2,4,0,kw_123,0,0,0,N_mdm(resplevs,responseLevels)}
  }

KeyWord kw_125[2]  [static]
Initial value:
  = {
    {0,0,1,0,0,kw_27},
    {"seed",0x19,0,1,0,0,0,0,0,N_mdm(pint,randomSeed)}
  }

KeyWord kw_126[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_127[3]  [static]
Initial value:
= {
    "gen_reliabilities",8,0,1,0,0,0,0,0,N_mdms(type, responseLevelTarget_GEN_RELIABILITIES),
    "probabilities",8,0,1,0,0,0,0,0,N_mdms(type, responseLevelTarget_PROBABILITIES),
    "system",8,2,0,kw_126
}

KeyWord kw_128[2]  [static]
Initial value:
= {
    "compute",8,3,0,kw_127,
    "num_response_levels",13,0,1,0,0,0,0,0,N_mdms(num_resplevs, responseLevels)
}

KeyWord kw_129[3]  [static]
Initial value:
= {
    "eval_id",8,0,2,0,0,0,0,0,0,N_mdms(augment_utype, approxExportFormat_TABULAR_EVAL_ID),
    "header",8,0,1,0,0,0,0,0,N_mdms(augment_utype, approxExportFormat_TABULAR_HEADER),
    "interface_id",8,0,3,0,0,0,0,0,N_mdms(augment_utype, approxExportFormat_TABULAR_IFACE_ID)
}

KeyWord kw_130[3]  [static]
Initial value:
= {
    "annotated",8,0,1,0,0,0,0,0,0,N_mdms(utype, approxExportFormat_TABULAR_ANNOTATED),
    "custom_annotated",8,3,1,0,kw_129,0,0,0,N_mdms(utype, approxExportFormat_TABULAR_NONE),
    "freeform",8,0,1,0,0,0,0,0,0,N_mdms(utype, approxExportFormat_TABULAR_NONE)
}

KeyWord kw_131[2]  [static]
Initial value:
= {
    "dakota",8,0,1,1,0,0,0,0,0,N_mdms(type, emulatorType_GP_EMULATOR),
    "surfpack",8,0,1,1,0,0,0,0,0,N_mdms(type, emulatorType_KRIGING_EMULATOR)
}

KeyWord kw_132[3]  [static]
Initial value:
= {
    "eval_id",8,0,2,0,0,0,0,0,0,N_mdms(augment_utype, approxImportFormat_TABULAR_EVAL_ID),
    "header",8,0,1,0,0,0,0,0,N_mdms(augment_utype, approxImportFormat_TABULAR_HEADER),
    "interface_id",8,0,3,0,0,0,0,0,N_mdms(augment_utype, approxImportFormat_TABULAR_IFACE_ID)
}
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_133[4]  [static]
Initial value:

= {
  {"active_only",8,0,2,0,0,0,0,0,N_mdm(true,approxImportActive)},
  {"annotated",8,1,0,0,0,0,0,N_mdm(utype,approxImportFormat_TABULAR_ANNOTATED)},
  {"custom_annotated",8,3,1,0,kw_132,0,0,0,N_mdm(utype,approxImportFormat_TABULAR_NONE)},
  {"freeform",8,0,1,0,0,0,0,0,N_mdm(utype,approxImportFormat_TABULAR_NONE)}
}

KeyWord kw_134[5]  [static]
Initial value:

= {
  {"export_points_file",11,3,4,0,kw_130,0,0,0,N_str(approxExportFile)},
  {"gaussian_process",8,2,1,0,kw_131},
  {"import_points_file",11,4,3,0,kw_133,0,0,0,N_str(approxImportFile)},
  {"kriging",0,2,1,0,kw_131,0,0,0,-2},
  {"use_derivatives",8,0,2,0,0,0,0,0,N_mdm(true,methodUseDerivsFlag)}
}

KeyWord kw_135[8]  [static]
Initial value:

= {
  {0,0,1,0,0,kw_27},
  {0,0,2,0,0,kw_28},
  {0,0,4,0,0,kw_33},
  {"ea",8,0,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_EA)},
  {"ego",8,5,1,0,kw_134,0,0,0,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_128,0,0,0,N_mdm(resplevs,responseLevels)},
  {"sbo",8,5,1,0,kw_134,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_SBO)}
}

KeyWord kw_136[2]  [static]
Initial value:

= {
  {"mt19937",8,0,1,1,0,0,0,0,N_lit(rngName_mt19937)},
  {"rnum2",8,0,1,1,0,0,0,0,N_lit(rngName_rnum2)}
}

KeyWord kw_137[3]  [static]
Initial value:

= {
  {"eval_id",8,0,2,0,0,0,0,0,N_mdm(augment_type,approxExportFormat_TABULAR_EVAL_ID)},
  {"header",8,0,1,0,0,0,0,0,N_mdm(augment_type,approxExportFormat_TABULAR_HEADER)},
  {"interface_id",8,0,1,0,0,0,0,0,N_mdm(augment_type,approxExportFormat_TABULAR_IFACE_ID)}
}
KeyWord kw_138[3]  [static]
Initial value:

= {
    {"annotated", 8, 0, 1, 0, 0, 0, 0, 0, 0, N, "mdm(utype, approxExportFormat_TABULAR_ANNOTATED)",}
    {"custom_annotated", 8, 3, 1, 0, kw_137, 0, 0, 0, 0, N, "mdm(utype, approxExportFormat_TABULAR_NONE)",}
    {"freeform", 8, 0, 1, 0, 0, 0, 0, 0, 0, N, "mdm(utype, approxExportFormat_TABULAR_NONE)"}
}

KeyWord kw_139[2]  [static]
Initial value:

= {
    {"dakota", 8, 0, 1, 1, 0, 0, 0, 0, 0, N, "mdm(type, emulatorType_GP_EMITOR)",}
    {"surfpack", 8, 0, 1, 1, 0, 0, 0, 0, 0, N, "mdm(type, emulatorType_KRIGING_EMITOR)"}
}

KeyWord kw_140[3]  [static]
Initial value:

= {
    {"eval_id", 8, 0, 2, 0, 0, 0, 0, 0, 0, N, "mdm(augment.utype, approxImportFormat_TABULAR_EVAL_ID)",}
    {"header", 8, 0, 1, 0, 0, 0, 0, 0, N, "mdm(augment.utype, approxImportFormat_TABULAR_HEADER)",}
    {"interface.id", 8, 0, 3, 0, 0, 0, 0, 0, N, "mdm(augment.utype, approxImportFormat_TABULAR_IFACE_ID)"}
}

KeyWord kw_141[4]  [static]
Initial value:

= {
    {"active_only", 8, 0, 2, 0, 0, 0, 0, 0, 0, N, "mdm(true, approxImportActive)",}
    {"annotated", 8, 0, 1, 0, 0, 0, 0, 0, N, "mdm(utype, approxImportFormat_TABULAR_ANNOTATED)",}
    {"custom_annotated", 8, 3, 1, 0, kw_140, 0, 0, 0, 0, N, "mdm(utype, approxImportFormat_TABULAR_NONE)",}
    {"freeform", 8, 0, 1, 0, 0, 0, 0, 0, 0, N, "mdm(utype, approxImportFormat_TABULAR_NONE)"}
}

KeyWord kw_142[5]  [static]
Initial value:

= {
    {"export_points_file", 11, 3, 4, 0, kw_138, 0, 0, 0, 0, N, "mdm(str, approxExportFile)",}
    {"gaussian_process", 8, 2, 1, 0, kw_139},
    {"import_points_file", 11, 4, 3, 0, kw_141, 0, 0, 0, 0, N, "mdm(str, approxImportFile)",}
    {"kriging", 0, 2, 1, 0, kw_139, 0, 0, -2},
    {"use_derivatives", 8, 0, 2, 0, 0, 0, 0, 0, 0, N, "mdm(true, methodUseDerivsFlag)"}
}
KeyWord kw.143[7] [static]
Initial value:

= 

{  
0,0,1,0,0,kw.27},
{0,0,2,0,0,kw.28},
{"ea",8,5,1,0,kw.142,0,0,0,Nmdm(utype,subMethod,SUBMETHOD_EA)},
{"ego",8,5,1,0,kw.142,0,0,0,Nmdm(utype,subMethod,SUBMETHOD_EGO)},
{"lhc",8,0,1,0,0,0,0,0,Nmdm(utype,subMethod,SUBMETHOD_LHS)},
{"rng",8,2,2,0,kw.136},
{"sbo",8,3,1,0,kw.142,0,0,0,Nmdm(utype,subMethod,SUBMETHOD_SBO)}  
}

KeyWord kw.144[2] [static]
Initial value:

= 

{"complementary",8,0,1,1,0,0,0,0,0,0,Nmdm(type,distributionType,COMPLEMENTARY)},
{"cumulative",8,0,1,1,0,0,0,0,0,0,Nmdm(type,distributionType,CUMULATIVE)}

KeyWord kw.145[1] [static]
Initial value:

= 

{"num_gen_reliability_levels",13,0,1,0,0,0,0,0,0,Nmdm(num_resplevs,genReliabilityLevels)}

KeyWord kw.146[1] [static]
Initial value:

= 

{"num_probability_levels",13,0,1,0,0,0,0,0,0,Nmdm(num_resplevs,probabilityLevels)}

KeyWord kw.147[3] [static]
Initial value:

= 

{"distribution",8,2,1,0,kw.144},
{"gen_reliability_levels",14,1,5,0,kw.145,0,0,0,0,Nmdm(resplevs,genReliabilityLevels)},
{"probability_levels",14,1,2,0,kw.146,0,0,0,0,Nmdm(resplevs01,probabilityLevels)}

KeyWord kw.148[3] [static]
Initial value:

= 

{"eval_id",8,0,2,0,0,0,0,0,0,Nmdm(augment_utype,approxExportFormat,TABULAR_EVAL_ID)},
{"header",8,0,1,0,0,0,0,0,0,Nmdm(augment_utype,approxExportFormat,TABULAR_HEADER)},
{"interface_id",8,0,3,0,0,0,0,0,0,Nmdm(augment_utype,approxExportFormat,TABULAR_IFACE_ID)}
KeyWord kw_149[3] [static]
Initial value:
= {
  {"annotated",8,0,0,0,0,0,0,N_mdm(utype,approxExportFormat_TABULAR_ANNOTATED)},
  {"custom_annotated",8,3,0,0,0,0,0,N_mdm(utype,approxExportFormat_TABULAR_NONE)},
  {"freeform",8,0,0,0,0,0,0,N_mdm(utype,approxExportFormat_TABULAR_NONE)}
}

KeyWord kw_150[3] [static]
Initial value:
= {
  {"eval_id",8,0,2,0,0,0,0,0,N_mdm(augment_utype,approxImportFormat_TABULAR_EVAL_ID)},
  {"header",8,0,0,0,0,0,0,0,N_mdm(augment_utype,approxImportFormat_TABULAR_HEADER)},
  {"interface_id",8,0,3,0,0,0,0,0,N_mdm(augment_utype,approxImportFormat_TABULAR_IFACE_ID)}
}

KeyWord kw_151[4] [static]
Initial value:
= {
  {"active_only",8,0,2,0,0,0,0,0,N_mdm(true,approxImportActive)},
  {"annotated",8,0,0,0,0,0,0,N_mdm(utype,approxImportFormat_TABULAR_ANNOTATED)},
  {"custom_annotated",8,3,1,0,kw_148,0,0,0,N_mdm(utype,approxImportFormat_TABULAR_NONE)},
  {"freeform",8,0,0,0,0,0,0,N_mdm(utype,approxImportFormat_TABULAR_NONE)}
}

KeyWord kw_152[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_153[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_152}
}

KeyWord kw_154[2] [static]
Initial value:
= {
  {"compute",8,3,2,0,kw_153},
  {"num_response_levels",13,0,1,0,0,0,0,0,N_mdm(num_resplevs,responseLevels)}
}
KeyWord kw_155[2]  [static]
Initial value:

```c
= {
    {"mt19937",8,0,1,1,0,0,0,0,N,mdm(lit,rngName_m19937)},
    {"rnum2",8,0,1,1,0,0,0,0,N,mdm(lit,rngName_rnum2)}
}
```

KeyWord kw_156[14]  [static]
Initial value:

```c
= {
    {0,0,1,0,0,kw},
    {0,0,3,0,0,kw},
    {"export_points_file",11,3,4,0,kw_149,0,0,0,N,mdm(str,approxExportFile)},
    {"import_points_file",11,4,3,0,kw_151,0,0,0,N,mdm(str,approxImportFile)},
    {"response_levels",14,2,8,0,kw_154,0,0,0,N,mdm(resplevs,responseLevels)},
    {"rng",8,2,7,0,kw_155},
    {"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",8,0,1,1,0,0,0,0,0,N,mdm(utype,reliabilitySearchType_EGRA_K)},
    {"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",8,0,1,1,0,0,0,0,0,N,mdm(utype,reliabilitySearchType_EGRA_X)}
}
```

KeyWord kw_157[2]  [static]
Initial value:

```c
= {
    {"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_158[3]  [static]
Initial value:

```c
= {
    {"iterator_scheduling",8,2,2,0,kw_157},
    {"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_159[1]  [static]
Initial value:

```c
= {
    {"model_pointer_list",11,0,1,0,0,0,0,0,N,mdm(strl,hybridModelPointers)}
}
```
KeyWord kw_160[2]  [static]
Initial value:

= {
    "method_name_list",15,1,1,kw_159,0.,0.,0,N_mdm(strL,hybridMethodNames)
}

KeyWord kw_161[1]  [static]
Initial value:

= {
    "global_model_pointer",11,0,1,0,0,0.,0.,0,N_mdm(str,hybridGlobalModelPointer)
}

KeyWord kw_162[1]  [static]
Initial value:

= {
    "local_model_pointer",11,0,1,0,0,0.,0.,0,N_mdm(str,hybridLocalModelPointer)
}

KeyWord kw_163[5]  [static]
Initial value:

= {
    "global_method_name",11,1,1,kw_161,0.,0.,0,N_mdm(str,hybridGlobalMethodName)
}

KeyWord kw_164[1]  [static]
Initial value:

= {
    "model_pointer_list",11,0,1,0,0,0.,0.,0,N_mdm(strL,hybridModelPointers)
}

KeyWord kw_165[2]  [static]
Initial value:

= {
    "method_name_list",15,1,1,kw_164,0.,0.,0,N_mdm(strL,hybridMethodNames)
}
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_166[6] [static]
Initial value:

= {  
  {0,0,3,0,kw.158},
  {"collaborative",8,2,1,1,kw_160,0,0,0,N,mdm(utype,subMethod_SUBMETHOD_COLLABORATIVE)},
  {"coupled",0,5,1,1,kw_163,0,0,0,1,N,mdm(utype,subMethod_SUBMETHOD_EMBEDDED)},
  {"embedded",8,5,1,1,kw_163,0,0,0,0,N,mdm(utype,subMethod_SUBMETHOD_EMBEDDED)},
  {"sequential",8,2,1,1,kw_165,0,0,0,0,N,mdm(utype,subMethod_SUBMETHODSEQUENTIAL)}
}

KeyWord kw_167[2] [static]
Initial value:

= {  
 {"parallel",8,0,1,1,0,0,0,0,0,N,mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
  {"series",8,0,1,1,0,0,0,0,0,N,mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
}

KeyWord kw_168[3] [static]
Initial value:

= {  
  {"generelabilities",8,0,1,1,0,0,0,0,0,N,mdm(type,responseLevelTarget_GENRELABILITIES)},
  {"probabilities",8,0,1,1,0,0,0,0,0,N,mdm(type,responseLevelTarget_PROBABILITIES)},
  {"system",8,2,2,0,kw.167}
}

KeyWord kw_169[2] [static]
Initial value:

= {  
  {"compute",8,3,2,0,kw.168},
  {"num_response_levels",13,0,1,0,0,0,0,0,0,N,mdm(num_resplevs,responseLevels)}
}

KeyWord kw_170[8] [static]
Initial value:

= {  
  {0,0,1,0,0,kw.27},
  {0,0,2,0,0,kw.28},
  {0,0,4,0,0,kw.33},
  {"adapt_import",8,0,1,1,0,0,0,0,0,N,mdm(utype,integrationRefine_AIS)},
  {"import",8,0,1,1,0,0,0,0,0,N,mdm(utype,integrationRefine_IS)},
  {"mmadapt_import",8,0,1,1,0,0,0,0,0,N,mdm(utype,integrationRefine_MMAIS)},
  {"refinement_samples",9,0,2,0,0,0,0,0,0,N,mdm(int,refineSamples)},
  {"responseLevels",14,2,3,0,kw_169,0,0,0,0,N,mdm(resplevs,responseLevels)}
}
KeyWord kw_171[3]  [static]
Initial value:

    = {
        {"eval_id",8,0,2,0,0,0,0,0,N,mdm(augment_utype,pstudyFileFormat_TABULAR_EVAL_ID)},
        {"header",8,0,1,0,0,0,0,0,N,mdm(augment_utype,pstudyFileFormat_TABULAR_HEADER)},
        {"interface_id",8,0,3,0,0,0,0,0,N,mdm(augment_utype,pstudyFileFormat_TABULAR_IFACE_ID)}
    }

KeyWord kw_172[4]  [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(utype,pstudyFileFormat_TABULAR_ANNOTATED)},
        {"custom_annotated",8,3,1,0,kw_171,0,0,0,N,mdm(utype,pstudyFileFormat_TABULAR_NONE)},
        {"freeform",8,0,1,0,0,0,0,0,N,mdm(utype,pstudyFileFormat_TABULAR_NONE)}
    }

KeyWord kw_173[3]  [static]
Initial value:

    = {
        {0,0,1,0,0,0,0,0,0,N,mdm(str,pstudyFilename)},
        {"list_of_points",14,0,1,1,0,0,0,0,N,mdm(RealDL,listOfPoints)}
    }

KeyWord kw_174[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_175[1]  [static]
Initial value:

    = {
        {"num_gen_reliability_levels",13,0,1,0,0,0,0,0,N,mdm(num_resplevs,genReliabilityLevels)}
    }

KeyWord kw_176[1]  [static]
Initial value:

    = {
        {"num_probability_levels",13,0,1,0,0,0,0,0,N,mdm(num_resplevs,probabilityLevels)}
    }
12.1. **DAKOTA NAMESPACE REFERENCE**

**KeyWord kw.177[2]**  
**[static]**

Initial value:

```plaintext
= {
    {"parallel",0,1,1,0,0,0,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL),
     "series",0,1,1,0,0,0,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
}
```

**KeyWord kw.178[3]**  
**[static]**

Initial value:

```plaintext
= {
    {"gen_reliabilities",0,1,1,0,0,0,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES),
     "probabilities",0,1,1,0,0,0,0,N_mdm(type,responseLevelTarget_PROBABILITIES),
     "system",0,2,0,kw.177}
}
```

**KeyWord kw.179[2]**  
**[static]**

Initial value:

```plaintext
= {
    {"compute",0,3,2,0,kw.178},
    {"num_response_levels",13,0,1,0,0,0,0,0,N_mdm(num_resplevs,responseLevels)}
}
```

**KeyWord kw.180[7]**  
**[static]**

Initial value:

```plaintext
= {
    0,0,1,0,0,kw.27},
    {"distribution",0,2,5,0,kw.176},
    {"gen_reliability_levels",0,1,1,0,0,0,0,0,N_mdm(resplevs,genReliabilityLevels),
     "nip",0,0,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_JIP),
     "probability_levels",0,1,1,0,0,0,0,0,N_mdm(resplevs01,probabilityLevels),
     "response_levels",0,1,1,0,0,0,0,0,N_mdm(resplevs,responseLevels),
     "sgp",0,0,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_SQP)}
}
```

**KeyWord kw.181[3]**  
**[static]**

Initial value:

```plaintext
= {
    0,0,1,0,0,kw.27},
    {"nip",0,0,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_JIP),
     "sgp",0,0,1,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_SQP)}
}
```

**KeyWord kw.182[5]**  
**[static]**

Initial value:

```plaintext
= {
    {"adapt_import",0,1,1,0,0,0,0,0,N_mdm(utype,integrationRefine_AIS),
     "import",0,1,1,0,0,0,0,0,N_mdm(utype,integrationRefine_AIS),
     "mm_adapt_import",0,0,1,1,0,0,0,0,N_mdm(utype,integrationRefine_MMAIS),
     "refinement_samples",0,0,2,0,0,0,0,0,N_mdm(int.refineSamples),
     "seed",0x19,0,3,0,0,0,0,0,N_mdm(pint,randomSeed)}
}
```
KeyWord kw_183[4]  [static]
Initial value:

```c
= {
    {"first_order",8,0,1,1,0,0,0,0,N_mdm(lit,reliabilityIntegration_first_order)},
    {"probability_refinement",8,5,2,0,kw_182},
    {"sample_refinement",0,5,2,0,kw_182,0,0,-1},
    {"second_order",8,0,1,1,0,0,0,0,N_mdm(lit,reliabilityIntegration_second_order)}
}
```

KeyWord kw_184[10]  [static]
Initial value:

```c
= {
    {"integration",8,4,3,0,kw_183},
    {"nip",8,0,2,0,0,0,0,0,N_mdm(utype,subMethod_SUBMETHOD_NIP)},
    {"no_approx",8,0,1,0,0,0,0,0,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_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_PLUS_U)},
    {"x_two_point",8,0,1,1,0,0,0,0,N_mdm(utype,reliabilitySearchType_TANA_PLUS_U)}
}
```

KeyWord kw_185[1]  [static]
Initial value:

```c
= {
    {"num_reliability_levels",13,0,1,0,0,0,0,0,N_mdm(num_resplevs,reliabilityLevels)}
}
```

KeyWord kw_186[2]  [static]
Initial value:

```c
= {
    {"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_187[4]  [static]
Initial value:

```c
= {
    {"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_186}
}
```
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_188[2] [static]
Initial value:
= {
    {"compute", 8, 4, 2, kw_187},
    {"num_response_levels", 13, 0, 1, 0, 0, 0, 0, 0, Nmdm(num_reslevs, responseLevels)}
}

KeyWord kw_189[5] [static]
Initial value:
= {
    {0, 0, 1, 0, 0, kw_27},
    {0, 0, 3, 0, kw_147},
    {"mpp_search", 8, 10, 1, 0, kw_184},
    {"reliability_levels", 14, 1, 3, 0, kw_185, 0.0, 0.0, 0.0, Nmdm(reslevs, reliabilityLevels)},
    {"response_levels", 14, 2, 2, 0, kw_188, 0.0, 0.0, Nmdm(reslevs, responseLevels)}
}

KeyWord kw_190[9] [static]
Initial value:
= {
    {0, 0, 1, 0, 0, kw_27},
    {0, 0, 9, 0, kw_44},
    {"display_all_evaluations", 8, 0, 7, 0, 0, 0, 0, 0, Nmdm(true, showAllEval)},
    {"display_format", 11, 0, 4, 0, 0, 0, 0, 0, Nmdm(str, displayFormat)},
    {"function_precision", 10, 0, 1, 0, 0, 0, 0, 0, Nmdm(Real, functionPrecision)},
    {"history_file", 11, 0, 3, 0, 0, 0, 0, 0, Nmdm(str, historyFile)},
    {"neighbor_order", 0x19, 0, 6, 0, 0, 0, 0, 0, Nmdm(pint, neighborOrder)},
    {"seed", 0x19, 0, 2, 0, 0, 0, 0, 0, 0, Nmdm(pint, randomSeed)},
    {"variable_neighborhood_search", 10, 0, 5, 0, 0, 0, 0, 0, Nmdm(Real, vns)}
}

KeyWord kw_191[2] [static]
Initial value:
= {
    {num_offspring", 0x19, 0, 2, 0, 0, 0, 0, 0, 0, Nmdm(pintz, numOffspring)},
    {num_parents", 0x19, 0, 1, 0, 0, 0, 0, 0, 0, Nmdm(pintz, numParents)}
}

KeyWord kw_192[5] [static]
Initial value:
= {
    {crossover_rate", 10, 0, 2, 0, 0, 0, 0, 0, 0, Nmdm(litz, TYPE_DATA, crossoverType, null, crossover)},
    {multi_point_binary", 9, 0, 1, 1, 0, 0, 0, 0, 0, Nmdm(lit2p, TYPE_DATA, crossoverType, multi_point, binary)},
    {multi_point parameterized_binary", 9, 0, 1, 1, 0, 0, 0, 0, 0, Nmdm(lit2p, TYPE_DATA, crossoverType, multi_point, parameterized_binary)},
    {multi_point real", 9, 0, 1, 1, 0, 0, 0, 0, 0, Nmdm(lit2p, TYPE_DATA, crossoverType, multi_point, real)},
    {shuffle_random", 8, 2, 1, 1, kw_191, 0.0, 0.0, 0.0, Nmdm(litc, TYPE_DATA, crossoverType, shuffle_random)}
}
KeyWord kw_193[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_194[1]  [static]
Initial value:
= {
    {"mutation_scale",10,0,1,0,0,,0,0,0,N_mdm(Real01,mutationScale)}
}

KeyWord kw_195[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_194,0,,0,0,N_mdm(litc,TYPE_DATA,mutationType,offset_cauchy)},
    {"offset_normal",8,1,1,1,kw_194,0,,0,0,N_mdm(litc,TYPE_DATA,mutationType,offset_normal)},
    {"offset_uniform",8,1,1,1,kw_194,0,,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_196[7]  [static]
Initial value:
= {
    {"crossover_type",8,5,5,0,kw_192},
    {"initialization_type",8,3,4,0,kw_193},
    {"log_file",11,0,2,0,0,0,,0,0,N_mdm(str,logFile)},
    {"mutation_type",8,6,6,0,kw_195},
    {"population_size",8x29,5,1,5,0,0,,0,0,N_mdm(nnint,populationSize)},
    {"print_each_pop",8,0,3,0,0,0,,0,0,N_mdm(true,printPopFlag)},
    {"seed",8x19,0,7,0,0,0,,0,0,N_mdm(pint,randomSeed)}
}

KeyWord kw_197[3]  [static]
Initial value:
= {
    {"metric_tracker",8,0,1,1,0,0,,0,0,N_mdm(lit,convergenceType,metric_tracker)},
    {"num_generations",8x29,0,3,0,0,0,,0,0,N_mdm(sizet,numGenerations)},
    {"percent_change",10,0,2,0,0,0,,0,0,N_mdm(Realz,convergenceTolerance)}
}
KeyWord kw_198[2] [static]
Initial value:
= {
    {"domination_count",8,0,1,0,0,0,0,Nmdm(lit,fitnessType_domination_count)},
    {"layer_rank",8,0,1,0,0,0,0,Nmdm(lit,fitnessType_layer_rank)}
}

KeyWord kw_199[1] [static]
Initial value:
= {
    {"num_designs",0x29,0,1,0,2,0,0,0,Nmdm(pintz,numDesigns)}
}

KeyWord kw_200[3] [static]
Initial value:
= {
    {"distance",14,0,1,0,0,0,0,Nmdm(RealLit,TYPE_DATA_nichingType_distance)},
    {"max_designs",14,1,1,kw_199,0,0,0,Nmdm(RealLit,TYPE_DATA_nichingType_max_designs)},
    {"radial",14,0,1,0,0,0,0,Nmdm(RealLit,TYPE_DATA_nichingType_radial)}
}

KeyWord kw_201[1] [static]
Initial value:
= {
    {"orthogonal_distance",14,0,1,0,0,0,0,0,Nmdm(RealLit,TYPE_DATA_postProcessorType_distance_postprocessor)}
}

KeyWord kw_202[2] [static]
Initial value:
= {
    {"shrinkage_fraction",10,0,1,0,0,0,0,0,Nmdm(RealLit,shrinkagePercent)},
    {"shrinkage_percentage",2,0,1,0,0,0,-1,Nmdm(RealLit,shrinkagePercent)}
}

KeyWord kw_203[4] [static]
Initial value:
= {
    {"below_limit",10,2,1,1,kw_202,0,0,0,0,Nmdm(litp,TYPE_DATA_replacementType_below_limit)},
    {"elitist",8,0,1,0,0,0,0,0,Nmdm(lit,replacementType_elitist)},
    {"roulette_wheel",8,0,1,1,0,0,0,0,Nmdm(lit,replacementType_roulette_wheel)},
    {"unique_roulette_wheel",8,0,1,1,0,0,0,0,Nmdm(lit,replacementType_unique_roulette_wheel)}
}
## Chapter 12. Namespace Documentation

**KeyWord kw_204[8]**  [static]

Initial value:

```plaintext
= {
    {0,0,1,0,0,kw_27},
    {0,0,9,0,0,kw_44},
    {0,0,7,0,0,kw_196},
    {"convergence_type",8,3,4,0,kw_197},
    {"fitness_type",8,2,1,0,kw_198},
    {"niching_type",8,3,3,0,kw_200},
    {"postprocessor_type",8,1,5,0,kw_201},
    {"replacement_type",8,4,2,0,kw_203}
}
```

**KeyWord kw_205[1]**  [static]

Initial value:

```plaintext
= {
    {"model_pointer",11,0,1,0,0,,0,0,0,N_mdm(str,subModelPointer)}
}
```

**KeyWord kw_206[1]**  [static]

Initial value:

```plaintext
= {
    {"seed",9,0,1,0,0,,0,0,0,N_mdm(int,randomSeed)}
}
```

**KeyWord kw_207[5]**  [static]

Initial value:

```plaintext
= {
    {0,0,3,0,0,kw_158},
    {"method_name",11,1,1,1,kw_205,0,,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_206,0,,0,0,N_mdm(int,concurrentRandomJobs)},
    {"starting_points",14,0,3,0,0,,0,0,0,N_mdm(RealDL,concurrentParameterSets)}
}
```

**KeyWord kw_208[2]**  [static]

Initial value:

```plaintext
= {
    {0,0,1,0,0,kw_27},
    {"partitions",13,0,1,1,0,,0,0,0,N_mdm(usharray,varPartitions)}
}
```

**KeyWord kw_209[5]**  [static]

Initial value:

```plaintext
= {
    {0,0,1,0,0,kw_27},
    {"minBoxsize_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)},
    {"volumeBoxsize_limit",10,0,3,0,0,,0,0,0,N_mdm(Real,volBoxSize)}
}
```
KeyWord kw_210[10] [static]
Initial value:

= {
    {0,0,1,0,0,kw_27},
    "absolute_conv_tol",10,0,2,0,0,0,0,0,0,N_mdm(Real,absConvTol),
    "covariance",9,0,0,0,0,0,0,0,N_mdm(int,covarianceType),
    "false_conv_tol",10,0,6,0,0,0,0,0,0,N_mdm(Real,falseConvTol),
    "function_precision",10,0,1,0,0,0,0,0,0,N_mdm(Real,functionPrecision),
    "initial_trust_radius",10,0,7,0,0,0,0,0,0,N_mdm(Real,initTRRadius),
    "regression_diagnostics",8,0,9,0,0,0,0,0,0,N_mdm(true,regressDiag),
    "singular_conv_diagnostics",8,0,9,0,0,0,0,0,0,N_mdm(true,singConvDiag),
    "singular_radius",10,0,5,0,0,0,0,0,0,N_mdm(Real,singRadius),
    "x_conv_tol",10,0,3,0,0,0,0,0,0,N_mdm(Real,xConvTol)
}

KeyWord kw_211[2] [static]
Initial value:

= {
    "global",8,0,1,1,0,0,0,0,0,N_mdm(lit,lipschitzType_global),
    "local",8,0,1,1,0,0,0,0,0,N_mdm(lit,lipschitzType_local)
}

KeyWord kw_212[2] [static]
Initial value:

= {
    "parallel",8,0,1,1,0,0,0,0,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL),
    "series",8,0,1,1,0,0,0,0,0,N_mdm(type,responseLevelTargetReduce_SYSTEM_SERIES)
}

KeyWord kw_213[3] [static]
Initial value:

= {
    "gen_reliabilities",8,0,1,1,0,0,0,0,0,N_mdm(type,responseLevelTarget_GEN_RELIABILITIES),
    "probabilities",8,0,1,1,0,0,0,0,0,N_mdm(type,responseLevelTarget_PROBABILITIES),
    "system",8,2,2,0,kw_212
}

KeyWord kw_214[2] [static]
Initial value:

= {
    "compute",8,3,2,0,kw_213,
    "num_response_levels",13,0,1,0,0,0,0,0,N_mdm(num_resplevs,responseLevels)
}
KeyWord kw.215[6] [static]
Initial value:

```
= {
    {0,0,1,0,0,kw.27},
    {0,0,2,0,0,kw.28},
    {0,0,4,0,0,kw.33},
    {"emulator_samples",9,0,2,0,0,.0,.0,.0,Nmdm(int,emulatorSamples)},
    {"lipschitz",8,2,1,0,kw.211},
    {"response_levels",14,2,3,0,kw.214,0,.0,.0,Nmdm(resplevs,responseLevels)}
}
```

KeyWord kw.216[1] [static]
Initial value:

```
= {
    {"num_reliability_levels",13,0,1,0,.0,.0,.0,.0,Nmdm(num_resplevs,reliabilityLevels)}
}
```

KeyWord kw.217[2] [static]
Initial value:

```
= {
    {"parallel",8,0,1,1,0,.0,.0,.0,.0,Nmdm(type,responseLevelTargetReduce_SYSTEM_PARALLEL)},
    {"series",8,0,1,1,0,.0,.0,.0,.0,Nmdm(type,responseLevelTargetReduce_SYSTEM_SERIES)}
}
```

KeyWord kw.218[4] [static]
Initial value:

```
= {
    {"gen_reliabilities",8,0,1,1,.0,.0,.0,.0,.0,Nmdm(type,responseLevelTarget_GEN_RELIABILITIES)},
    {"probabilities",8,0,1,1,.0,.0,.0,.0,.0,Nmdm(type,responseLevelTarget_PROBABILITIES)},
    {"reliabilities",8,0,1,1,.0,.0,.0,.0,.0,Nmdm(type,responseLevelTarget_RELIABILITIES)},
    {"system",8,2,2,0,kw.217}
}
```

KeyWord kw.219[2] [static]
Initial value:

```
= {
    {"compute",8,4,2,0,kw.218},
    {"num_response_levels",13,0,1,0,.0,.0,.0,.0,Nmdm(num_resplevs,responseLevels)}
}
```

KeyWord kw.220[2] [static]
Initial value:

```
= {
    {"reliability_levels",14,1,1,0,kw.216,.0,.0,.0,.0,Nmdm(resplevs,reliabilityLevels)},
    {"response_levels",14,2,2,0,kw.219,.0,.0,.0,.0,Nmdm(resplevs,responseLevels)}
```
KeyWord kw_221[1]  [static]
Initial value:
= {
    {"fixed_seed", 8, 0, 1, 0, 0, 0, 0, 0, N, mmd(true, fixedSeedFlag)}
}

KeyWord kw_222[3]  [static]
Initial value:
= {
    {"eval_id", 8, 0, 1, 0, 0, 0, 0, 0, N, mmd(augment_utype, approxImportFormat_TABULAR_EVAL_ID)},
    {"header", 8, 0, 1, 0, 0, 0, 0, 0, N, mmd(augment_utype, approxImportFormat_TABULAR_HEADER)},
    {"interface_id", 8, 0, 3, 0, 0, 0, 0, 0, N, mmd(augment_utype, approxImportFormat_TABULAR_IFACE_ID)}
}

KeyWord kw_223[4]  [static]
Initial value:
= {
    {"active_only", 8, 0, 2, 0, 0, 0, 0, 0, N, mmd(true, approxImportActive)},
    {"annotated", 8, 0, 1, 0, 0, 0, 0, 0, N, mmd(utype, approxImportFormat_TABULAR_ANNOTATED)},
    {"custom_annotated", 8, 3, 1, 0, kw_222[0], 0, 0, 0, N, mmd(utype, approxImportFormat_TABULAR_NONE)},
    {"freeform", 8, 0, 1, 0, 0, 0, 0, 0, N, mmd(utype, approxImportFormat_TABULAR_NONE)}
}

KeyWord kw_224[1]  [static]
Initial value:
= {
    {"import_points_file", 11, 4, 1, 0, kw_223[0], 0, 0, 0, N, mmd(str, approxImportFile)}
}

KeyWord kw_225[2]  [static]
Initial value:
= {
    {"advancements", 9, 0, 1, 0, 0, 0, 0, 0, N, mmd(ushint, adaptedBasisAdvancements)},
    {"soft_convergence_limit", 9, 0, 2, 0, 0, 0, 0, 0, N, mmd(ushint, softConvLimit)}
}

KeyWord kw_226[3]  [static]
Initial value:
= {
    {"adapted", 8, 2, 1, 1, kw_225[0], 0, 0, 0, N, mmd(type, expansionBasisType_ADAPTED_BASIS_EXPANDING_FRONT)},
    {"tensor_product", 8, 0, 1, 1, 0, 0, 0, 0, N, mmd(type, expansionBasisType_TENSOR_PRODUCT_BASIS)},
    {"total_order", 8, 0, 1, 1, 0, 0, 0, 0, N, mmd(type, expansionBasisType_TOTAL_ORDER_BASIS)}
}
KeyWord kw_227[1]  [static]
Initial value:
= {
    {"noise_tolerance",14,0,1,0,0,0.,0.,0.,N_{mdm}(Real,regressionNoiseTol)}
}

KeyWord kw_228[1]  [static]
Initial value:
= {
    {"noise_tolerance",14,0,1,0,0,0.,0.,0.,N_{mdm}(Real,regressionNoiseTol)}
}

KeyWord kw_229[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}(Real,regressionNoiseTol)}
}

KeyWord kw_230[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_231[1]  [static]
Initial value:
= {
    {"noise_tolerance",14,0,1,0,0,0.,0.,0.,N_{mdm}(Real,regressionNoiseTol)}
}

KeyWord kw_232[17]  [static]
Initial value:
= {
    {"basis_pursuit",8,0,2,0,0,0,0.,0.,0.,N_{mdm}(type,regressionType_BASIS_Pursuit)},
    {"basis_pursuit_denoising",8,1,2,0,kw_227,0.,0.,0.,N_{mdm}(type,regressionType_BASIS_Pursuit_DENOISING)},
    {"bpdn",0,0,2,0,0,0,0.,0.,0.,N_{mdm}(type,regressionType_BASIS_Pursuit_DENOISING)},
    {"cross_validation",8,0,3,0,0,0,0.,0.,0.,N_{mdm}(true, crossValidation)},
    {"lars",0,1,2,0,kw_228,0.,0.,0.,N_{mdm}(type,regressionType_LEAST_ANGLE_REGRESSION)},
    {"lasso",0,0,2,0,kw_229,0.,0.,0.,N_{mdm}(type,regressionType_LASSO_REGRESSION)},
    {"least_absolute_shrinkage",8,2,2,0,kw_229,0.,0.,0.,N_{mdm}(type,regressionType_LASSO_REGRESSION)},
    {"least_angle_regression",8,1,2,0,kw_228,0.,0.,0.,N_{mdm}(type,regressionType_LEAST_ANGLE_REGRESSION)},
12.1. DAKOTA NAMESPACE REFERENCE

```
{ "least_squares", 8, 2, 2, 0, kw_230, 0, 0, 0, N, mdm(type, regressionType_DEFAULT_LEAST_SQ_REGRESSION),
  [ "comp", 0, 1, 2, 0, kw_231, 0, 0, 0, N, mdm(type, regressionType_ORTHOG_MATCH_Pursuit)],
  [ "orthogonal_matching_pursuit", 8, 1, 2, 0, kw_231, 0, 0, 0, N, mdm(type, regressionType_ORTHOG_MATCH_Pursuit)],
  [ "ratio_order", 10, 0, 1, 0, 0, 0, 0, 0, N, mdm(Realp, collocationRatioTermsOrder)],
  [ "reuse_points", 8, 0, 0, 0, 0, 0, 0, 0, N, mdm] ],
  [ "reuse_samples", 0, 0, 0, 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_233[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, -1, N, mdm(lit, pointReuse_all)]
}
```

**KeyWord kw_234[6]** [static]
Initial value:

```
= {
  [ 0, 0, 1, 0, 0, kw_224],
  [ collocation_points", 13, 17, 3, 1, kw_232, 0, 0, 0, 0, N, mdm(szarray, collocationPoints)],
  [ collocation_ratio", 10, 17, 3, 1, kw_232, 0, 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, 2, 1, kw_233, 0, 0, 0, 0, N, mdm(szarray, expansionSamples)]
}
```

**KeyWord kw_235[3]** [static]
Initial value:

```
= {
  [ evalid", 8, 0, 2, 0, 0, 0, 0, 0, N, mdm(augment_utype, approxExportFormat_TABULAR_EVAL_ID)],
  [ header", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(augment_utype, approxExportFormat_TABULAR_HEADER)],
  [ interface_id", 8, 0, 3, 0, 0, 0, 0, 0, N, mdm(augment_utype, approxExportFormat_TABULAR_IFACE_ID)]
}
```

**KeyWord kw_236[3]** [static]
Initial value:

```
= {
  [ annotated", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(utype, approxExportFormat_TABULAR_ANNOTATED)],
  [ custom_annotated", 8, 3, 1, 0, kw_235, 0, 0, 0, 0, N, mdm(utype, approxExportFormat_TABULAR_NONE)],
  [ freeform", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(utype, approxExportFormat_TABULAR_NONE)]
}
```
KeyWord kw_237[6] [static]
Initial value:
= {
  {0,0,1,0,0,kw_224},
  "collocation_points",13,0,1,1,0,0,0,0,N,mdm(szarray,collocationPoints)},
  "cross_validation",8,0,1,0,0,0,0,0,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",13,0,1,0,0,0,0,0,N,mdm(usharray,tensorGridOrder)}

KeyWord kw_238[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_239[2] [static]
Initial value:
= {
  "dimension_adaptive",8,3,1,1,kw_238},
  "uniform",8,0,1,1,0,0,0,0,N,mdm(type,refinementControl_UNIFORM_CONTROL)}

KeyWord kw_240[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",9,5,2,0,0,0,0,0,N,mdm(int,refineSamples)}

KeyWord kw_241[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_NONNESTED)}

KeyWord kw_242[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) }
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_243[3] [static]
Initial value:

= {
  {0,0,3,0,0,kw_241},
  {"restricted",8,0,1,0,0,0,0,0,Nmdm(type,growthOverride,RESTRICTED)},
  {"unrestricted",8,0,1,0,0,0,0,0,Nmdm(type,growthOverride,UNRESTRICTED)}
}

KeyWord kw_244[2] [static]
Initial value:

= {
  {"drop_tolerance",10,0,2,0,0,0,0,0,Nmdm(Real,vbdDropTolerance)},
  {"interaction_order",0x19,0,0,0,0,0,0,0,Nmdm(ushint,vbdOrder)}
}

KeyWord kw_245[25] [static]
Initial value:

= {
  {0,0,1,0,0,kw_27},
  {0,0,2,0,0,kw_28},
  {0,0,4,0,0,kw_33},
  {0,0,2,0,0,kw_220},
  {0,0,1,0,0,kw_221},
  {"askey",8,0,2,0,0,0,0,0,Nmdm(type,expansionType,ASKEY_U)},
  {"cubature_integrand",9,0,3,1,0,0,0,0,Nmdm(ushint,cubIntOrder)},
  {"diagonal_covariance",8,0,5,0,0,0,0,0,Nmdm(type,covarianceControl,DIAGONAL_COVARIANCE)},
  {"expansion_order",13,5,3,1,kw_234,0.,0.,0,Nmdm(usharray,expansionOrder)},
  {"export_expansion_file",11,0,10,0,0,0,0,0,Nmdm(str,expansionExportFile)},
  {"full_covariance",8,0,5,0,0,0,0,0,Nmdm(type,covarianceControl,FULL_COVARIANCE)},
  {"import_expansion_file",11,0,3,1,0,0,0,0,0,Nmdm(str,expansionImportFile)},
  {"least_interpolation",0,5,3,1,kw_237,0.,0,3,Nmdm(type,regressionType,ORTHOG_LEAST_INTERPOLATION)},
  {"normalized",8,0,6,0,0,0,0,0,Nmdm(true,normalizedCoeffs)},
  {"oli",8,0,5,3,1,kw_237,0.,0,1,Nmdm(type,regressionType,ORTHOG_LEAST_INTERPOLATION)},
  {"orthogonal_least_interpolation",8,5,3,1,kw_237,0.,0,3,Nmdm(type,regressionType,ORTHOG_LEAST_INTERPOLATION)},
  {"p_refinement",8,2,1,0,kw_239,0.,0,0,Nmdm(type,refinementType,P_REFINEMENT)},
  {"probability_refinement",8,4,8,0,kw_240},
  {"quadrature_order",13,3,3,1,kw_241,0.,0,0,Nmdm(usharray,quadratureOrder)},
  {"variance_based_decomp",8,2,4,0,kw_244,0.,0,0,Nmdm(true,vbdFlag)},
  {"wiener",8,0,2,0,0,0,0,0,Nmdm(type,expansionType,STD_NORMAL_U)}
}

KeyWord kw_246[1] [static]
Initial value:

= {
  {"previous_samples",9,0,1,1,0,0,0,0,0,Nmdm(int,previousSamples)}
}
KeyWord kw_247[4] [static]

Initial value:

= {
    "incremental_lhs", 8, 1, 1, kw_246, 0, 0, 0, N, mdm(utype, sampleType_SUBMETHOD_INCREMENTAL_LHS),
    "incremental_random", 8, 1, 1, kw_246, 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_248[1] [static]

Initial value:

= {
    "drop_tolerance", 10, 0, 1, 0, 0, 0, 0, 0, N, mdm(Real, vbdDropTolerance)
}

KeyWord kw_249[8] [static]

Initial value:

= {
    0, 0, 1, 0, 0, kw_27,
    0, 0, 2, 0, 0, kw_28,
    0, 0, 4, 0, 0, kw_33,
    0, 0, 2, 0, 0, kw_228,
    0, 0, 1, 0, 0, kw_221,
    "backfill", 8, 0, 3, 0, 0, 0, 0, 0, N, mdm(true, backfillFlag),
    "sample_type", 8, 4, 1, 0, kw_247,
    "variance_based_decomp", 8, 1, 1, 0, kw_248, 0, 0, 0, 0, N, mdm(true, vbdFlag)
}

KeyWord kw_250[3] [static]

Initial value:

= {
    "eval_id", 8, 0, 2, 0, 0, 0, 0, 0, N, mdm(augment_utype, approxExportFormat_TABULAR_EVAL_ID),
    "header", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(augment_utype, approxExportFormat_TABULAR_HEADER),
    "interface_id", 8, 0, 3, 0, 0, 0, 0, 0, N, mdm(augment_utype, approxExportFormat_TABULAR_IFACE_ID)
}

KeyWord kw_251[3] [static]

Initial value:

= {
    "annotated", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(utype, approxExportFormat_TABULAR_ANNOTATED),
    "custom_annotated", 8, 3, 1, 0, kw_250, 0, 0, 0, 0, N, mdm(utype, approxExportFormat_TABULAR_NONE),
    "freeform", 8, 0, 1, 0, 0, 0, 0, 0, N, mdm(utype, approxExportFormat_TABULAR_NONE)
}
KeyWord kw252[2] [static]
Initial value:

= {
   {"generalized",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,refinementControl,DIMENSION,ADAPTIVE,CONTROL,GENERALIZED)},
   {"sobol",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,refinementControl,DIMENSION,ADAPTIVE,CONTROL,SOBOL)}
}

KeyWord kw253[3] [static]
Initial value:

= {
   {"dimension_adaptive",8,2,1,1,kw252},
   {"local_adaptive",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,refinementControl,LOCAL,ADAPTIVE,CONTROL)},
   {"uniform",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,refinementControl,UNIFORM,CONTROL)}
}

KeyWord kw254[2] [static]
Initial value:

= {
   {"generalized",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,refinementControl,DIMENSION,ADAPTIVE,CONTROL,GENERALIZED)},
   {"sobol",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,refinementControl,DIMENSION,ADAPTIVE,CONTROL,SOBOL)}
}

KeyWord kw255[2] [static]
Initial value:

= {
   {"dimension_adaptive",8,2,1,1,kw254},
   {"uniform",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,refinementControl,UNIFORM,CONTROL)}
}

KeyWord kw256[4] [static]
Initial value:

= {
   {"adapt_import",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,integrationRefine,AIS)},
   {"import",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,integrationRefine,IS)},
   {"mm_adapt_import",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,integrationRefine,MMAIS)},
   {"refinement_samples",9,0,2,0,0.,0.,0.,0,N,mdm(int,refineSamples)}
}

KeyWord kw257[2] [static]
Initial value:

= {
   {"lhs",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,sampleType,SUBMETHOD,LHS)},
   {"random",8,0,1,1,0,0.,0.,0.,0,N,mdm(utype,sampleType,SUBMETHOD,RANDOM)}
}
KeyWord kw_258[4] [static]
Initial value:

```c++
= {
    {"hierarchical",8,0,2,0,0,0,0,0,0,N\_mdm(type,expansionBasisType,HIERARCHICAL\_INTERPOLANT)},
    {"nodal",8,0,2,0,0,0,0,0,0,N\_mdm(type,expansionBasisType,NODAL\_INTERPOLANT)},
    {"restricted",8,0,1,0,0,0,0,0,0,N\_mdm(type,growthOverride,RESTRICTED)},
    {"unrestricted",8,0,1,0,0,0,0,0,0,N\_mdm(type,growthOverride,UNRESTRICTED)}
}
```

KeyWord kw_259[2] [static]
Initial value:

```c++
= {
    {"drop\_tolerance",10,0,2,0,0,0,0,0,0,N\_mdm(Real,vbd\_DropTolerance)},
    {"interaction\_order",0x19,0,1,0,0,0,0,0,0,N\_mdm(ushint,vbd\_Order)}
}
```

KeyWord kw_260[23] [static]
Initial value:

```c++
= {
    0,0,1,0,0,kw_27,
    0,0,2,0,0,kw_28,
    0,0,4,0,0,kw_33,
    0,0,2,0,0,kw_220,
    0,0,1,0,0,kw_221,
    {"askey",8,0,2,0,0,0,0,0,0,N\_mdm(type,expansionType,ASKEY\_U)},
    {"diagonal\_covariance",8,0,6,0,0,0,0,0,0,N\_mdm(type,covarianceControl,DIAGONAL\_COVARIANCE)},
    {"dimension\_preference",14,0,4,0,0,0,0,0,0,N\_mdm(RealDL,aniso\_DimPref)},
    {"export\_points\_file",11,3,11,0,kw_251,0,0,0,0,N\_mdm(str,approx\_ExportFile)},
    {"full\_covariance",8,0,8,0,0,0,0,0,0,N\_mdm(type,covarianceControl,FULL\_COVARIANCE)},
    {"h\_refinement",8,3,1,0,kw_253,0,0,0,0,N\_mdm(type,refinementType,H\_REFINEMENT)},
    {"nested\_refinement",8,0,6,0,0,0,0,0,0,N\_mdm(type,nestingOverride,NESTED)},
    {"p\_refinement",8,2,1,0,kw_255,0,0,0,0,N\_mdm(type,refinementType,P\_REFINEMENT)},
    {"piecewise",8,0,2,0,0,0,0,0,0,N\_mdm(ushint,problemDesc\_DB:\method,piecewise)},
    {"probability\_refinement",8,4,10,0,kw_256},
    {"quadrature\_order",13,0,3,1,0,0,0,0,0,N\_mdm(usharray,quadrature\_Order)},
    {"sample\_refinement",0,4,10,0,kw_256,0,0,0,-2},
    {"sample\_type",8,2,9,0,kw_257},
    {"sparse\_grid\_level",13,4,3,1,kw_258,0,0,0,0,N\_mdm(usharray,sparse\_Grid\_Level)},
    {"use\_derivatives",8,0,5,0,0,0,0,0,0,N\_mdm(true,methodUse\_Derivs\_Flag)},
    {"variance\_based\_decomp",8,2,7,0,kw_259,0,0,0,0,N\_mdm(true,vbd\_Flag)},
    {"wiener",8,0,2,0,0,0,0,0,0,N\_mdm(type,expansionType,STD\_NORMAL\_U)}
}
```

KeyWord kw_261[2] [static]
Initial value:

```c++
= {
    0,0,1,0,0,kw_27,
    {"misc\_options",15,0,1,0,0,0,0,0,0,N\_mdm(strL,misc\_Options)}
}
```
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_262[3]  [static]
Initial value:
= {
  {"function_precision",10,0,2,0,0,0,0,0,N_mdm(Real,functionPrecision)},
  {"linesearch_tolerance",10,0,1,0,0,0,0,0,N_mdm(Real,lineSearchTolerance)},
  {"verify_level",9,0,1,0,0,0,0,0,N_mdm(int,verifyLevel)}
}

KeyWord kw_263[4]  [static]
Initial value:
= {
  {0,0,1,0,0,kw_27},
  {0,0,9,0,0,kw_44},
  {0,0,3,0,0,kw_262},
  {""}
}

KeyWord kw_264[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_265[4]  [static]
Initial value:
= {
  {0,0,1,0,0,kw_27},
  {0,0,9,0,0,kw_44},
  {0,0,2,0,0,kw_264},
  {""}
}

KeyWord kw_266[3]  [static]
Initial value:
= {
  {0,0,1,0,0,kw_27},
  {0,0,9,0,0,kw_44},
  {"search_scheme_size",9,0,1,0,0,0,0,0,N_mdm(int,searchSchemeSize)}
}

KeyWord kw_267[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_268[4]  [static]

Initial value:

= {
    
    "gradient_based_line_search",8,0,1,0,0.,0.,0,N,mdm(lit, searchMethod.gradient_based_line_search),
    
    "tr_pds",8,0,1,0,0.,0.,0,N,mdm(lit,searchMethod.tr_pds),
    
    "trust_region",8,0,1,0,0.,0.,0,N,mdm(lit,searchMethod.trust_region),
    
    "value_based_line_search",8,0,1,0,0.,0.,0,N,mdm(lit,searchMethod.value_based_line_search)

  }

KeyWord kw_269[7]  [static]

Initial value:

= {
    
    0,0,1,0,0,kw_27,
    
    0,0,9,0,0,kw_44,
    
    0,0,2,0,0,kw_264,
    
    "centering_parameter",10,0,4,0,0.,0.,0,N,mdm(Real,centeringParam),
    
    "merit_function",8,3,2,0,kw_267,
    
    "search_method",8,4,1,0,kw_268,
    
    "steplength_to_boundary",10,0,3,0,0.,0.,0,N,mdm(Real,stepLenToBoundary)

  }

KeyWord kw_270[5]  [static]

Initial value:

= {
    
    "debug",8,0,1,0,0.,0.,0.,0,N,mdm(type,methodOutput.DEBUG_OUTPUT),
    
    "normal",8,0,1,0,0.,0.,0.,0,N,mdm(type,methodOutput.NORMAL_OUTPUT),
    
    "quiet",8,0,1,0,0.,0.,0.,0,N,mdm(type,methodOutput.QUICK_OUTPUT),
    
    "slient",8,0,1,0,0.,0.,0.,0,N,mdm(type,methodOutput.SILENT_OUTPUT),
    
    "verbose",8,0,1,0,0.,0.,0.,0,N,mdm(type,methodOutput.VERBOSE_OUTPUT)

  }

KeyWord kw_271[2]  [static]

Initial value:

= {
    
    "model_pointer",11,0,1,0,0.,0.,0.,0,N,mdm(str,subModelPointer),
    
    "opt_model_pointer",1,0,1,0,0.,0.,-1,N,mdm(str,subModelPointer)

  }

KeyWord kw_272[1]  [static]

Initial value:

= {
    
    "seed",9,0,1,0,0.,0.,0.,0,N,mdm(int,randomSeed)

  }
12.1. **DAKOTA NAMESPACE REFERENCE**

**KeyWord kw.273** [static]

Initial value:

```plaintext
= {
    {0,0,3,0,0,kw.158},
    {"method_name",11,2,1,kw.271,0,0,0,0,N_mdm(str,subMethodName)},
    {"method_pointer",11,0,1,1,0,0,0,0,N_mdm(str,subMethodPointer)},
    {"multi_objective_weight_sets",6,0,3,0,0,0,0,0,4,N_mdm(RealDL, concurrentParameterSets)},
    {"opt_method_name",3,2,1,kw.271,0,0,0,0,0,N_mdm(str,subMethodName)},
    {"opt_method_pointer",3,0,1,1,0,0,0,0,-3,N_mdm(str,subMethodPointer)},
    {"random_weight_sets",9,1,2,0,kw.272,0,0,0,0,N_mdm(int, concurrentRandomJobs)},
    {"weight_sets",14,0,3,0,0,0,0,0,0,N_mdm(RealDL, concurrentParameterSets)}
}
```

**KeyWord kw.274** [static]

Initial value:

```plaintext
= {
    {0,0,1,0,0,kw.27},
    {0,0,2,0,0,kw.28},
    {"partitions",13,0,1,0,0,0,0,0,0,N_mdm(usharray, varPartitions)}
}
```

**KeyWord kw.275** [static]

Initial value:

```plaintext
= {
    {0,0,1,0,0,kw.27},
    {"converge_order",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod\_SUBMETHOD\_CONVERGE\_ORDER)},
    {"converge_qoi",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod\_SUBMETHOD\_CONVERGE\_QOI)},
    {"estimate_order",8,0,1,1,0,0,0,0,0,N_mdm(utype,subMethod\_SUBMETHOD\_ESTIMATE\_ORDER)},
    {"refinement_rate",10,0,2,0,0,0,0,0,0,N_mdm(Real, refinementRate)}
}
```

**KeyWord kw.276** [static]

Initial value:

```plaintext
= {
    {"num_generations",0x29,0,2,0,0,0,0,0,0,N_mdm(sizet,numGenerations)},
    {"percent_change",10,0,1,0,0,0,0,0,0,N_mdm(Realz, convergenceTolerance)}
}
```

**KeyWord kw.277** [static]

Initial value:

```plaintext
= {
    {"num_generations",0x29,0,2,0,0,0,0,0,0,N_mdm(sizet,numGenerations)},
    {"percent_change",10,0,1,0,0,0,0,0,0,N_mdm(Realz, convergenceTolerance)}
}
```
KeyWord kw_278[2]  [static]
Initial value:
= {
    {"average_fitness_tracker",8,2,1,1,kw_276,0,0,0,N,mdm(lit, convergenceType_average_fitness_tracker)},
    {"best_fitness_tracker",8,2,1,1,kw_277,0,0,0,N,mdm(lit, convergenceType_best_fitness_tracker)}
}

KeyWord kw_279[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_280[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_281[6]  [static]
Initial value:
= {
    {0,0,1,0,0,kw_27},
    {0,0,9,0,0,kw_44},
    {0,0,7,0,0,kw_196},
    {"convergence_type",8,2,3,0,kw_278},
    {"fitness_type",8,2,1,0,kw_279},
    {"replacement_type",8,4,2,0,kw_280}
}

KeyWord kw_282[5]  [static]
Initial value:
= {
    {0,0,1,0,0,kw_27},
    {0,0,9,0,0,kw_44},
    {0,0,3,0,0,kw_262},
    {"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_283[7] [static]
Initial value:
= {
  {"approx.method.name",3,0,1,1,0,0,0,0,3,N_mdm(str,subMethodName)},
  {"approx.method_pointer",3,0,1,1,0,0,0,0,3,N_mdm(str,subMethodPointer)},
  {"method.name",11,0,1,1,0,0,0,0,0,N_mdm(str,subMethodName)},
  {"method pointer",11,0,1,1,0,0,0,0,0,N_mdm(str,subMethodPointer)},
  {"replace_points",8,0,3,0,0,0,0,0,0,N_mdm(true,surrBasedGlobalReplacePts)}
}

KeyWord kw_284[2] [static]
Initial value:
= {
  {"filter",8,0,1,1,0,0,0,0,0,N_mdm(type,surrBasedLocalAcceptLogic_FILTER)},
  {"tr_ratio",8,0,1,1,0,0,0,0,0,N_mdm(type,surrBasedLocalAcceptLogic_TR_RATIO)}
}

KeyWord kw_285[7] [static]
Initial value:
= {
  {"augmented_lagrangian_objective",8,0,1,1,0,0,0,0,0,N_mdm(type,
         surrBasedLocalSubProbObj_AUGMENTED_LAGRANGIAN_OBJECTIVE)},
  {"lagrangian_objective",8,0,1,1,0,0,0,0,0,N_mdm(type,
         surrBasedLocalSubProbObj_LAGRANGIAN_OBJECTIVE)},
  {"linearized_constraints",8,0,2,2,0,0,0,0,0,N_mdm(type,
         surrBasedLocalSubProbCon_LINEARIZED_CONSTRAINTS)},
  {"no_constraints",8,0,2,2,0,0,0,0,0,N_mdm(type,
         surrBasedLocalSubProbCon_NO_CONSTRAINTS)},
  {"original_constraints",8,0,2,2,0,0,0,0,0,N_mdm(type,
         surrBasedLocalSubProbCon_ORIGINAL_CONSTRAINTS)},
  {"original_primary",8,0,1,1,0,0,0,0,0,N_mdm(type,
         surrBasedLocalSubProbObj_ORIGINAL_PRIMARY) },
  {"single_objective",8,0,1,1,0,0,0,0,0,N_mdm(type,
         surrBasedLocalSubProbObj_SINGLE_OBJECTIVE) }
}

KeyWord kw_286[1] [static]
Initial value:
= {
  {"homotopy",8,0,1,1,0,0,0,0,0,N_mdm(type,surrBasedLocalConstrRelax_HOMOTOPY)}
}

KeyWord kw_287[4] [static]
Initial value:
= {
  {"adaptive_penalty_merit",8,0,1,1,0,0,0,0,0,N_mdm(type,
         surrBasedLocalMeritFn_ADAPTIVE_PENALTY_MERIT)},
  {"augmented_lagrangian_merit",8,0,1,1,0,0,0,0,0,N_mdm(type,
         surrBasedLocalMeritFn_AUGMENTED_LAGRANGIAN_MERIT)},
  {"lagrangian_merit",8,0,1,1,0,0,0,0,0,N_mdm(type,surrBasedLocalMeritFn_LAGRANGIAN_MERIT)},
  {"penalty_merit",8,0,1,1,0,0,0,0,0,N_mdm(type,surrBasedLocalMeritFn_PENALTY_MERIT)}
}
KeyWord kw_288[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)},
  {"expansion_factor",10,0,6,0,0,0,0,0,N,mdm(Real,surrBasedLocalTRExpandTrigger)},
  {"expand_threshold",10,0,4,0,0,0,0,0,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_289[13] [static]
Initial value:
= {
  {"acceptance_logic",8,2,8,0,0,N,mdm(str,subMethodName)},
  {"approx_method_name","3,0,1,1,0,0,0,0,0,6,mdm(str,subMethodPointer)},
  {"approx_model_pointer","3,0,1,1,0,0,0,0,0,6,mdm(str,modelPointer)},
  {"approx_subproblem","8,7,6,0,0,N,mdm(str,subMethodScheduling_MASTER_SCHEDULING)},
  {"constraint_relax",8,1,9,0,0,0,N,mdm(str,subMethodScheduling_FINAL)},
  {"merit_function",8,4,7,0,0,N,mdm(str,subMethodScheduling_FINAL)},
  {"method_name","11,0,1,1,0,0,0,0,0,6,mdm(str,subMethodName)},
  {"method_pointer","11,0,1,1,0,0,0,0,0,6,mdm(str,subMethodPointer)},
  {"model_pointer","11,0,2,2,0,0,0,0,0,6,mdm(str,modelPointer)},
  {"soft_convergence_limit","9,0,3,0,0,0,0,0,0,0,mdm(ushint,softConvLimit)},
  {"trust_region","8,6,5,0,kw_284,0,0,0,0,0,0,mdm(true,surrBasedLocalLayerBypass)},
  {"truth_surrogate_bypass",8,4,0,0,0,0,0,0,0,mdm(true,surrBasedLocalLayerBypass)}
}

KeyWord kw_290[4] [static]
Initial value:
= {
  {0,0,1,0,0,0,kw_27},
  {"final_point",14,0,1,1,0,0,0,0,0,0,mdm(RealDL,finalPoint)},
  {"num_steps","9,0,2,2,0,0,0,0,0,0,mdm(int,numSteps)},
  {"step_vector","14,0,1,1,0,0,0,0,0,0,mdm(RealDL,stepVector)}
}

KeyWord kw_292[1] [static]
Initial value:
= {
  {"optional_interface_responses_pointer",11,0,1,0,0,0,0,0,0,0,mdm(type,optionalInterfRespPointer)}
}

KeyWord kw_293[2] [static]
Initial value:
= {
  {"master",8,0,1,1,0,0,0,0,0,0,mdm(type,subMethodScheduling_MASTER_SCHEDULING)},
  {"peer",8,0,1,1,0,0,0,0,0,0,mdm(type,subMethodScheduling_PEER_SCHEDULING)}
}
KeyWord kw_294[7] [static]
Initial value:

```c
{ "iterator_scheduling", 8, 2, 2, 0, kw_293 },
"iterator_servers", 8x19, 0, 1, 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", 8x19, 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_295[2] [static]
Initial value:

```c
{ "optional_interface_pointer", 11, 1, 1, 0, kw_292, 0, 0, 0, N, mom(str, interfacePointer),
"sub_method_pointer", 11, 7, 2, 1, kw_294, 0, 0, 0, N, mom(str, subMethodPointer) }
```

KeyWord kw_296[1] [static]
Initial value:

```c
{ "interface_pointer", 11, 0, 1, 0, 0, 0, 0, 0, N, mom(str, interfacePointer) }
```

KeyWord kw_297[3] [static]
Initial value:

```c
{ "eval_id", 8, 0, 2, 0, 0, 0, 0, 0, N, mom(augment_utype, approxChallengeFormat_TABULAR_EVAL_ID),
"header", 8, 0, 1, 0, 0, 0, 0, 0, N, mom(augment_utype, approxChallengeFormat_TABULAR_HEADER),
"interface_id", 8, 0, 0, 0, 0, 0, 0, 0, N, mom(augment_utype, approxChallengeFormat_TABULAR_IFACE_ID) }
```

KeyWord kw_298[4] [static]
Initial value:

```c
{ "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(utype, approxChallengeFormat_TABULAR_ANNOTATED),
"custom_annotated", 8, 3, 1, 0, kw_297, 0, 0, 0, 0, N, mom(utype, approxChallengeFormat_TABULAR_NONE) },
"freeform", 8, 0, 1, 0, 0, 0, 0, 0, N, mom(utype, approxChallengeFormat_TABULAR_NONE) }
```
KeyWord kw_299[6]  [static]
Initial value:

```cpp
= {
    "additive",8,0,2,0,0.,0.,0.,0,Nmom(type,approxCorrectionType_ADDITIVE_CORRECTION),
    "combined",8,0,2,0,0.,0.,0.,0,Nmom(type,approxCorrectionType_COMBINED_CORRECTION),
    "first_order",8,0,1,1,0,0.,0.,0,Nmom(order,approxCorrectionOrder_1),
    "multiplicative",8,0,2,2,0,0.,0.,0,Nmom(type,approxCorrectionType_MULTIPLICATIVE_CORRECTION),
    "second_order",8,0,1,1,0,0.,0.,0,Nmom(order,approxCorrectionOrder_2),
    "zeroth_order",8,0,1,1,0,0.,0.,0,Nmom(order,approxCorrectionOrder_0)
}
```

KeyWord kw_300[2]  [static]
Initial value:

```cpp
= {
    "folds",9,0,1,0,0.,0.,0.,0,Nmom(int,numFolds)},
    "percent",10,0,1,0,0.,0.,0.,0,Nmom(Real,percentFold)
}
```

KeyWord kw_301[2]  [static]
Initial value:

```cpp
= {
    "cross_validation",8,2,1,0,kw_300,0,0,0,Nmom(true,crossValidateFlag),
    "press",8,0,2,0,0.,0.,0,Nmom(true,pressFlag)
}
```

KeyWord kw_302[3]  [static]
Initial value:

```cpp
= {
    "eval_id",8,0,2,0,0.,0.,0.,0,Nmom(augment_utype,approxExportFormat_TABULAR_EVAL_ID)},
    "header",8,0,1,0,0.,0.,0.,0,Nmom(augment_utype,approxExportFormat_TABULAR_HEADER)},
    "interface_id",8,0,3,0,0.,0.,0.,0,Nmom(augment_utype,approxExportFormat_TABULAR_IFACE_ID)}
```

KeyWord kw_303[3]  [static]
Initial value:

```cpp
= {
    "annotated",8,0,1,0,0.,0.,0.,0,Nmom(utype,approxExportFormat_TABULAR_ANNOTATED)},{
    "custom_annotated",8,3,1,0,kw_302,0,0,0,Nmom(utype,approxExportFormat_TABULAR_NONE)},
    "freeform",8,0,1,0,0.,0.,0.,0,Nmom(utype,approxExportFormat_TABULAR_NONE)}
```

KeyWord kw_304[3]  [static]
Initial value:

```cpp
= {
    "constant",8,0,1,1,0,0.,0.,0,Nmom(lit,trendOrder_constant)},{
    "linear",8,0,1,1,0,0.,0.,0,Nmom(lit,trendOrder_linear)},{
    "reduced_quadratic",8,0,1,1,0,0.,0.,0,Nmom(lit,trendOrder_reduced_quadratic)}
```
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_305[2] [static]
Initial value:
= {
  {"point_selection",8,0,1,0,0,0,0,0,N_mom(true,pointSelection)},
  {"trend",8,3,2,0,kw_304}
}

KeyWord kw_306[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_307[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,0,4,0,0,0,0,0,0,N_mom(Real,krigingNugget)},
  {"optimization_method",11,0,2,0,0,0,0,0,N_mom(str,krigingOptMethod)},
  {"trend",8,4,1,0,kw_306}
}

KeyWord kw_308[2] [static]
Initial value:
= {
  {"dakota",8,2,1,1,kw_305,0,0,0,0,N_mom(lit,surrogateType_global_gaussian)},
  {"surfpack",8,7,1,1,kw_307,0,0,0,0,N_mom(lit,surrogateType_global_kriging)}
}

KeyWord kw_309[3] [static]
Initial value:
= {
  {"eval_id",8,0,2,0,0,0,0,0,N_mom(augment_utype,approxImportFormat_TABULAR_EVAL_ID)},
  {"header",8,0,1,0,0,0,0,0,N_mom(augment_utype,approxImportFormat_TABULAR_HEADER)}
}

KeyWord kw_310[4] [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(utype,approxImportFormat_TABULAR_ANNOTATED)},
  {"custom_annotated",8,3,1,0,kw_305,0,0,0,0,N_mom(utype,approxImportFormat_TABULAR_ANNOTATED)},
  {"freeform",8,0,1,0,0,0,0,0,N_mom(utype,approxImportFormat_TABULAR_NONE)}
}
KeyWord kw_311[2]  [static]
Initial value:
= {
    {"cubic",8,0,1,0,0.,0.,0,N_mom(lit,marsInterpolation_cubic)},
    {"linear",8,0,1,0,0.,0.,0,N_mom(lit,marsInterpolation_linear)}
}

KeyWord kw_312[3]  [static]
Initial value:
= {
    {"export_model_file",11,0,3,0,0.,0.,0.,0,N_mom(str, approxExportModelFile)},
    {"interpolation",8,0,2,0,kw_311},
    {"max_bases",9,0,1,0,0.,0.,0.,0,N_mom(shint,marsMaxBases)}
}

KeyWord kw_313[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_314[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.,0,N_mom(shint,annNodes)},
    {"random_weight",9,0,1,0,0.,0.,0.,0,N_mom(shint,annRandomWeight)},
    {"range",10,0,2,0,0.,0.,0.,0,N_mom(Real,annRange)}
}

KeyWord kw_315[2]  [static]
Initial value:
= {
    {"gradient_threshold",10,0,1,1,0,0.,0.,0,N_mom(Real,discontGradThresh)},
    {"jump_threshold",10,0,1,1,0,0.,0.,0,N_mom(Real,discontJumpThresh)}
}

KeyWord kw_316[3]  [static]
Initial value:
= {
    {"cell_type",11,0,1,0,0.,0.,0.,0,N_mom(str,decompCellType)},
    {"discontinuity_detection",8,2,3,0,kw_315,0.,0.,0,N_mom(true,decompDiscontDetect)},
    {"support_layers",9,0,2,0,0.,0.,0.,0,N_mom(int,decompSupportLayers)}
}
KeyWord kw_317[4] [static]
Initial value:

= {
    {'cubic', 8, 0, 1, 1, 0, 0, 0, 0, NMom(order, polynomialOrder, 3),
     'export_model_file', 11, 0, 2, 0, 0, 0, 0, NMom(str, approxExportModelFile),
     'quadratic', 8, 0, 1, 1, 0, 0, 0, 0, NMom(order, polynomialOrder, 2)}
}

KeyWord kw_318[5] [static]
Initial value:

= {
    {'bases', 9, 0, 1, 0, 0, 0, 0, 0, NMom(shint, rbfBases),
     'export_model_file', 11, 0, 5, 0, 0, 0, 0, NMom(str, approxExportModelFile),
     'max_pts', 9, 0, 2, 0, 0, 0, 0, NMom(shint, rbfMaxPts),
     'max_subsets', 9, 0, 4, 0, 0, 0, 0, NMom(shint, rbfMaxSubsets),
     'min_partition', 9, 0, 3, 0, 0, 0, 0, NMom(shint, rbfMinPartition)}
}

KeyWord kw_319[3] [static]
Initial value:

= {
    {'all', 8, 0, 1, 1, 0, 0, 0, 0, NMom(lit, approxPointReuse_all),
     'none', 8, 0, 1, 1, 0, 0, 0, 0, NMom(lit, approxPointReuse_none),
     'region', 8, 0, 1, 1, 0, 0, 0, 0, NMom(lit, approxPointReuse_region)}
}

KeyWord kw_320[23] [static]
Initial value:

= {
    {'actual_model_pointer', 11, 0, 4, 0, 0, 0, 0, 0, NMom(str, truthModelPointer),
     'challenge_points_file', 11, 4, 11, 0, kw_308, 0, 0, 0, NMom(str, approxChallengeFile),
     'correction', 8, 6, 9, 0, kw_329},
    {'dace_method_pointer', 11, 0, 4, 0, 0, 0, 0, 0, NMom(str, subMethodPointer),
     'diagnostics', 7, 4, 10, 0, kw_291, 0, 0, 0, 0, NMom(str, diagMetrics),
     'export_points_file', 11, 3, 7, 0, kw_303, 0, 0, 0, 0, NMom(str, approxExportFile),
     'gaussian_process', 8, 2, 1, 1, kw_308},
    {'import_points_file', 11, 4, 6, 0, kw_310, 0, 0, 0, NMom(str, approxImportFile),
     'kriging', 0, 2, 1, 1, kw_308, 0, 0, 0, 0, -3},
    {'mars', 8, 3, 1, 1, kw_312, 0, 0, 0, 0, NMom(lit, surrogateType_global_mars),
     'metrics', 15, 2, 10, 0, kw_301, 0, 0, 0, 0, NMom(str, diagMetrics),
     'minimum_points', 8, 6, 3, 0, 0, 0, 0, 0, NMom(type, pointsManagement_MINIMUM_POINTS),
     'moving_least_squares', 8, 3, 1, 1, kw_313, 0, 0, 0, 0, NMom(lit, surrogateType_global_moving_least_squares),
     'neural_network', 8, 5, 1, 1, kw_314, 0, 0, 0, 0, NMom(lit, surrogateType_global_neural_network),
     'piecewise_decomposition', 8, 3, 2, 0, kw_316, 0, 0, 0, NMom(true, piecewiseDecomp),
     'polynomial', 8, 4, 1, 1, kw_317, 0, 0, 0, NMom(lit, surrogateType_global_polynomial),
     'radial_basis', 8, 5, 1, 1, kw_318, 0, 0, 0, 0, NMom(lit, surrogateType_global_radial_basis),
     'recommended_points', 8, 0, 3, 0, 0, 0, 0, 0, NMom(type, pointsManagement_RECOMMENDED_POINTS),
     'reuse_points', 8, 3, 5, 0, kw_219},
    {'samples_file', 3, 4, 6, 0, kw_310, 0, 0, 0, 0, -13, NMom(str, approxImportFile),
     'total_points', 9, 0, 3, 0, 0, 0, 0, 0, NMom(int, pointsTotal),
     'use_derivatives', 8, 6, 0, 5, 0, 0, 0, 0, 0, NMom(true, modelUseDerivsFlag)}
}
KeyWord kw_321[6] [static]
Initial value:
= {
   "additive", 8, 0, 2, 0, 0, 0, 0, N\_mom(type, approx\_CorrectionType\_ADDITIVE\_CORRECTION) },
   "combined", 8, 0, 2, 0, 0, 0, 0, N\_mom(type, approx\_CorrectionType\_COMBINED\_CORRECTION) },
   "first\_order", 8, 0, 1, 0, 0, 0, 0, N\_mom(order, approx\_CorrectionOrder, 1) },
   "multiplicative", 8, 0, 2, 0, 0, 0, 0, N\_mom(type, approx\_CorrectionType\_MULTIPlicative\_CORRECTION) },
   "second\_order", 8, 0, 1, 0, 0, 0, 0, N\_mom(order, approx\_CorrectionOrder, 2) },
   "zeroth\_order", 8, 0, 1, 0, 0, 0, 0, N\_mom(order, approx\_CorrectionOrder, 0) }

KeyWord kw_322[3] [static]
Initial value:
= {
   "correction", 8, 6, 3, 3, kw_321},
   "high\_fidelity\_model\_pointer", 11, 0, 2, 0, 0, 0, 0, N\_mom(str, truthModel\_Pointer) },
   "low\_fidelity\_model\_pointer", 11, 0, 1, 0, 0, 0, 0, N\_mom(str, lowFidelityModel\_Pointer) }

KeyWord kw_323[1] [static]
Initial value:
= {
   "actual\_model\_pointer", 11, 0, 1, 0, 0, 0, 0, N\_mom(str, truthModel\_Pointer) }

KeyWord kw_324[2] [static]
Initial value:
= {
   0, 0, 1, 0, 0, kw_323},
   "taylor\_series", 8, 0, 1, 1 }

KeyWord kw_325[2] [static]
Initial value:
= {
   0, 0, 1, 0, 0, kw_323},
   "tana", 8, 0, 1, 1 }

KeyWord kw_326[5] [static]
Initial value:
= {
   "global", 8, 23, 2, 1, kw_32D},
   "hierarchical", 8, 3, 2, 1, kw_322, 0, 0, 0, 0, N\_mom(lit, surrogate\_Type\_hierarchical) },
   "id\_surrogates", 13, 0, 1, 0, 0, 0, 0, 0, N\_mom(int\_set\_m1, surrogate\_Fn\_Indices) },
   "local\_*, 8, 1, 2, 1, kw_324, 0, 0, 0, N\_mom(lit, surrogate\_Type\_local\_taylor) },
   "multipoint\_*, 8, 1, 2, 1, kw_325, 0, 0, 0, 0, N\_mom(lit, surrogate\_Type\_Multipoint\_tana) }
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_327[7] [static]
Initial value:

```c
= {
    {"hierarchical_tagging", 8, 0, 4, 0, 0, 0, 0, Nmom(true, hierarchicalTags),
     "id_model", 11, 0, 1, 0, 0, 0, 0, 0, Nmom(str, idModel),
     "nested", 8, 2, 5, 1, kw_325, 0, 0, 0, Nmom(lit, modelType_nested),
     "responses_pointer", 11, 0, 3, 0, 0, 0, 0, 0, Nmom(str, responsesPointer),
     "single", 8, 1, 5, 1, kw_326, 0, 0, 0, Nmom(lit, modelType_single),
     "surrogate", 8, 5, 5, 1, kw_326, 0, 0, 0, Nmom(lit, modelType_surrogate),
     "variables_pointer", 11, 0, 2, 0, 0, 0, 0, 0, Nmom(str, variablesPointer) }
```}

KeyWord kw_328[2] [static]
Initial value:

```c
= {
    {"exp_id", 8, 0, 2, 0, 0, 0, 0, 0, Nrem(augment_utype, scalarDataFormat_TABULAR_EVAL_ID) },
    {"header", 8, 0, 1, 0, 0, 0, 0, 0, Nrem(augment_utype, scalarDataFormat_TABULAR_HEADER) } }
```}

KeyWord kw_329[3] [static]
Initial value:

```c
= {
    {"annotated", 8, 0, 1, 0, 0, 0, 0, 0, Nrem(utype, scalarDataFormat_TABULAR_EXPERANNOT) },
    {"custom_annotated", 8, 2, 1, 0, kw_328, 0, 0, 0, Nrem(utype, scalarDataFormat_TABULARNONE) },
    {"freeform", 8, 0, 1, 0, 0, 0, 0, 0, Nrem(utype, scalarDataFormat_TABULARNONE) } }
```}

KeyWord kw_330[6] [static]
Initial value:

```c
= {
    {"interpolate", 8, 0, 5, 0, 0, 0, 0, 0, Nrem(true, interpolateFlag) },
    {"num_config_variables", 0x29, 0, 2, 0, 0, 0, 0, 0, Nrem(size_t, numExpConfigVars) },
    {"num_experiments", 0x29, 0, 1, 0, 0, 0, 0, 0, Nrem(size_t, numExperiments) },
    {"read_field_coordinates", 8, 0, 6, 0, 0, 0, 0, 0, Nrem(true, readFieldCoords) },
    {"scalar_data_file", 11, 3, 4, 0, kw_329, 0, 0, 0, Nrem(str, scalarDataFileName) },
    {"variance_type", 0x80f, 0, 3, 0, 0, 0, 0, 0, Nrem(strL, varianceType) } }
```}

KeyWord kw_331[2] [static]
Initial value:

```c
= {
    {"exp_id", 8, 0, 2, 0, 0, 0, 0, 0, Nrem(augment_utype, scalarDataFormat_TABULAR_EVAL_ID) },
    {"header", 8, 0, 1, 0, 0, 0, 0, 0, Nrem(augment_utype, scalarDataFormat_TABULAR_HEADER) } }
```}
KeyWord kw_332[6] [static]
Initial value:

= {
  {"annotated", 8, 0, 1, 0, 0, 0, 0, N, rem(utype, scalarDataFormat_TABULAR_EXPER_ANNOT)),
  {"custom_annotated", 8, 0, 1, 0, kw_331, 0, 0, 0, N, rem(utype, scalarDataFormat_TABULAR_NONE)),
  {"freeform", 8, 0, 0, 0, 0, 0, 0, 0, N, rem(utype, scalarDataFormat_TABULAR_NONE)),
  {"num_config_variables", 0x29, 0, 2, 0, 0, 0, 0, N, rem(sizet, numExpConfigVars)),
  {"num_experiments", 0x29, 0, 2, 0, 0, 0, 0, N, rem(sizet, numExperiments)),
  {"variance_type", 0x80f, 0, 4, 0, 0, 0, 0, 0, N, rem(strL, varianceType))
}

KeyWord kw_333[4] [static]
Initial value:

= {
  {"coordinate_data_file", 11, 0, 3, 0, 0, 0, 0, 0, 0, rem(str, coordDataFileName)),
  {"coordinate_list", 14, 0, 3, 0, 0, 0, 0, 0, N, rem(RealL, coordList)),
  {"lengths", 13, 0, 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_334[6] [static]
Initial value:

= {
  {"nonlinear_inequality_scale_types", 0x807, 0, 2, 0, 0, 0, 0, 0, 3, N, rem(strL, nonlinearEqScaleTypes)),
  {"nonlinear_inequality_scales", 0x806, 0, 3, 0, 0, 0, 0, 0, 3, N, rem(RealDL, nonlinearEqScales)),
  {"nonlinear_inequality_targets", 0x41, 0, 0, 0, 0, 0, 0, 0, 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, 0, N, rem(RealDL, nonlinearEqScales)),
  {"targets", 14, 0, 1, 0, 0, 0, 0, 0, 0, N, rem(RealDL, nonlinearEqTargets))
}

KeyWord kw_335[8] [static]
Initial value:

= {
  {"lower_bounds", 14, 0, 1, 0, 0, 0, 0, 0, N, rem(RealDL, nonlinearIneqLowerBounds)),
  {"nonlinear_inequality_lower_bounds", 6, 0, 1, 0, 0, 0, 0, 0, N, rem(RealDL, nonlinearIneqLowerBounds))
},
  {"nonlinear_inequality_scale_types", 0x807, 0, 3, 0, 0, 0, 0, 0, 3, N, rem(strL, nonlinearIneqScaleTypes)),
  {"nonlinear_inequality_scales", 0x806, 0, 4, 0, 0, 0, 0, 0, 3, N, rem(RealDL, nonlinearIneqScales)),
  {"nonlinear_inequality_upper_bounds", 6, 0, 2, 0, 0, 0, 0, 0, 3, N, rem(RealDL, nonlinearIneqUpperBounds))
},
  {"scale_types", 0x80f, 0, 3, 0, 0, 0, 0, 0, 0, N, rem(strL, nonlinearIneqScaleTypes)),
  {"scales", 0x80e, 0, 4, 0, 0, 0, 0, 0, 0, N, rem(RealDL, nonlinearIneqScales)),
  {"upper_bounds", 14, 0, 2, 0, 0, 0, 0, 0, 0, N, rem(RealDL, nonlinearIneqUpperBounds))
}
KeyWord kw_336[18]  [static]
Initial value:
= {
   {"calibration_data", 8, 6, 6, 0, kw_330, 0, 0, 0, 0, N_rem(true, calibrationDataFlag)},
   {"calibration_data_filename", 11, 6, 6, 0, kw_332, 0, 0, 0, 0, N_rem(true, scalarDataFileName)},
   {"field_calibration_term_scale_types", 0x807, 0, 3, 0, 0, 0, 0, 12, N_rem(true, primaryRespFnScaleTypes)},
   {"least_squares_term_scale_types", 0x807, 0, 3, 0, 0, 0, 0, 7, N_rem(true, primaryRespFnScaleTypes)},
   {"primary_scale_types", 0x80f, 0, 3, 0, 0, 0, 0, 0, N_rem(true, primaryRespFnScaleTypes)},
   {"primary_scales", 0x80e, 0, 4, 0, 0, 0, 0, 0, N_rem(true, primaryRespFnScaleTypes)},
   {"scalar_calibration_terms", 0x29, 0, 1, 0, 0, 0, 0, 0, N_rem(true, scalarLeastSqTerms)},
   {"least_squares_term_scales", 0x806, 0, 4, 0, 0, 0, 0, 7, N_rem(true, primaryRespFnScaleTypes)},
   {"least_squares_term_scale_types", 0x807, 0, 3, 0, 0, 0, 0, 7, N_rem(true, primaryRespFnScaleTypes)},
   {"least_squares_term_scales", 0x806, 0, 4, 0, 0, 0, 0, 7, N_rem(true, primaryRespFnScaleTypes)},
   {"least_squares_weights", 6, 0, 5, 0, 0, 0, 0, 8, N_rem(true, primaryRespFnWeights)},
   {"scalar_calibration_weights", 6, 0, 5, 0, 0, 0, 0, 8, N_rem(true, primaryRespFnWeights)},
   {"nonlinear_equation_constraints", 0x29, 6, 8, 0, kw_334, 0, 0, 0, N_rem(true, numNonlinearEqConstraints)},
   {"nonlinear_inequality_constraints", 0x29, 8, 7, 0, kw_335, 0, 0, 0, N_rem(true, numNonlinearIneqConstraints)},
   {"primary_scale_types", 0x80f, 0, 3, 0, 0, 0, 0, 0, N_rem(true, primaryRespFnScaleTypes)},
   {"primary_scales", 0x80e, 0, 4, 0, 0, 0, 0, 0, N_rem(true, primaryRespFnScaleTypes)},
   {"weights", 14, 0, 5, 0, 0, 0, 0, 0, N_rem(true, primaryRespFnWeights)}
}

KeyWord kw_337[4]  [static]
Initial value:
= {
   {"absolute", 8, 0, 2, 0, 0, 0, 0, 0, N_rem(true, fdGradStepType_absolute)},
   {"bounds", 8, 0, 2, 0, 0, 0, 0, 0, N_rem(true, 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(true, fdGradStepType_relative)}
}

KeyWord kw_338[8]  [static]
Initial value:
= {
   {"central", 8, 0, 4, 0, 0, 0, 0, 0, N_rem(true, intervalType_central)},
   {"dakota", 8, 4, 2, 0, kw_337, 0, 0, 0, N_rem(true, methodSource_dakota)},
   {"fd_gradient_step_size", 6, 0, 5, 0, 0, 0, 0, 1, N_rem(true, fdGradStepSize)},
   {"fd_step_size", 14, 0, 5, 0, 0, 0, 0, 0, N_rem(true, fdGradStepSize)},
   {"forward", 8, 0, 4, 0, 0, 0, 0, 0, N_rem(true, intervalType_forward)},
   {"interval_type", 8, 0, 3},
   {"method_source", 8, 0, 1},
   {"vendor", 8, 0, 2, 0, 0, 0, 0, 0, N_rem(true, methodSource_vendor)}
}

KeyWord kw_339[3]  [static]
Initial value:
= {
   {0, 0, 8, 0, 0, kw_338},
   {"id_analytic_gradients", 13, 0, 2, 2, 0, 0, 0, 0, N_rem(true, idAnalyticGrads)},
   {"id_numerical_gradients", 13, 0, 1, 1, 0, 0, 0, 0, N_rem(true, idNumericalGrads)}
}
CHAPTER 12. NAMESPACE DOCUMENTATION

KeyWord kw_340[2] [static]
Initial value:
= {
    {"fd_hessian_step_size",6,0,1,0,0...1,N_rem(RealL,fdHessStepSize)},
    {"fd_step_size",14,0,1,0,0...0,N_rem(RealL,fdHessStepSize)}
}

KeyWord kw_341[1] [static]
Initial value:
= {
    {"damped",8,0,1,0,0...0,N_rem(lit,quasiHessianType_damped_bfgs)}
}

KeyWord kw_342[2] [static]
Initial value:
= {
    {"bfgs",8,1,1,1,kw_341,0...0,N_rem(lit,quasiHessianType_bfgs)},
    {"sr1",8,0,1,0,0...0,N_rem(lit,quasiHessianType_sr1)}
}

KeyWord kw_343[8] [static]
Initial value:
= {
    {"absolute",8,0,2,0,0...0,N_rem(lit,fdHessStepType_absolute)},
    {"bounds",8,0,2,0,0...0,N_rem(lit,fdHessStepType_bounds)},
    {"central",8,0,3,0,0...0,N_rem(true,centralHess)},
    {"forward",8,0,1,0,0...0,N_rem(false,centralHess)},
    {"id_analytic_hessians",13,0,5,0,0...0,N_rem(intset,idAnalyticHessians)},
    {"id_numerical_hessians",13,2,4,0,kw_342,0...0,N_rem(intset,idNumericalHessians)},
    {"id_quasi_hessians",13,3,4,0,kw_342,0...0,N_rem(intset,idQuasiHessians)},
    {"relative",8,0,3,0,0...0,N_rem(lit,fdHessStepType_relative)}
}

KeyWord kw_344[4] [static]
Initial value:
= {
    {"coordinate_data_file",11,0,3,0,0...0,N_rem(str,coordDataFileName)},
    {"coordinate_list",14,0,3,0,0...0,N_rem(RealL,coordList)},
    {"lengths",13,0,1,0,0...0,N_rem(ivec,fieldLengths)},
    {"num_coordinates_per_field",13,0,2,0,0...0,N_rem(ivec,numCoordsPerField)}
}

KeyWord kw_345[6] [static]
Initial value:
= {
    {"nonlinear_equality_scale_types",0x807,0,2,0,0...0,N_rem(strL,nonlinearEqScaleTypes)},
    {"nonlinear_equality_scales",0x806,0,3,0,0...0,N_rem(RealDL,nonlinearEqScales)},
    {"nonlinear_equality_targets",6,0,1,0,0...0,N_rem(RealDL,nonlinearEqTargets)},
    {"scale_types",0x805,0,2,0,0...0,N_rem(strL,nonlinearEqScaleTypes)},
    {"scales",0x804,0,3,0,0...0,N_rem(RealDL,nonlinearEqScales)},
    {"targets",14,0,1,0,0...0,N_rem(RealDL,nonlinearEqTargets)}
}
KeyWord kw_346[8]  [static]
Initial value:
= {
  {"lower_bounds",14,0,1,0,0,0,0.,0.,0.,N_rem(RealDL,nonlinearIneqLowerBnds)},
  {nonlinear_inequality_lower_bounds*,6,0,1,0,0,0,0.,0.,0.,N_rem(RealDL,nonlinearIneqLowerBnds)}},
  {nonlinear_inequality_scale_types*,0x807,0,0,0,0,0,0.,0.,3,N_rem(strL,nonlinearIneqScaleTypes)},
  {nonlinear_inequality_scales*,0x806,0,0,0,0,0,0.,0.,3,N_rem(RealDL,nonlinearIneqScales)},
  {nonlinear_inequality_upper_bounds*,6,0,2,0,0,0,0.,0.,3,N_rem(RealDL,nonlinearIneqUpperBnds)}
},
  {"scale_types",0x80f,0,0,0,0,0,0.,0.,0,N_rem(strL,nonlinearIneqScaleTypes)},
  {"scales",0x80e,0,4,0,0,0,0,0.,0,N_rem(RealDL,nonlinearIneqScales)},
  {"upper_bounds",14,0,2,0,0,0,0.,0.,0,N_rem(RealDL,nonlinearIneqUpperBnds)}
}

KeyWord kw_347[15]  [static]
Initial value:
= {
  {field_objectives",0x29,4,8,0,kw_344,0,0,0,0.,0.,N_rem(sizet,numFieldObjectiveFunctions)},
  {multi_objective_weights",6,0,4,0,0,0,0,0,0,13,N_rem(RealDL,primaryRespFnWeights)},
  {nonlinear_equality_constraints",0x29,6,6,0,kw_345,0,0,0,0,0.,N_rem(sizet,numNonlinearEqConstraints)},
  {nonlinear_inequality_constraints",0x29,8,5,0,kw_346,0,0,0,0,0.,N_rem(sizet,numNonlinearIneqConstraints)},
  {num_field_objectives",0x21,4,6,0,kw_344,0,0,4,0,0,0,N_rem(sizet,numFieldObjectiveFunctions)},
  {num_nonlinear_equality_constraints",0x21,6,6,0,kw_345,0,0,4,0,0,0,N_rem(sizet,numNonlinearEqConstraints)},
  {num_nonlinear_inequality_constraints",0x21,8,5,0,kw_346,0,0,4,0,0,0,N_rem(sizet,numNonlinearIneqConstraints)},
  {objective_function_scale_types",0x807,0,2,0,0,0,0,0,2,N_rem(strL,primaryRespFnScaleTypes)},
  {objective_function_scales",0x806,0,3,0,0,0,0,0,2,N_rem(RealDL,primaryRespFnScales)},
  {primary_scales",0x80e,0,3,0,0,0,0,0,0,N_rem(RealDL,primaryRespFnScaleTypes)},
  {scalar_objectives",0x29,6,5,0,0,0,0,0,0,N_rem(sizet,numScalarObjectiveFunctions)},
  {sense",0x80f,0,1,0,0,0,0,0,0,N_rem(strL,primaryRespFnSense)},
  {weights",14,0,4,0,0,0,0,0,0,N_rem(RealDL,primaryRespFnWeights)}
}

KeyWord kw_348[4]  [static]
Initial value:
= {
  {coordinate_data_file",11,0,3,0,0,0,0,0,0,N_rem(str,coordDataFileName)},
  {coordinate_list",14,0,3,0,0,0,0,0,0,N_rem(RealL,coordList)},
  {lengths",13,0,1,1,0,0,0,0,0,N_rem(ivec,fieldLengths)}
}

KeyWord kw_349[4]  [static]
Initial value:
= {
  {field_responses",0x29,4,2,0,kw_348,0,0,0,0,0,N_rem(sizet,numFieldResponseFunctions)},
  {num_field_responses",0x21,4,2,0,kw_348,0,0,0,0,0,N_rem(sizet,numFieldResponseFunctions)},
  {scalar_responses",0x21,6,6,0,kw_349,0,0,0,0,0,N_rem(sizet,numScalarResponseFunctions)},
  {scale_responses",0x29,6,5,0,0,0,0,0,0,N_rem(sizet,numScalarResponseFunctions)}
}
CHAPTER 12. NAMESPACE DOCUMENTATION

KeyWord kw\_350[7]  [static]

Initial value:

```c
= {
    {"absolute",8,0,2,0,0,0,0,0,N\_rem(lit,fd\_Hess\_Step\_Type\_absolute)},
    {"bounds",8,0,2,0,0,0,0,0,N\_rem(lit,fd\_Hess\_Step\_Type\_bounds)},
    {"central",8,0,3,0,0,0,0,0,N\_rem(true,central\_Hess)},
    {"fd\_hessian\_step\_size",6,0,1,0,0,0,0,1,N\_rem(\_RealL,fd\_Hess\_Step\_Size)},
    {"fd\_step\_size",14,0,1,0,0,0,0,0,N\_rem(\_RealL,fd\_Hess\_Step\_Size)},
    {"forward",8,0,3,0,0,0,0,0,N\_rem(false,central\_Hess)},
    {"relative",8,0,2,0,0,0,0,0,N\_rem(lit,fd\_Hess\_Step\_Type\_relative)}
}
```

KeyWord kw\_351[1]  [static]

Initial value:

```c
= {
    {"damped",8,0,1,0,0,0,0,0,N\_rem(lit,quasi\_Hessian\_Type\_damped\_bfgs)}
}
```

KeyWord kw\_352[2]  [static]

Initial value:

```c
= {
    {"bfgs",8,1,1,1,\_kw\_351,0,0,0,N\_rem(lit,quasi\_Hessian\_Type\_bfgs)},
    {"sr1",8,0,1,1,0,0,0,0,N\_rem(lit,quasi\_Hessian\_Type\_sr1)}
}
```

KeyWord kw\_353[19]  [static]

Initial value:

```c
= {
    {"analytic\_gradients",8,0,4,2,0,0,0,0,0,N\_rem(lit,gradient\_Type\_analytic)}},
    {"analytic\_hessians",8,0,5,3,0,0,0,0,0,N\_rem(lit,hessian\_Type\_analytic)}},
    {"calibration\_terms",0x29,18,3,1,\_kw\_336,0,0,0,N\_rem(sizet,num\_Least\_Sq\_Terms)}},
    {"descriptors",15,0,2,0,0,0,0,0,N\_rem(strL,response\_Labels)}},
    {"id\_responses",11,0,1,0,0,0,0,0,N\_rem(str,\_id\_Responses)}},
    {"least\_squares\_terms",0x21,18,3,1,\_kw\_336,0,0,0,0,N\_rem(sizet,num\_Least\_Sq\_Terms)}},
    {"mixed\_gradients",8,1,4,2,\_kw\_339,0,0,0,0,N\_rem(lit,gradient\_Type\_mixed)}},
    {"mixed\_hessians",8,8,5,3,\_kw\_343,0,0,0,0,N\_rem(lit,hessian\_Type\_mixed)}},
    {"no\_gradients",8,0,4,2,0,0,0,0,0,N\_rem(lit,gradient\_Type\_none)}},
    {"no\_hessians",8,0,5,3,0,0,0,0,0,N\_rem(lit,hessian\_Type\_none)}},
    {"num\_least\_squares\_terms",0x21,18,3,1,\_kw\_336,0,0,0,0,N\_rem(sizet,num\_Least\_Sq\_Terms)}},
    {"num\_objective\_functions",0x21,15,3,1,\_kw\_347,0,0,0,0,N\_rem(sizet,num\_Objective\_Functions)}},
    {"num\_response\_functions",0x21,4,3,1,\_kw\_349,0,0,0,0,N\_rem(sizet,num\_Response\_Functions)}},
    {"numerical\_gradients",8,8,4,2,\_kw\_338,0,0,0,0,N\_rem(lit,gradient\_Type\_numerical)}},
    {"numerical\_hessians",8,8,5,3,\_kw\_350,0,0,0,0,N\_rem(lit,hessian\_Type\_numerical)}},
    {"objective\_functions",0x29,15,3,1,\_kw\_347,0,0,0,0,N\_rem(sizet,num\_Objective\_Functions)}},
    {"quasi\_hessians",8,2,5,3,\_kw\_352,0,0,0,0,N\_rem(lit,hessian\_Type\_quasi)}},
    {"response\_descriptors",7,0,2,0,0,0,0,14,0,N\_rem(strL,response\_Labels)}},
    {"response\_functions",0x29,4,3,1,\_kw\_349,0,0,0,0,N\_rem(sizet,num\_Response\_Functions)}
```
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_354[6]  [static]

Initial value:

= {
  {"aleatory",8,0,1,1,0,0.,0.,0.,0.,N_vam(type,varsView,ALEATORY,UNCERTAIN,VIEW)},
  {"all",8,0,1,1,0,0.,0.,0.,0.,N_vam(type,varsView,ALL,VIEW)},
  {"design",8,0,1,1,0,0.,0.,0.,0.,N_vam(type,varsView,DESIGN,VIEW)},
  {"epistemic",8,0,1,1,0,0.,0.,0.,0.,N_vam(type,varsView,EPISTEMIC,UNCERTAIN,VIEW)},
  {"state",8,0,1,1,0,0.,0.,0.,0.,N_vam(type,varsView,STATE,VIEW)},
  {"uncertain",8,0,1,1,0,0.,0.,0.,0.,N_vam(type,varsView,UNCERTAIN,VIEW)}
}

KeyWord kw_355[11]  [static]

Initial value:

= {
  {"alphas",14,0,1,1,0,0.,0.,0.,0.,N_vam(RealLb,betaUncAlphas)},
  {"betas",14,0,2,2,0,0.,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.,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.,0.,N_vam(rvec,betaUncLowerBnds)},
  {"buv_upper_bounds",6,0,4,4,0,0.,0.,0.,N_vam(rvec,betaUncUpperBnds)},
  {"descriptors",15,0,0,0,0,0.,0.,0.,N_vae(caulbl,CAUVar_beta)},
  {"initial_point",14,0,5,0,0,0,0,0,0,N_vam(rvec,betaUncVars)},
  {"lower_bounds",14,0,3,3,0,0,0,0,0,N_vam(rvec,betaUncLowerBnds)},
  {"upper_bounds",14,0,4,4,0,0,0,0,0,N_vam(rvec,betaUncUpperBnds)}
}

KeyWord kw_356[5]  [static]

Initial value:

= {
  {"descriptors",15,0,4,0,0,0,0,0,N_vae(desilbl,DAUIVar_binomial)},
  {"initial_point",13,0,3,3,0,0,0,0,0,N_vam(IntLb,binomialUncVars)},
  {"num_trials",13,0,2,2,0,0,0,0,0,N_vam(IntLb,binomialUncNumTrials)},
  {"prob_per_trial",6,0,1,1,0,0,0,0,1,N_vam(rvec,binomialUncProbPerTrial)},
  {"probability_per_trial",14,0,1,1,0,0,0,0,0,N_vam(rvec,binomialUncProbPerTrial)}
}

KeyWord kw_357[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",0x806,0,5,0,0,0,0,0,6,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",14,0,1,0,0,0,0,0,0,N_vam(rvec,continuousDesignVars)},
  {"lower_bounds",14,0,2,0,0,0,0,0,0,N_vam(rvec,continuousDesignLowerBnds)},
  {"scale_types",0x80f,0,4,0,0,0,0,0,0,N_vam(strL,continuousDesignScaleTypes)},
  {"scales",0x80e,0,5,0,0,0,0,0,0,N_vam(rvec,continuousDesignScales)},
  {"upper_bounds",14,0,3,0,0,0,0,0,0,N_vam(rvec,continuousDesignUpperBnds)}
}
**KeyWord kw_358[10] [static]**

Initial value:

```c
= {
    {"descriptors",15,0,6,0,0,0..0,0,N_vae(ceuibl,CEUVar_interval)},
    {"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_vam(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(newarray,Var_Info_CI)},
    {"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(newarray,Var_Info_CI)},
    {"upper_bounds",14,0,4,2,0,0,0,0,N_vam(newrvec,Var_Info_CIub)}
}
```

**KeyWord kw_359[8] [static]**

Initial value:

```c
= {
    {"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..0,4,N_vam(strL,continuousStateLabels)},
    {"initial_point",13,0,1,0,0,0..0,0,N_vam(rvec,continuousStateVars)},
    {"lower_bounds",13,0,2,0,0,0..0,0,N_vam(rvec,continuousStateLowerBnds)},
    {"upper_bounds",14,0,3,0,0,0..0,0,N_vam(rvec,continuousStateUpperBnds)}
}
```

**KeyWord kw_360[8] [static]**

Initial value:

```c
= {
    {"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..0,4,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_361[1] [static]**

Initial value:

```c
= {
    {"adjacency_matrix",13,0,1,0,0,0..0,0,N_vam(newivec,Var_Info吊顶)}
}
```
KeyWord kw_362[7]  [static]
Initial value:

= {
{"categorical",15,1,3,0,kw_361,0,0,0,0,N_vam(categorical,discreteDesignSetIntCat)},
{"descriptors",15,0,5,0,0,0,0,0,N_vam(strl,discreteDesignSetIntLabels)},
{"elements",13,0,2,1,0,0,0,0,0,N_vam(newivec,Var_Info_ddsi)},
{"elements_per_variable",0x80d,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_ddssi)}},
{"initial_point",13,0,4,0,0,0,0,0,0,N_vam(ivec,discreteDesignSetIntVars)},
{"num_set_values",2x805,0,1,0,0,0,0,0,0,0,0,N_vam(newiarray,Var_Info_ddssi)}},
{"set_values",5,0,2,1,0,0,0,0,0,0,0,0,N_vam(newivec,Var_Info_ddsi)}

KeyWord kw_363[1]  [static]
Initial value:

= {
{"adjacency_matrix",13,0,1,0,0,0,0,0,0,N_vam(newivec,Var_Info_ddsra)}
}

KeyWord kw_364[7]  [static]
Initial value:

= {
{"categorical",15,1,3,0,kw_363,0,0,0,0,N_vam(categorical,discreteDesignSetRealCat)},
{"descriptors",15,0,5,0,0,0,0,0,N_vam(strl,discreteDesignSetRealLabels)},
{"elements",14,0,2,1,0,0,0,0,0,N_vam(newrvec,Var_Info_ddsr)},
{"elements_per_variable",0x80d,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_ddds)},
{"initial_point",14,0,4,0,0,0,0,0,0,N_vam(rvec,discreteDesignSetRealVars)},
{"num_set_values",2x805,0,1,0,0,0,0,0,0,0,0,N_vam(newiarray,Var_Info_ddds)}},
{"set_values",6,0,2,1,0,0,0,0,0,0,0,0,N_vam(newrvec,Var_Info_ddds)}

KeyWord kw_365[7]  [static]
Initial value:

= {
{"adjacency_matrix",13,0,3,0,0,0,0,0,0,N_vam(newivec,Var_Info_ddds)}},
{"descriptors",15,0,5,0,0,0,0,0,N_vam(strl,discreteDesignSetStrLabels)},
{"elements",15,0,2,1,0,0,0,0,0,N_vam(newrvec,Var_Info_ddss)},
{"elements_per_variable",0x80d,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_ddds)},
{"initial_point",15,0,4,0,0,0,0,0,0,N_vam(rvec,discreteDesignSetStrVars)},
{"num_set_values",8x805,0,1,0,0,0,0,0,0,0,0,N_vam(newiarray,Var_Info_ddds)}},
{"set_values",7,0,2,1,0,0,0,0,0,0,0,0,N_vam(newrvec,Var_Info_ddds)}

KeyWord kw_366[3]  [static]
Initial value:

= {
{"integer",0x19,7,1,0,kw_362,0,0,0,0,N_vam(pintz,numDiscreteDesSetIntVars)},
{"real",0x19,7,3,0,kw_364,0,0,0,0,N_vam(pintz,numDiscreteDesSetRealVars)},
{"string",0x19,7,2,0,kw_365,0,0,0,0,N_vam(pintz,numDiscreteDesSetStrVars)}
}
CHAPTER 12. NAMESPACE DOCUMENTATION

KeyWord kw_367[9] [static]
Initial value:

= {
    {"descriptors",15,0,6,0,0,0,0,0,0,N_vam(deuilbl,DEUIVar_interval)},
    {"initial_point",13,0,5,0,0,0,0,0,0,N_vam(ivec,discreteIntervalUncVars)},
    {"interval_probabilities",14,0,2,0,0,0,0,0,0,N_vam(newrvec,Var_Info_DIp)},
    {"interval_probs",6,0,2,0,0,0,0,0,-1,N_vam(newrvec,Var_Info_DIp)},
    {"lower_bounds",13,0,1,0,0,0,0,0,0,N_vam(newivec,Var_Info_DIlb)},
    {"num_intervals",13,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_nDI)},
    {"range_probabilities",6,0,2,0,0,0,0,0,-4,N_vam(newrvec,Var_Info_DIp)},
    {"range_probs",6,0,2,0,0,0,0,0,-5,N_vam(newrvec,Var_Info_DIp)},
    {"upper_bounds",13,0,4,2,0,0,0,0,0,N_vam(newivec,Var_Info_DIub)}
}

KeyWord kw_368[8] [static]
Initial value:

= {
    {"descriptors",15,0,4,0,0,0,0,0,0,N_vam(strL,discreteStateRangeLabels)},
    {"dsv_descriptors",7,0,4,0,0,0,0,0,-1,N_vam(strL,discreteStateRangeLabels)},
    {"dsv_initial_state",5,0,1,0,0,0,0,0,3,N_vam(ivec,discreteStateRangeVars)},
    {"dsv_lower_bounds",5,0,2,0,0,0,0,0,3,N_vam(ivec,discreteStateRangeLowerBnds)},
    {"dsv_upper_bounds",5,0,3,0,0,0,0,0,3,N_vam(ivec,discreteStateRangeUpperBnds)},
    {"initial_state",13,0,4,2,0,0,0,0,0,N_vam(ivec,discreteStateRangeVars)},
    {"lower_bounds",13,0,2,0,0,0,0,0,0,N_vam(ivec,discreteStateRangeLowerBnds)},
    {"upper_bounds",13,0,3,0,0,0,0,0,0,N_vam(ivec,discreteStateRangeUpperBnds)}
}

KeyWord kw_369[7] [static]
Initial value:

= {
    {"categorical",15,0,3,0,0,0,0,0,0,N_vam(categorical,discreteStateSetIntCat)},
    {"descriptors",15,0,5,0,0,0,0,0,0,N_vam(strL,discreteStateSetIntLabels)},
    {"elements",13,0,2,1,0,0,0,0,0,N_vam(newivec,Var_Info_dssi)},
    {"elements_per_variable",0x80d,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_ndssi)},
    {"initial_state",13,0,4,0,0,0,0,0,0,N_vam(ivec,discreteStateSetIntVars)},
    {"num_set_values",0x805,0,1,0,0,0,0,0,-2,N_vam(newiarray,Var_Info_ndssi)},
    {"set_values",5,0,2,1,0,0,0,0,-4,N_vam(newivec,Var_Info_dssi)}
}

KeyWord kw_370[7] [static]
Initial value:

= {
    {"categorical",15,0,3,0,0,0,0,0,0,N_vam(categorical,discreteStateSetRealCat)},
    {"descriptors",15,0,5,0,0,0,0,0,0,N_vam(strL,discreteStateSetRealLabels)},
    {"elements",14,0,2,1,0,0,0,0,0,N_vam(newrvec,Var_Info_dssr)},
    {"elements_per_variable",0x80d,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_ndssr)},
    {"initial_state",14,0,4,0,0,0,0,0,0,N_vam(rvec,discreteStateSetRealVars)},
    {"num_set_values",0x805,0,1,0,0,0,0,0,-2,N_vam(newiarray,Var_Info_ndssr)},
    {"set_values",6,0,2,1,0,0,0,0,-4,N_vam(newrvec,Var_Info_dssr)}
}
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_371[6] [static]
Initial value:

```c
= {
    {"descriptors",15,0,4,0,0,0,0,0,0,N,\vam(strl,discreteStateSetStrLabels)},
    {"elements",15,0,4,0,0,0,0,0,0,N,\vam(newarray,Var_Info_dssss)},
    {"initial_state",15,0,3,0,0,0,0,0,0,N,\vam(strl,discreteStateSetStrVars)},
    {"num_set_values",8x805,0,1,0,0,0,0,0,-2,N,\vam(newarray,Var_Info_dssss)},
    {"set_values",7,0,2,1,0,0,0,0,0,-4,N,\vam(newarray,Var_Info_dssss)}
}
```

KeyWord kw_372[3] [static]
Initial value:

```c
= {
    {"integer",0x19,7,1,0,kw_369,0,0,0,0,N,\vam(pintz,numDiscreteStateSetIntVars)},
    {"real",0x19,7,3,0,kw_370,0,0,0,0,N,\vam(pintz,numDiscreteStateSetRealVars)},
    {"string",0x19,6,2,0,kw_371,0,0,0,0,N,\vam(pintz,numDiscreteStateSetStrVars)}
}
```

KeyWord kw_373[9] [static]
Initial value:

```c
= {
    {"categorical",15,0,4,0,0,0,0,0,0,N,\vam(categorical,discreteUncSetIntCat)},
    {"descriptors",15,0,6,0,0,0,0,0,0,N,\vam(deuilbl,DEUIVar_set_int)},
    {"elements",13,0,2,1,0,0,0,0,0,0,N,\vam(newvec,Var_Info_dusi)},
    {"initial_point",13,0,5,0,0,0,0,0,0,0,N,\vam(ivec,discreteUncSetIntVars)},
    {"num_set_values",5,0,1,0,0,0,0,0,0,-2,N,\vam(newarray,Var_Info_dusi)},
    {"set_probabilities",14,0,3,0,0,0,0,0,0,0,N,\vam(newrvec,Var_Info_DSIp)},
    {"set_probs",6,0,3,0,0,0,0,0,0,0,0,N,\vam(newrvec,Var_Info_DSIp)},
    {"set_values",5,0,2,1,0,0,0,0,0,-6,N,\vam(newvec,Var_Info_dusi)}
}
```

KeyWord kw_374[9] [static]
Initial value:

```c
= {
    {"categorical",15,0,4,0,0,0,0,0,0,N,\vam(categorical,discreteUncSetRealCat)},
    {"descriptors",15,0,6,0,0,0,0,0,0,N,\vam(deurlbl,DEURVar_set_real)},
    {"elements",14,0,2,1,0,0,0,0,0,0,N,\vam(newrvec,Var_Info_dusr)},
    {"initial_point",14,0,5,0,0,0,0,0,0,0,N,\vam(rvec,discreteUncSetRealVars)},
    {"num_set_values",5,0,1,0,0,0,0,0,0,-2,N,\vam(newarray,Var_Info_dusi)},
    {"set_probabilities",14,0,3,0,0,0,0,0,0,0,N,\vam(newrvec,Var_Info_DSRp)},
    {"set_probs",6,0,3,0,0,0,0,0,0,0,0,N,\vam(newrvec,Var_Info_DSRp)},
    {"set_values",6,0,2,1,0,0,0,0,0,-6,N,\vam(newvec,Var_Info_dusi)}
}
```
CHAPTER 12. NAMESPACE DOCUMENTATION

KeyWord kw_375[8] [static]
Initial value:
= {
  "descriptors",15,0,5,0,0.,0.,0,N_vae(deuslbl,DEUSVar_set_str),
  "elements",15,0,2,1,0,0.,0.,0,N_vam(newarray,Var_Info_duss),
  "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(newarray,Var_Info_duss),
  "set_probabilities",14,0,3,0,0.,0.,0.,0,N_vam(newrvec,Var_Info_DSSp),
  "set_probs",6,0,3,0,0.,0.,-1,N_vam(newrvec,Var_Info_DSSp),
  "set_values",7,0,2,1,0,0.,0.,-6,N_vam(newarray,Var_Info_duss)
}

KeyWord kw_376[3] [static]
Initial value:
= {
  "integer",0x19,9,1,0,pintz,numDiscreteUncSetIntVars),
  "real",0x19,9,3,0,pintz,numDiscreteUncSetRealVars)),
  "string",0x19,8,2,0,pintz,numDiscreteUncSetStrVars)
}

KeyWord kw_377[5] [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(RealLb,frechetUncVars)
}

KeyWord kw_378[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_379[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)
}

12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_380[4]  [static]

Initial value:

\[
= \{
\{\text{"descriptors"}, 15, 0, 3, 0, 0, 0, 0, 0, 0, \text{N,vam(\text{dautilbl, DAUIVar, geometric})} \},
\{\text{"prob_per_trial"}, 6, 0, 1, 1, 0, 0, 0, 0, \text{N,vam(\text{rvec, geometricUncProbPerTrial})} \},
\{\text{"probability_per_trial"}, 14, 0, 1, 1, 0, 0, 0, 0, \text{N,vam(\text{rvec, geometricUncProbPerTrial})} \}
\}
\]

KeyWord kw_381[7]  [static]

Initial value:

\[
= \{
\{\text{"alphas"}, 14, 0, 1, 0, 0, 0, 0, 0, \text{N,vam(\text{RealLb, gumbelUncAlphas})} \},
\{\text{"betas"}, 14, 0, 2, 0, 0, 0, 0, \text{N,vam(\text{rvec, gumbelUncBetas})} \},
\{\text{"descriptors"}, 15, 0, 4, 0, 0, 0, 0, 0, \text{N,vae(\text{caulbl, CAUVar, gumbel})} \},
\{\text{"guuv_alphas"}, 6, 0, 1, 1, 0, 0, 0, -3, \text{N,vam(\text{RealLb, gumbelUncAlphas})} \},
\{\text{"guuv_betas"}, 6, 0, 2, 0, 0, 0, 0, -3, \text{N,vam(\text{rvec, gumbelUncBetas})} \},
\{\text{"guuv_descriptors"}, 7, 0, 4, 0, 0, 0, 0, -3, \text{N,vae(\text{caulbl, CAUVar, gumbel})} \},
\{\text{"initial_point"}, 14, 0, 3, 0, 0, 0, 0, 0, \text{N,vam(\text{rvec, gumbelUncVars})} \}
\}
\]

KeyWord kw_382[11]  [static]

Initial value:

\[
= \{
\{\text{"abscissas"}, 14, 0, 2, 1, 0, 0, 0, 0, 0, \text{N,vam(\text{newrvec, VarInfo, hba})} \},
\{\text{"counts"}, 14, 0, 3, 2, 0, 0, 0, 0, \text{N,vam(\text{newrvec, VarInfo, hbc})} \},
\{\text{"descriptors"}, 15, 0, 5, 0, 0, 0, 0, 0, \text{N,vae(\text{caulbl, CAUVar, histogram_bin})} \},
\{\text{"huv_bin_abscissas"}, 6, 0, 2, 1, 0, 0, 0, -3, \text{N,vam(\text{newrvec, VarInfo, hba})} \},
\{\text{"huv_bin_counts"}, 6, 0, 3, 2, 0, 0, 0, 0, 0, \text{N,vam(\text{newrvec, VarInfo, hbc})} \},
\{\text{"huv_bin_descriptors"}, 7, 0, 4, 0, 0, 0, 0, -3, \text{N,vae(\text{caulbl, CAUVar, histogram_bin})} \},
\{\text{"huv_bin ordinates"}, 6, 0, 2, 1, 0, 0, 0, -3, \text{N,vam(\text{newrvec, VarInfo, hbo})} \},
\{\text{"initial_point"}, 14, 0, 3, 0, 0, 0, 0, 0, \text{N,vam(\text{rvec, histogramBinUncVars})} \},
\{\text{"num_pairs"}, 5, 0, 1, 0, 0, 0, 0, -2, \text{N,vam(\text{newiarray, VarInfo, nhbp})} \},
\{\text{"ordinates"}, 14, 0, 3, 2, 0, 0, 0, 0, \text{N,vam(\text{newrvec, VarInfo, hbo})} \},
\{\text{"pairs_per_variable"}, 13, 0, 1, 0, 0, 0, 0, 0, \text{N,vam(\text{newiarray, VarInfo, nhbp})} \}
\}
\]

KeyWord kw_383[6]  [static]

Initial value:

\[
= \{
\{\text{"abscissas"}, 13, 0, 2, 1, 0, 0, 0, 0, \text{N,vam(\text{newivec, VarInfo, hpia})} \},
\{\text{"counts"}, 14, 0, 3, 2, 0, 0, 0, 0, \text{N,vam(\text{newrvec, VarInfo, hpic})} \},
\{\text{"descriptors"}, 15, 0, 5, 0, 0, 0, 0, 0, \text{N,vae(\text{dautilbl, DAUIVar, histogram_point_int})} \},
\{\text{"initial_point"}, 13, 0, 4, 0, 0, 0, 0, 0, \text{N,vam(\text{rvec, histogramPointIntUncVars})} \},
\{\text{"num_pairs"}, 5, 0, 1, 0, 0, 0, 0, -1, \text{N,vam(\text{newiarray, VarInfo, nhpip})} \},
\{\text{"ordinates"}, 14, 0, 3, 2, 0, 0, 0, 0, \text{N,vam(\text{newrvec, VarInfo, hbo})} \},
\{\text{"pairs_per_variable"}, 13, 0, 1, 0, 0, 0, 0, 0, \text{N,vam(\text{newiarray, VarInfo, nhpip})} \}
\}
\]
KeyWord kw_384[6] [static]
Initial value:
= {
   {"abscissas",14,0,2,1,0,0,0,0,0,N_vam(newrvec,Var_Info_hpra)},
   {"counts",14,0,3,2,0,0,0,0,0,N_vam(newrvec,Var_Info_hpcr)},
   {"descriptors",15,0,3,0,0,0,0,0,0,N_vae dauibr1,DMURVar_histogram_point_real}),
   {"initial_point",14,0,4,0,0,0,0,0,0,N_vam(rvec, histogramPointRealUncVars)},{
   "num_pairs",5,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_nhprp)},
   {"pairs_per_variable",13,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_nhprp)}
   }

KeyWord kw_385[6] [static]
Initial value:
= {
   {"abscissas",15,0,2,1,0,0,0,0,0,N_vam(newarray,Var_Info_hpsa)},
   {"counts",14,0,3,2,0,0,0,0,0,N_vam(newrvec,Var_Info_hpsr)},
   {"descriptors",15,0,5,0,0,0,0,0,0,N_vae (dauslbl, DAUSVar_histogram_point_real)},
   {"initial_point",15,0,4,0,0,0,0,0,0,N_vam(strl_histogramPointStrUncVars)},
   {"num_pairs",5,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_nhpsp)},
   {"pairs_per_variable",13,0,1,0,0,0,0,0,0,N_vam(newiarray,Var_Info_nhpsp)}
   }

KeyWord kw_386[3] [static]
Initial value:
= {
   {"integer",0x19,6,1,0,kw_383,0,0,0,0,N_vam(pintz,numHistogramPtIntUncVars)},
   {"real",0x19,6,3,0,kw_384,0,0,0,0,N_vam(pintz,numHistogramPtRealUncVars)},
   {"string",0x19,6,2,0,kw_385,0,0,0,0,N_vam(pintz,numHistogramPtStrUncVars)}
   }

KeyWord kw_387[5] [static]
Initial value:
= {
   {"descriptors",15,0,5,0,0,0,0,0,0,N_vae dauulbl,DAUIVar_hypergeometric)},
   {"initial_point",13,0,4,0,0,0,0,0,0,N_vam(Intlb, hyperGeomUncVars)},
   {"num_drawn",13,0,1,3,0,0,0,0,0,N_vam(Intlb, hyperGeomUncNumDrawn)},
   {"selected_population",13,0,2,2,0,0,0,0,0,N_vam(Intlb, hyperGeomUncSelectedPop)},
   {"total_population",13,0,1,1,0,0,0,0,0,N_vam(Intlb, hyperGeomUncTotalPop)}
   }

KeyWord kw_388[2] [static]
Initial value:
= {
   {"lnuv_zetas",6,0,1,1,0,0,0,0,1,N_vam(Reallb, lognormalUncZetas)},
   {"zetas",14,0,1,1,0,0,0,0,0,N_vam(Reallb, lognormalUncZetas)}
   }
12.1. DAKOTA NAMESPACE REFERENCE

KeyWord kw_389[4] [static]
Initial value:
= {
  \{"error_factors",14,0,1,1,0,0,0,0,N_vam(RealLb,lognormalUncErrFacts)},
  \{"lnuv.std_deviations",6,0,1,1,0,0,0,0,1,N_vam(RealLb,lognormalUncStdDevs)},
  \{"std_deviations",14,0,1,1,0,0,0,0,N_vam(RealLb,lognormalUncStdDevs)}
}

KeyWord kw_390[11] [static]
Initial value:
= {
  \{"descriptors",15,0,5,0,0,0,0,0,N_vae(caulbl,CAUVar_lognormal)},
  \{"initial_point",14,0,6,0,0,0,0,0,N_vam(RealLb,lognormalUncVars)},
  \{"lambdas",14,2,1,1,kw_388,0,0,0,0,N_vam(rvec,lognormalUncLambdas)},
  \{"lnuv.descriptors",7,0,2,0,0,0,0,0,3,N_vae(caulbl,CAUVar_lognormal)},
  \{"lnuv.lambdas",6,2,1,1,kw_388,0,0,0,0,N_vam(rvec,lognormalUncLambdas)},
  \{"lnuv.lower_bounds",6,0,2,0,0,0,0,0,3,N_vam(RealLb,lognormalUncLowerBnds)},
  \{"lnuv.means",6,4,1,1,kw_389,0,0,0,0,3,N_vam(RealLb,lognormalUncMeans)},
  \{"lnuv.upper_bounds",6,0,3,0,0,0,0,0,3,N_vam(RealUb,lognormalUncUpperBnds)},
  \{"means",14,4,1,1,kw_389,0,0,0,0,N_vam(RealLb,lognormalUncMeans)},
  \{"upper_bounds",14,0,3,0,0,0,0,0,N_vam(RealUb,lognormalUncUpperBnds)}
}

KeyWord kw_391[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",7,0,4,0,0,0,0,0,3,N_vae(caulbl,CAUVar_loguniform)},
  \{"luuv.lower_bounds",6,0,1,1,0,0,0,0,0,2,N_vam(RealLb,loguniformUncLowerBnds)},
  \{"luuv.means",6,4,1,1,kw_389,0,0,0,0,3,N_vam(RealLb,loguniformUncMeans)},
  \{"luuv.upper_bounds",6,0,2,2,0,0,0,0,1,N_vam(RealUb,loguniformUncUpperBnds)},
  \{"upper_bounds",14,0,2,2,0,0,0,0,0,N_vam(RealUb,loguniformUncUpperBnds)}
}

KeyWord kw_392[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",13,0,2,2,0,0,0,0,0,N_vam(IntLb,negBinomialUncNumTrials)},
  \{"probability.per_trial",6,0,1,1,0,0,0,0,1,N_vam(rvec,negBinomialUncProbPerTrial)},
  \{"probability",14,0,1,1,0,0,0,0,0,N_vam(rvec,negBinomialUncProbPerTrial)}}
KeyWord kw.393[11] [static]
Initial value:

```cpp
= {
  {"descriptors",15,0,6,0,0,0,0,0,Nstdarg(caulbl,CAUVar_normal)},
  {"initial_point",14,0,5,0,0,0,0,0,0,Nstdarg(rvec,normalUncVars)},
  {"loweربع bounds",14,0,3,0,0,0,0,0,0,Nstdarg(rvec,normalUncLowerBnds)},
  {"means",14,0,1,0,0,0,0,0,0,Nstdarg(rvec,normalUncMeans)},
  {"nuv descriptors",7,0,6,0,0,0,0,0,Nstdarg(caulbl,CAUVar_normal)},
  {"nuv_lower_bounds",6,0,3,0,0,0,0,0,Nstdarg(rvec,normalUncLowerBnds)},
  {"nuv_means",6,0,1,0,0,0,0,0,Nstdarg(rvec,normalUncMeans)},
  {"nuv_std_deviations",6,0,2,2,0,0,0,0,Nstdarg(RealLb,normalUncStdDevs)},
  {"nuv_upper_bounds",6,0,4,0,0,0,0,0,2,Nstdarg(rvec,normalUncUpperBnds)},
  {"std deviations",14,0,2,2,0,0,0,0,Nstdarg(RealLb,normalUncStdDevs)},
  {"upper bounds",14,0,4,0,0,0,0,0,0,Nstdarg(rvec,normalUncUpperBnds)}
}
```

KeyWord kw.394[3] [static]
Initial value:

```cpp
= {
  {"descriptors",15,0,3,0,0,0,0,0,Nstdarg(dauilbl,DAUIVar_poisson)},
  {"initial_point",13,0,2,0,0,0,0,0,Nstdarg(IntLb,poissonUncVars)},
  {"lambdas",14,0,1,1,0,0,0,0,0,Nstdarg(RealLb,poissonUncLambdas)}
}
```

KeyWord kw.395[9] [static]
Initial value:

```cpp
= {
  {"descriptors",15,0,5,0,0,0,0,0,Nstdarg(caulbl,CAUVar_triangle)},
  {"initial_point",14,0,4,0,0,0,0,0,0,Nstdarg(rvec,triangleUncVars)},
  {"lower_bounds",14,0,2,2,0,0,0,0,0,0,Nstdarg(rvec,triangleUncLowerBnds)},
  {"modes",14,0,1,1,0,0,0,0,0,0,Nstdarg(rvec,triangleUncModes)},
  {"uuv descriptors",7,0,5,0,0,0,0,0,0,Nstdarg(caulbl,CAUVar_triangle)},
  {"uuv_lower_bounds",6,0,2,2,0,0,0,0,0,0,Nstdarg(rvec,triangleUncLowerBnds)},
  {"uuv_modes",6,0,1,1,0,0,0,0,0,0,Nstdarg(rvec,triangleUncModes)},
  {"uuv_upper_bounds",6,0,3,3,0,0,0,0,0,0,Nstdarg(rvec,triangleUncUpperBnds)},
  {"upper bounds",14,0,3,3,0,0,0,0,0,0,Nstdarg(RealUb,triangleUncUpperBnds)}
}
```

KeyWord kw.396[7] [static]
Initial value:

```cpp
= {
  {"descriptors",15,0,4,0,0,0,0,0,0,Nstdarg(caulbl,CAUVar_uniform)},
  {"initial_point",14,0,3,0,0,0,0,0,0,Nstdarg(rvec,uniformUncVars)},
  {"lower_bounds",14,0,1,1,0,0,0,0,0,0,Nstdarg(rvec,uniformUncLowerBnds)},
  {"uuv_descriptors",7,0,4,0,0,0,0,0,0,Nstdarg(caulbl,CAUVar_uniform)},
  {"uuv_lower_bounds",6,0,1,1,0,0,0,0,0,0,Nstdarg(rvec,uniformUncLowerBnds)},
  {"uuv_upper_bounds",6,0,2,2,0,0,0,0,0,0,Nstdarg(rvec,uniformUncUpperBnds)}
}
```
KeyWord kw_397[7] [static]
Initial value:

```c++
= {
    "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",6,0,1,1,0,0,0,0,-4,N_vam(RealLb,weibullUncAlphas),
    "wuv_betas",6,0,2,2,0,0,0,0,-4,N_vam(RealLb,weibullUncBetas),
    "wuv_descriptors",7,0,4,0,0,0,0,-4,N_vae(caulbl,CAUVar_weibull)
}
```

KeyWord kw_399[6] [static]
Initial value:

```c++
= {
    "environment",0x108,15,1,1,kw,12,0.,0.,0,NIDRProblemDescDB::env_start,
    "interface",0x308,9,5,5,kw,26,0.,0.,0,ifm3(start,0,stop),
    "method",0x308,90,2,2,kw,291,0.,0,0,N_dmdm3(start,0,stop),
    "model",8,7,3,kw,327,0.,0,0,N_mom3(start,0,stop),
    "responses",0x308,19,6,6,kw,353,0.,0,0,N_reml3(start,0,stop),
    "variables",0x308,34,4,kw,398,0.,0,0,N_vam3(start,0,stop)
}
```

Var_uinfo CAUVLbl[CAUVar_Nkinds] [static]
Initial value:

```c++
= {
    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 DAUVLbl[DAUVar_Nkinds] [static]
Initial value:

```c++
= {
    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:

```c
  = {
    VarLabelInfo(hpsuv, HistogramPtStrUnc)
  }
```

Var_uinfo DAURVLbl[DAURVar_Nkinds] [static]
Initial value:

```c
  = {
    VarLabelInfo(hpruv, HistogramPtRealUnc)
  }
```

Var_uinfo CEUVLbl[CEUVar_Nkinds] [static]
Initial value:

```c
  = {
    VarLabelInfo(ciuv, ContinuousIntervalUnc)
  }
```

Var_uinfo DEUIVLbl[DEUIVar_Nkinds] [static]
Initial value:

```c
  = {
    VarLabelInfo(diuv, DiscreteIntervalUnc),
    VarLabelInfo(dusiv, DiscreteUncSetInt)
  }
```

Var_uinfo DEUSVLbl[DEUSVar_Nkinds] [static]
Initial value:

```c
  = {
    VarLabelInfo(dussv, DiscreteUncSetStr)
  }
```

Var_uinfo DEURVLbl[DEURVar_Nkinds] [static]
Initial value:

```c
  = {
    VarLabelInfo(dusrv, DiscreteUncSetReal)
  }
```
12.1. DAKOTA NAMESPACE REFERENCE

Var_uinfo DiscSetLbl[DiscSetVar_Nkinds] [static]

Initial value:

```c
= {
  VarLabelInfo(ddsv, DiscreteDesSetStr),
  VarLabelInfo(ddssv, DiscreteDesSetStr),
  VarLabelInfo(ddsr, DiscreteDesSetReal),
  VarLabelInfo(dss, DiscreteStateSetStr),
  VarLabelInfo(dssv, DiscreteStateSetStr),
  VarLabelInfo(dssr, DiscreteStateSetReal)
}
```

VarLabelChk DesignAndStateLabelsCheck[] [static]

Initial value:

```c
= {
  { 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, "dssrv", "dssrv_descriptors" },
  { AVI numContinuousStateVars, AVI continuousStateLabels, "csv", "csv_descriptors" }
}
```

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:

```c
= {
  {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:

= {
  {DAUIVarNkinds, AVI DAUIv, DAUIVLbl,
   DVR discreteIntAleatoryUncLabels,
   DVR discreteIntAleatoryUncLowerBnds,
   DVR discreteIntAleatoryUncUpperBnds,
   DVR discreteIntAleatoryUncVars},
  {DEUIVarNkinds, 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:

= {
  {DAUSVarNkinds, AVI DAUSv, DAUSVLbl,
   DVR discreteStrAleatoryUncLabels,
   DVR discreteStrAleatoryUncLowerBnds,
   DVR discreteStrAleatoryUncUpperBnds,
   DVR discreteStrAleatoryUncVars},
  {DEUSVarNkinds, 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) }
12.1. **DAKOTA NAMESPACE REFERENCE**

Var_check var_mp_check_cau[] [static]

Initial value:

```c
= {
    Vchk_j(normal_uncertain,NormalUnc),
    Vchk_j(lognormal_uncertain,LognormalUnc),
    Vchk_j(uniform_uncertain,UniformUnc),
    Vchk_j(loguniform_uncertain,LoguniformUnc),
    Vchk_j(triangular_uncertain,TriangularUnc),
    Vchk_j(exponential_uncertain,ExponentialUnc),
    Vchk_j(beta_uncertain,BetaUnc),
    Vchk_j(gamma_uncertain,GammaUnc),
    Vchk_j(gumbel_uncertain,GumbelUnc),
    Vchk_j(frechet_uncertain,FrechetUnc),
    Vchk_j(weibull_uncertain,WeibullUnc),
    Vchk_j(histogram_bin_uncertain,HistogramBinUnc) }
```

Var_check var_mp_check_daiui[] [static]

Initial value:

```c
= {
    Vchk_j(poisson_uncertain,PoissonUnc),
    Vchk_j(binomial_uncertain,BinomialUnc),
    Vchk_j(negative_binomial_uncertain,NegBinomialUnc),
    Vchk_j(geometric_uncertain,GeometricUnc),
    Vchk_j(hypergeometric_uncertain,HyperGeomUnc),
    Vchk_j(histogram_point_int_uncertain,HistogramPtIntUnc) }
```

Var_check var_mp_check_daus[] [static]

Initial value:

```c
= {
    Vchk_j(histogram_point_str_uncertain,HistogramPtStrUnc) }
```

Var_check var_mp_check_daur[] [static]

Initial value:

```c
= {
    Vchk_j(histogram_point_real_uncertain,HistogramPtRealUnc) }
```

Var_check var_mp_check_ceu[] [static]

Initial value:

```c
= {
    Vchk_j(continuous_interval_uncertain,ContinuousIntervalUnc) }
```

Var_check var_mp_check_deui[] [static]

Initial value:

```c
= {
    Vchk_j(discrete_interval_uncertain,DiscreteIntervalUnc),
    Vchk_j(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_check 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(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_check 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",
  "numADRVT", "numDerivVars", "xC", "xDI",
  "xDR", "xDLabels", "xDLabels",
  "xDKLabels", "directFnAV", "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().
12.2. SIM NAMESPACE REFERENCE

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)
13.2. ANALYZER CLASS REFERENCE

- **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

- **update model’s current variables with data from sample**
  
  - 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
  
  bool `returns_multiple_points()` () const
  
  returns Analyzer::compactMode

- **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`
13.2. ANALYZER CLASS REFERENCE

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

```cpp
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, FSUDESDesignCompExp, and PSUADEDesignCompExp.

References Model::derivative_concurrency(), Iterator::iteratedModel, and Iterator::maxEvalConcurrency.

Referenced by NonDGlobalReliability::get_best(), NonDPolynomialChaos::ratio_samples_to_order(), Analyzer::samples_to_variables_array(), Analyzer::variables_array_to_samples(), and Analyzer::variance_based_decomp().

```cpp
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().

```cpp
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().
```cpp
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(), ParallelLibrary::command_line_user_modes(), Analyzer::compactMode, Variables::copy(), Model::current_response(), Model::current_variables(), Model::interface_id(), Iterator::iteratedModel, Iterator::outputLevel, Iterator::parallelLib, ProgramOptions::pre_run_output_format(), ParallelLibrary::program_options(), 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, NonDPOFDarts, NonDQUESOBayesCalibration, NonDLocalReliability, NonDAdaptiveSampling, NonDGPlmpSampling, NonDAdaptImpSampling, NonDGlobalReliability, NonDIncrementalHSSampling, NonDInterval, NonDExpansion, PStudyDACE, RichExtrapVerification, and Verification.

References Analyzer::bestVarsRespMap, ParamResponsePair::eval_id(), Response::function_values(), Analyzer::numLSqTerms, Analyzer::numObjFns, ParamResponsePair::response(), and ParamResponsePair::variables().

Referenced by Analyzer::post_run, Verification::print_results(), PStudyDACE::print_results(), and NonDLHSSampling::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, FSUSSDesignCompExp, 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(), FSUSSDesignCompExp::extract_trends(), ParamStudy::extract_trends(), DDACEDesignCompExp::extract_trends(), NonDIncrementLHSSampling::quantify_uncertainty(), NonDAdaptImpSampling::quantify_uncertainty(), NonDLHSSampling::quantify_uncertainty(), NonDIntegration::quantify_uncertainty(), NonDQUESOBayesCalibration::update_model(), and Analyzer::variance_based_decomp().

```cpp
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 FSUSSDesignCompExp::extract_trends(), DDACEDesignCompExp::extract_trends(), and NonDLHSSampling::quantify_uncertainty().

```cpp
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, ProgramOptions::post_run_input_format(), ParallelLibrary::program_options(), Variables::read_tabular(), Analyzer::update_best(), and Analyzer::variables_to_sample().

Referenced by PSUADSSDesignCompExp::post_input(), FSUSSDesignCompExp::post_input(), ParamStudy::post_input(), DDACEDesignCompExp::post_input(), and NonDLHSSampling::post_input().

```cpp
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().
CHAPTER 13. CLASS DOCUMENTATION

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:

```
  Interface
    ApplicationInterface
  DirectApplicInterface
  ProcessApplicInterface
    MatlabInterface
    PythonInterface
    ScilabInterface
    TestDriverInterface
    ParallelDirectApplicInterface
    SerialDirectApplicInterface
    ProcessHandleApplicInterface
    SysCallApplicInterface
```

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 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 asyncLocalAnalysisFlag
  flag for asynchronous local parallelism of analyses
• int asyncLocalAnalysisConcurrency
  limits the number of concurrent analyses in asynchronous local scheduling and specifies hybrid concurrency when message passing
• int asyncLocalEvalConcSpec
  user specification for asynchronous local evaluation concurrency
• int asyncLocalAnalysisConcSpec
  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 init_default_asv (size_t num_fns)
  initialize default ASV if needed; this is done at run time due to post-construct time Response size changes.

• 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_async_local (PRPQueueIter &prp_it)
  launch an asynchronous local evaluation

• void process_async_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 async 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 async 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 async 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
13.3. APPLICATION INTERFACE CLASS REFERENCE

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 asychLocalEvalStatic
  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 async 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).

- **String** `gradientType`  
  type of gradients present in associated Response

- **String** `hessianType`  
  type of Hessians present in associated Response

- **IntSet** `gradMixedAnalyticIds`  
  IDs of analytic gradients when mixed gradients present.

- **IntSet** `hessMixedAnalyticIds`  
  IDs of analytic gradients when mixed gradients present.

- **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 `DirectApplicationInterface::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::asv_mapping(), ApplicationInterface::asvControlFlag, ApplicationInterface::beforeSynchAlgPRPQueue, ApplicationInterface::beforeSynchCorePRPQueue, ApplicationInterface::beforeSynchDuplicateMap, 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(), ApplicationInterface::init_default_asv(), Interface::init_evaluation_counters(), 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().

\textbf{const IntResponseMap & synch( ) [protected], [virtual]} \hspace{1cm}

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 \texttt{synchronize()} in derived \texttt{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().

\textbf{const IntResponseMap & synch_nowait( ) [protected], [virtual]} \hspace{1cm}

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 \texttt{synchronize\_nowait()} in derived \texttt{Model} classes.

Reimplemented from Interface.

References Dakota::abort_handler(), Interface::algebraic_mappings(), 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(), Interface::rawResponseMap, ParamResponsePair::response(), Interface::response_mapping(), and Response::update().

\textbf{void serve_evaluations( ) [protected], [virtual]} \hspace{1cm}

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(), 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_synch(), ApplicationInterface::serve_evaluations_asynch(), and ApplicationInterface::serve_evaluations_synch_peer().

```cpp
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::num EvalServers, Interface::outputLevel, ParallelLibrary::parallel_configuration(), and ApplicationInterface::parallelLib.

```cpp
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.
References by ApplicationInterface::init_communicators().

```cpp
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.

References by ApplicationInterface::set_communicators().

```cpp
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().

```cpp
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::irecv_ea(), ApplicationInterface::synchronous_local_analysis(), and ParallelLibrary::wait().

Referenced by ProcessHandleApplicInterface::create_evaluation_process(), SysCallApplicInterface::create_evaluation_process(), and DirectApplicInterface::derived_map().

```cpp
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 history-DuplicateMap 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().

```cpp
void init_default_asv( size_t num_fns ) [private]

initialize default ASV if needed; this is done at run time due to post-construct time Response size changes.

If the user has specified active_set_vector as off, then map() uses a default ASV which is constant for all function evaluations (so that the user need not check the content of the ASV on each evaluation). Only initialized if needed and not already sized.

References ApplicationInterface::asvControlFlag, ApplicationInterface::defaultASV, ApplicationInterface::gradient-Type, ApplicationInterface::gradMixedAnalyticIds, ApplicationInterface::hessianType, and ApplicationInterface::hessMixedAnalyticIds.

Referenced by ApplicationInterface::map().

```cpp
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::recv Buffers`, `ApplicationInterface::recvRequests`, `ApplicationInterface::send_evaluation()`, `ApplicationInterface::sendBuffers`, `ParallelLibrary::waitall()`, and `ParallelLibrary::waitsome()`.

Referenced by `ApplicationInterface::synch()`.

```cpp
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(nowait()) 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 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::recv Buffers`, `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::recv Buffers`, `ApplicationInterface::recvRequests`, `ParallelLibrary::waitall()`, and `ParallelLibrary::waitsome()`.

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 async 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 asynchronous_local_queue(), ApplicationInterface::assign asynchronous_local_queue_nowait(), ApplicationInterface::asynchLocalActivePRPQueue, ApplicationInterface::asynchLocalEvalConcurrency, ApplicationInterface::beforeSynchCorePRPQueue, Dakota::lookup by eval id(), ApplicationInterface::msgPassRunningMap, ApplicationInterface::numEvalServers, ApplicationInterface::recv BUFFERS, ApplicationInterface::recvRequests, ApplicationInterface::send_evaluation(), ApplicationInterface::send BUFFERS, 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, non-blocking fork, or threads). It is called from synch_nowait() and passed the complete set of all asynchronous jobs (beforeSynchCorePRPQueue). It uses derived_map asynchronous() 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 raw-ResponseMap, 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 asynchronous_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 ie(), Interface::currEvalId, ApplicationInterface::derived_map(), ApplicationInterface::evalCommRank, ParallelLibrary::send 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::recv_iid(), ApplicationInterface::lenResponseMessage, ApplicationInterface::lenVarsActSetMessage, Dakota::lookup_by_eval_id(), Interface::multiProcEvalFlag, ApplicationInterface::parallelLib, ParallelLibrary::recv_iid(), MPIUnpackBuffer::reset(), ParallelLibrary::send_iid(), 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 Real value (const RealVector &c_vars)**
  
  *retrieve the approximate function value for a given parameter vector*

- **virtual const RealVector & gradient (const RealVector &c_vars)**
virtual const RealSymMatrix & hessian (const RealVector &c_vars)  
retrieve the approximate function Hessian for a given parameter vector

virtual Real prediction_variance (const RealVector &c_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)
13.4. APPROXIMATION CLASS REFERENCE

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)
CHAPTER 13. CLASS DOCUMENTATION

- **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
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, VPSApproximation, 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(), VPSApproximation::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(), ProblemDescDB::get_bool(), 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:

   Interface
     ApproximationInterface
Public Member Functions

- \texttt{ApproximationInterface} (\texttt{ProblemDescDB &problem\_db}, const \texttt{Variables &am\_vars}, bool \texttt{am\_cache}, const \texttt{String &am\_interface\_id}, size\_t \texttt{num\_fns})
  
  
  primary constructor

- \texttt{ApproximationInterface} (const \texttt{String &approx\_type}, const \texttt{UShortArray &approx\_order}, const \texttt{Variables &am\_vars}, bool \texttt{am\_cache}, const \texttt{String &am\_interface\_id}, size\_t \texttt{num\_fns}, short \texttt{data\_order}, short \texttt{output\_level})

  alternate constructor for instantiations on the fly

- \texttt{\sim ApproximationInterface} ()

Protected Member Functions

- void \texttt{map} (const \texttt{Variables &vars}, const \texttt{ActiveSet &set}, \texttt{Response &response}, bool \texttt{asynch\_flag=false})
  
  the function evaluator: provides an approximate “mapping” from the variables to the responses using function-Surfaces

- int \texttt{minimum\_points} (bool \texttt{constraint\_flag}) const

  returns the minimum number of samples required to build the functionSurfaces

- int \texttt{recommended\_points} (bool \texttt{constraint\_flag}) const

  returns the recommended number of samples recommended to build the functionSurfaces

- void \texttt{approximation\_function\_indices} (const \texttt{IntSet &approx\_fn\_indices})

  set the (currently active) approximation function index set

- void \texttt{update\_approximation} (const \texttt{Variables &vars}, const \texttt{IntResponsePair &response\_pr})

- void \texttt{append\_approximation} (const \texttt{VariablesArray &vars\_array}, const \texttt{IntResponseMap &resp\_map})

- void \texttt{build\_approximation} (const \texttt{RealVector &c\_lbnds}, const \texttt{RealVector &c\_ubnds}, const \texttt{IntVector &dim\_lbnds}, const \texttt{IntVector &dim\_ubnds}, const \texttt{RealVector &dr\_lbnds}, const \texttt{RealVector &dr\_ubnds})

- void \texttt{rebuild\_approximation} (const \texttt{BoolDeque &rebuild\_deque})

- void \texttt{pop\_approximation} (bool \texttt{save\_surr\_data})

- void \texttt{restore\_approximation} ()

- bool \texttt{restore\_available} ()

  queries the approximation for the ability to restore a previous increment

- void \texttt{finalize\_approximation} ()

  finalizes the approximation by applying all trial increments

- void \texttt{store\_approximation} ()

  move the current approximation into storage for later combination

- void \texttt{combine\_approximation} (short \texttt{corr\_type})

  combine the current approximation with one previously stored

- void \texttt{clear\_current} ()

  clears current data from an approximation interface

- void \texttt{clear\_all} ()
13.5. APPROXIMATIONINTERFACE CLASS REFERENCE

- **void clear_saved()**
  clears all data 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)

- unsigned short challengeFormat
  tabular format of the challenge points file

- 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 & d_l_bnds, const IntVector & d_u_bnds, const RealVector & d_r_l_bnds, const RealVector & d_r_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::challengeFile, ApproximationInterface::challengeFormat, 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::EvalRequest-Type apps_request)**
  
  performs a function evaluation at APPS-provided \( x_{\text{in}} \)

- **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
chapter 13. class documentation

• 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()
13.6.2 Constructor & Destructor Documentation

APPSEvalMgr ( Model & model )

caractern
Evaluation manager class for APPSPACK.

The APPSEvalMgr class is derived from APPSPACK’s Executor class. It implements the methods of that class in such a way 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, const HOPSPACK::Vector & apps_f, const HOPSPACK::Vector & apps_cEqs, const HOPSPACK::Vector & apps_cIneqs, string & apps_msg )
returns a function value to APPS

Retrieve a set of response 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*
13.7. APPSOPTIMIZER CLASS REFERENCE

- 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 gen-
eralized 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, AP-
PSOptimizer::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 con-
struction time.

References APPSOptimizer::citizenParams, Minimizer::constraintTol, APPSOptimizer::evalMgr, Problem-
DescDB::get_real(), ProblemDescDB::get_string(), ProblemDescDB::is_null(), APPSOptimizer::linearParams, Iterator-
::maxEvalConcurrency, Iterator::maxFunctionEvals, APPSOptimizer::mediatorParams, Minimizer::numContinuous-
Vars, Minimizer::numNonlinearConstraints, Iterator::outputLevel, APPSOptimizer::params, Iterator::probDesc-
DB, 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, 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
13.10. COLINAPPLICATION CLASS REFERENCE

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. Required 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)
  Helper 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 async 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 async 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 evaluations 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 Mockup supports async 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 & requests, utilib::seed & seed, colin::AppResponse::response_map & colin_responses ) [virtual]

Perform a function evaluation at the domain point. Wait for and return the response. This is only called by
COLIN’s serial evaluator, which is only instantiated when the Mockup does not support async evals. The domain
point is guaranteed to be compatible with data type specified by map_domain(...) (utilib::MixedIntVars).
    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 & colin_responses, utilib::seed & 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 & requests, utilib::seed & 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 & colin_responses ) [virtual]

Helper 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:

```
        Iterator
          |  
        Minimizer
          |  
        Optimizer
          |  
  COLINOptimizer
```

#### Public Member Functions

- **COLINOptimizer (ProblemDescDB &problem_db, Model &model)**<br>  
  *standard constructor*

- **COLINOptimizer (const String &method_name, Model &model, int seed, int max_iter, int max_eval)**<br>  
  *alternate constructor for on-the-fly instantiations*

- **COLINOptimizer (const String &method_name, Model &model)**<br>  
  *alternate constructor for Iterator instantiations by name*

- **~COLINOptimizer ()**<br>  
  *destructor*

- **void reset ()**<br>  
  *clears internal optimizer state*

- **void find_optimum ()**<br>  
  *iterates the COLIN solver to determine the optimal solution*

- **bool returns_multiple_points () const**<br>  
  *some COLIN methods can return multiple points*

#### Protected Member Functions

- **void solver_setup (unsigned short method_name)**<br>  
  *convenience function for setting up the particular COLIN solver and appropriate Application*

- **void set_rng (int seed)**<br>  
  *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 constraintViolation (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 evaluation 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.
13.11. COLINOPTIMIZER CLASS REFERENCE

Referenced by COLINOptimizer::COLINOptimizer().

```cpp
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().

```cpp
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().

```cpp
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::constraintViolation(), 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().

```cpp
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::numObjective-Fns.
- 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:

```
  iterator
  metaiterator
    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*
13.13. COMMANDLINEHANDLER CLASS REFERENCE

- IntIntPair estimate_partition_bounds ()
  estimate the minimum and maximum partition sizes that can be utilized by this Iterator
- 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:

```
CommandLineHandler
  └ GetLongOpt
    └ 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:

```
  Iterator
     |    
     v    
MetaIterator
     |    
     v    
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*

- **IntIntPair estimate_partition_bounds ()**
  
  *estimate the minimum and maximum partition sizes that can be utilized by this Iterator*

- **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 fn(s) 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::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, ConcurrentMetaIterator::prpResults, ParamResponsePair::response(), ParamResponsePair::variables(), 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:

```
+----------------------------------------
|                  CONMINOptimizer       |
|                            \         |
|                             \      |
|                              \     |
|                               \    |
|                                \   |
|                                 \  |
|                                  \|
+----------------------------------------

  Optimizer
    |
    Minimizer
      |
      Iterator
```

Public Member Functions

- **CONMINOptimizer (ProblemDescDB &problem_db, Model &model)**
  *standard constructor*

- **CONMINOptimizer (const String &method_string, Model &model)**
  *alternate constructor; construct without ProblemDescDB*

- **~CONMINOptimizer ()**
CHAPTER 13. CLASS DOCUMENTATION

**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`
13.16. CONMINOPTIMIZER CLASS REFERENCE

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 NPSOL-Optimizer 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().

SizeArray constraintMappingIndices [private]

a container of indices for referencing the corresponding Response constraints used in computing the CONMIN
- 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 CON-
  MINOptimizer::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)

- \texttt{\~}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 & \texttt{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:

```
Constraints
    MixedVarConstraints
    RelaxedVarConstraints
```

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
• 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 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
13.18. CONSTRAINTS CLASS REFERENCE

- 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().
CHAPTER 13. CLASS DOCUMENTATION

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


References Constraints::constraintsRep, and Constraints::referenceCount.
13.18. CONSTRAINTS CLASS REFERENCE

**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::nonlinearEqConCoeffs, 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 DataEnvironmentRep. 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

- unsigned short preRunOutputFormat
  tabular format for pre_run output
- unsigned short postRunInputFormat
  tabular format for post_run input

- 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)

- unsigned short tabularFormat
  format for tabular data files (see enum)
- 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)
13.21. DATAFITSURRMODEL CLASS REFERENCE

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:

```
        Model
           |
           V
        SurrogateModel
           |
           V
DataFitSurrModel
```
Public Member Functions

- **DataFitSurrModel (ProblemDescDB &problem_db)**
  constructor

- **DataFitSurrModel (Iterator &dace_iterator, Model &actual_model, const ActiveSet &set, 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(), unsigned short export_format=TABLE_ANNOTATED, const String &import_points_file=String(), unsigned short import_format=TABLE_ANNOTATED, 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::surrFnIndices and ApproximationInterface::approxFnIndices
• void build_approximation ()
  Builds the local/multipoint/global approximation using dacellerator/actualModel to generate new data points.

• bool build_approximation (const Variables &vars, const IntResponsePair &response_pr)
  Builds the local/multipoint/global approximation using dacellerator/actualModel to generate new data points that augment the vars/response anchor point.

• void update_approximation (bool rebuild_flag)
  replaces the approximation data with dacellerator 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 update_approximation (const RealMatrix &samples, 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 dacellerator 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 append_approximation (const RealMatrix &samples, const IntResponseMap &resp_map, bool rebuild_flag)
  appends a matrix 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

- IntIntPair estimate_partition_bounds (int max_eval_concurrency)
  estimate the minimum and maximum partition sizes that can be utilized by this Model

- 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 (unsigned short tabular_format, 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_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

- unsigned short exportFormat
  file export format for 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 correc-
tion (if active) to the results.
Reimplemented from Model.

References DiscrepancyCorrection::active(), Response::active_set(), DataFitSurrModel::actualModel, Discrepan-
correction::apply(), SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface, SurrogateModel::asv_
 mapping(), DataFitSurrModel::build_approximation(), DataFitSurrModel::component_parallel_mode(), Discrepan-
correction::compute(), Model::compute_response(), Response::copy(), Model::current_response(), Model::current-
Response, Model::currentVariables, SurrogateModel::deltaCorr, Model::eval_tag_prefix(), DataFitSurrModel::eval-
TagPrefix, DataFitSurrModel::export_point(), SurrogateModel::force_rebuild(), Model::hierarchicalTagging, Interface-
::map(), Model::outputLevel, ActiveSet::request_vector(), SurrogateModel::response_mapping(), SurrogateModel-
::responseMode, DataFitSurrModel::surrEvalCntr, Response::update(), and DataFitSurrModel::update_actual-
_model().

void derived_asynch_compute_response ( const ActiveSet & set ) [protected], [virtual]

portion of async_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 re-
results for return in derived_synchronize() below).
Reimplemented from Model.

References DataFitSurrModel::actualModel, SurrogateModel::approxBuilds, DataFitSurrModel::approxInterface,
SurrogateModel::asv_mapping(), Model::async_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, Surrogate-
Model::force_rebuild(), Model::hierarchicalTagging, Interface::map(), SurrogateModel::rawVarsMap, ActiveSet-
::request_vector(), SurrogateModel::responseMode, SurrogateModel::surrIdMap, DataFitSurrModel::surrModel-
EvalCntr, 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, Surrogate-
Model::surrIdMap, SurrogateModel::surrResponseMap, Interface::synch(), Model::synchronize(), and Surrogate-
Model::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_n-
nowait() is designed for the general case where derived_async_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_async_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()`.

```cpp
void update_approximation ( const Variables & vars, const IntResponsePair & response, 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()`.

```cpp
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()`.

```cpp
void update_approximation ( const RealMatrix & samples, 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()`.

```cpp
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.

```cpp
void append_approximation ( const Variables & vars, const IntResponsePair & response, 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.

```cpp
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.

```cpp
void append_approximation ( const RealMatrix & samples, const IntResponseMap & resp_map, bool rebuild_flag ) [protected], [virtual]
```
appends a matrix 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.

```cpp
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().

```cpp
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.

```cpp
void import_points ( unsigned short tabular_format, 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().

```cpp
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().

```cpp
void manage_data_recastings( ) [private]
```

initialize manageRecast and recastFlags for data import/export

Constructor helper to manage model recastings for data import/export.

Referenced by DataFitSurrModel::DataFitSurrModel().

```cpp
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().
CHAPTER 13. CLASS DOCUMENTATION

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::dvr(), Model::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_async_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_async_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::estimate_partition_bounds(), DataFitSurrModel::fine_grained_evaluation_counts(), 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 datafaceRep*
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 DataInterface-Rep. 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
13.24. DATAMETHODREP CLASS REFERENCE

**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 idMethod 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 matrices 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 `trialType` 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
• int neighborOrder
  the `NEIGHBOR_ORDER` 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

- bool standardizedSpace
  use of standardized probability spaces for MCMC within Bayesian inference

- bool adaptPosteriorRefine
  flag indicating adaptive refinement of the emulator in regions of high posterior probability

- bool logitTransform
  flag indicating user activation of logit transform option within QUESO

- String proposalCovType
  the type of proposal covariance: user, derivatives, or prior

- int proposalCovUpdates
  number of updates of the proposal covariance from computing the misfit Hessian using residual values and derivatives

- String proposalCovInputType
  the format of proposal covariance input: diagonal or matrix

- RealVector proposalCovData
  raw list of real data for the proposal covariance

- String proposalCovFile
  file from which to read proposal covariance in diagonal or matrix format

- 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 Gelman-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
• unsigned short pstudyFileFormat
  tabular format for the parameter study points file
• 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
• unsigned short approxImportFormat
  tabular format for approximation import
• bool approxImportActive
  whether to import active variables only
• String approxExportFile
  the file name for point export in surrogate-based methods
• unsigned short approxExportFormat
  tabular format for approximation export
Private Member Functions

- DataMethodRep ()
  \textit{constructor}
- \sim DataMethodRep ()
  \textit{destructor}
- void write (std::ostream &s) const
  \textit{write a DataInterfaceRep object to an std::ostream}
- void read (MPIUnpackBuffer &s)
  \textit{read a DataInterfaceRep object from a packed MPI buffer}
- void write (MPIPackBuffer &s) const
  \textit{write a DataInterfaceRep object to a packed MPI buffer}

Private Attributes

- int referenceCount
  \textit{number of handle objects sharing this dataMethodRep}

Friends

- class DataMethod
  \textit{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 ()
  \textit{constructor}
- DataModel (const DataModel &)
  \textit{copy constructor}
- \sim DataModel ()
  \textit{destructor}
- DataModel & operator= (const DataModel &)

### 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 idModel 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

- **unsigned short approxImportFormat**

  tabular format for the point import file

- **bool approxImportActive**

  whether to import active variables only

- **String approxExportFile**

  the file name from the export_points_file specification in ModelSurrG

- **unsigned short approxExportFormat**

  tabular format for the point export file
- String `approxExportModelFile`
  
  the file name from the `export.model_file` specification in `ModelSurrG`

- short `approxCorrectionType`
  
  correction type for global and hierarchical approximations: `NO_CORRECTION`, `ADDITIONAL_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*

- bool **piecewiseDecomp**
  
  *whether piecewise decomposition is enabled*

- String **decompCellType**
  
  *type of local cell of piecewise decomp*

- int **decompSupportLayers**
  
  *number of support layers for each local basis function*

- bool **decompDiscontDetect**
  
  *whether discontinuity detection is enabled*

- Real **discontJumpThresh**
  
  *function value (jump) threshold for discontinuity detection in piecewise decomp*

- Real **discontGradThresh**
  
  *gradient threshold for discontinuity detection in piecewise decomp*

- 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*

- unsigned short **approxChallengeFormat**
  
  *tabular format of the challenge data file*

- 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 \_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 numLeastSqTerms
  number of least squares terms (from the num_least_squares_terms specification in RespFnLS)
- 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 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 numScalarLeastSqTerms
  scalar_calibration_terms: number of calibration terms which are scalar
- size_t numScalarNonlinearIneqConstraints
  number of scalar nonlinear inequality constraints (from the num_scalar_nonlinear_inequality_constraints specification in RespFnOpt)
- size_t numScalarNonlinearEqConstraints
  number of scalar nonlinear equality constraints (from the num_scalar_nonlinear_equality_constraints specification in RespFnOpt)
- size_t numScalarResponseFunctions
  scalar_responses: number of response functions which are scalar
• size_t numFieldObjectiveFunctions
  field_objectives: number of objective functions which are field-valued
• size_t numFieldLeastSqTerms
  field_calibration_terms: number of calibration terms which are field-valued
• size_t numFieldNonlinearIneqConstraints
  number of field nonlinear inequality constraints (from the num_scalar_nonlinear_inequality_constraints specification in RespFnOpt)
• size_t numFieldNonlinearEqConstraints
  number of field nonlinear equality constraints (from the num_scalar_nonlinear_equality_constraints specification in RespFnOpt)
• 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)
• bool calibrationDataFlag
whether calibration data was specified

- 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

- RealVector expConfigVars
  list of num_experiments x num_config_vars configuration variable values

- bool interpolateFlag
  whether one should interpolate between the experiment and simulation field data

- RealVector expObservations
  list of num_calibration_terms observation data

- RealVector expStdDeviations
  list of 1 or num_calibration_terms observation standard deviations

- String scalarDataFileName
  name of experimental data file containing response data (with optional state variable and sigma data) to read

- unsigned short scalarDataFormat
  tabular format of the scalar data file

- 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 \texttt{fd\_hessian-step\_size} specification in \texttt{RespHessNum} and \texttt{RespHessMixed})

- \texttt{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

- \texttt{IntSet idNumericalGrads}
  mixed gradient numerical identifiers (from the \texttt{id\_numerical\_gradients} specification in \texttt{RespGradMixed})

- \texttt{IntSet idAnalyticGrads}
  mixed gradient analytic identifiers (from the \texttt{id\_analytic\_gradients} specification in \texttt{RespGradMixed})

- \texttt{IntSet idNumericalHessians}
  mixed Hessian numerical identifiers (from the \texttt{id\_numerical\_hessians} specification in \texttt{RespHessMixed})

- \texttt{IntSet idQuasiHessians}
  mixed Hessian quasi identifiers (from the \texttt{id\_quasi\_hessians} specification in \texttt{RespHessMixed})

- \texttt{IntSet idAnalyticHessians}
  mixed Hessian analytic identifiers (from the \texttt{id\_analytic\_hessians} specification in \texttt{RespHessMixed})

- \texttt{IntVector fieldLengths}
  number of entries in each field

- \texttt{IntVector numCoordsPerField}
  number of coordinates per field

- \texttt{RealVector coordList}
  values of coordinates per field

- \texttt{String coordDataFileName}
  data file which contains the values of the simulation coordinates per field

- \texttt{bool readFieldCoords}
  Field data related storage: whether to read experimental field coordinates.

- \texttt{StringArray varianceType}
  Array which specifies the sigma type per response (none, one constant value, one per response (vector) or a full covariance matrix

**Private Member Functions**

- \texttt{DataResponsesRep ()}
  constructor

- \texttt{\sim DataResponsesRep ()}
  destructor

- \texttt{void write (std::ostream \&s) const}
  write a \texttt{DataResponsesRep} object to an \texttt{std::ostream}

- \texttt{void read (MPIUnpackBuffer \&s)}
  read a \texttt{DataResponsesRep} object from a packed MPI buffer

- \texttt{void write (MPIPackBuffer \&s) const}
  write a \texttt{DataResponsesRep} object to a packed MPI buffer
13.29. **DataVariables Class Reference**

**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
13.30. DATA VARIABLES REP CLASS REFERENCE

- 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 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 specification in VarDV)
- StringArray discreteDesignSetStrVars
  initial values for the discrete design variables defined by a string set (from the discrete_design_set string specification in VarDV)
- RealVector discreteDesignSetRealVars
  initial values for the discrete design variables defined by a real set (from the discrete_design_set real 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)

• RealMatrixArray discreteDesignSetIntAdj
  Adjacency matrices for each of the discrete design variables defined by an integer set (from the discrete_design_set integer categorical adjacency specification in VarDV).

• RealMatrixArray discreteDesignSetStrAdj
  Adjacency matrices for each of the discrete design variables defined by a string set (from the discrete_design_set string adjacency specification in VarDV).

• RealMatrixArray discreteDesignSetRealAdj
  Adjacency matrices for each of the discrete design variables defined by a real set (from the discrete_design_set real categorical adjacency specification in VarDV).

• 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_factorsspecification 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 triangularUncLowerBnds
  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 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)
13.30. DATAVARIABLESREP CLASS REFERENCE

- **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
13.30. DATAVARIABLESREP CLASS REFERENCE

- 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 VarCEUV_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 discreteIntervalUncIntCat`
    - 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)
  - `RealMapArray 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_real_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 discreteDesignSetIntLowerBnds
  discrete design integer set lower bounds inferred from set values
- StringArray discreteDesignSetStrLowerBnds
  discrete design string set lower bounds inferred from set values
- StringArray discreteDesignSetStrUpperBnds
  discrete design string 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, tvu_lower_bounds, and buv_lower_bounds specifications in VarAUV, and derived for gamma, gumbel, 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, tvu_upper_bounds, and buv_upper_bounds specifications in VarAUV, and derived for gamma, gumbel, weibull and histogram bin specifications)
- StringArray continuousAleatoryUncLabels
  labels for all continuous aleatory uncertain variables (collected from nuv_descriptors, lnuv_descriptors, uuv_descriptors, luuv_descriptors, tvu_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:

```
  Iterator
    
  Analyzer
    
  PStudyDACE
    
  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*

- **boost::shared_ptr< DDaceSamplerBase > create_sampler (Model &model)**
  
  *create a DDACE sampler*

- **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(), D-DACEDesignCompExp::compute_main_effects(), DDACEDesignCompExp::create_sampler(), Iterator::iterated-Model, DDACEDesignCompExp::mainEffectsFlag, Analyzer::post_run(), PStudyDACE::pStudyDACESensGlobal, Iterator::subIteratorFlag, DDACEDesignCompExp::symbolMapping, and PStudyDACE::varBasedDecompFlag.

```cpp
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.

```cpp
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, DDACEDesignCompExp::numSymbols, and Iterator::submethod_enum_to_string().

Referenced by DDACEDesignCompExp::DDACEDesignCompExp(), DDACEDesignCompExp::get_parameter_sets(), and DDACEDesignCompExp::post_input().

```cpp
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**
  
  - 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_async** (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.
  
- **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.
  
- **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 discrete integer variables
- **`size_t`** `numADRV`  
  total number of discrete 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/ 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:

```c++
// API declaration
int sim(const Variables& vars, const ActiveSet& set, Response& response);
// use of API within derived_map_ac()
if (ac_name == "sin")
  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 in-
  dex, 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 addi-
  tive 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

```cpp
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(),
13.34. DOTOptimizer Class Reference

Wrapper class for the DOT optimization library.

Inheritance diagram for DOTOptimizer:

```
  Iterator
      ↓
  Minimizer
      ↓
  Optimizer
      ↓
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
13.34. DOTOPTIMIZER CLASS REFERENCE

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 dotFDInfo [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.
13.35. **JEGAOMIZER::DRIVER CLASS REFERENCE**

**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**

<table>
<thead>
<tr>
<th>probConfig</th>
<th>The definition of the problem to be solved by this Driver whenever ExecuteAlgorithm is called.</th>
</tr>
</thead>
</table>

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**

<table>
<thead>
<tr>
<th>algConfig</th>
<th>The fully loaded configuration object containing the database of parameters for the algorithm to be run on the known problem.</th>
</tr>
</thead>
</table>

**Returns**

The fully configured and loaded GA ready to be run using the PerformIterations method.

Referenced by JEGAOMizer::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 destroying 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:

```
Iterator
  |
---Minimizer
  |
SurrBasedMinimizer
  |
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 improvement 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 (ParLevIter pl_iter)**
  specialization of init comms due to varied use of the original model
- **void derived_set_communicators (ParLevIter pl_iter)**
  specialization of init comms due to varied use of the original model
- **void derived_free_communicators (ParLevIter 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 ]
13.37. **EFFICIENTSUBSPACEMETHOD CLASS REFERENCE**

- **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**

```cpp
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.

```cpp
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().

```cpp
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().

```cpp
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().

```cpp
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
- IntIntPair **estimate_partition_bounds**()
  estimate the minimum and maximum partition sizes that can be utilized by this Iterator
- 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 \texttt{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:

\begin{center}
\begin{tikzpicture}
\node (env) at (0,0) {Environment};
\node (exe) at (-2,-1) {ExecutableEnvironment};
\node (lib) at (2,-1) {LibraryEnvironment};
\draw (env) -- (exe);
\draw (env) -- (lib);
\end{tikzpicture}
\end{center}

Public Member Functions

• \texttt{Environment ()}

\textit{default constructor: empty envelope}

• \texttt{Environment (int argc, char \*argv[ ])}

\textit{envelope constructor for \texttt{ExecutableEnvironment} letter}

• \texttt{Environment (ProgramOptions prog_opts)}

• \texttt{Environment (MPI_Comm dakota_mpi_comm, ProgramOptions prog_opts=ProgramOptions())}

• \texttt{Environment (const String \&env_type)}

\textit{envelope constructor for letter type identified by String}

• \texttt{Environment (const Environment \&env)}

\textit{copy constructor}

• virtual \texttt{\~Environment ()}

\textit{destructor}

• \texttt{Environment operator= (const Environment \&env)}

\textit{assignment operator}

• virtual void \texttt{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, beast_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
CHAPTER 13. CLASS DOCUMENTATION

- **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 )
evelope constructor for letter type identified 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::destruc(), 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.
CHAPTER 13. CLASS DOCUMENTATION

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


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.
13.40. NOMADOPTIMIZER::EVALUATOR CLASS REFERENCE

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
13.40.3 Member Function Documentation

```c++
bool eval ( 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>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.</td>
</tr>
<tr>
<td>h_max</td>
<td>Current value of the barrier parameter. Not used in this implementation.</td>
</tr>
<tr>
<td>count_eval</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

```cpp
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>from</code></td>
<td>The Design class object from which to extract the discrete design variable values.</td>
</tr>
<tr>
<td><code>intoDiscInt</code></td>
<td>The vector into which to place the extracted discrete integer values.</td>
</tr>
<tr>
<td><code>intoDiscReal</code></td>
<td>The vector into which to place the extracted discrete real values.</td>
</tr>
<tr>
<td><code>intoCont</code></td>
<td>The vector into which to place the extracted continuous values.</td>
</tr>
</tbody>
</table>

**References**

JEGAOptimizer::Evaluator::model, Model::cv(), Model::discrete_int_sets(), Model::div(), and Model::drv().

```cpp
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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>from</code></td>
<td>The vector of responses to install into <code>into</code>.</td>
</tr>
<tr>
<td><code>into</code></td>
<td>The Design to which the responses belong and into which they must be written.</td>
</tr>
</tbody>
</table>

```cpp
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.

```cpp
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

<table>
<thead>
<tr>
<th>group</th>
<th>The group of Design class objects to be evaluated.</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>des</th>
<th>A Design that would be evaluated if this method worked.</th>
</tr>
</thead>
</table>

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]

Overridden 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:

```
Environment

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

Interpolation method for interpolating between experimental and model data. I need to work on inputs/outputs to this method. For now, this assumes interpolation of functional data.

Public Member Functions

- **ExperimentData ()**
  
  *default constructor*

- **ExperimentData (const ProblemDescDB &prob_desc_db, const SharedResponseData &srd, short output_level)**

  *typical DB-based constructor*

- **ExperimentData (size_t num_experiments, size_t num_config_vars, const boost::filesystem::path &data_prefix, const SharedResponseData &srd, const StringArray &variance_types, short output_level, std::string scalarDataFilename="")**

  *temporary? constructor for testing*

- void **load_data (const std::string &context_message, bool calc_sigma_from_data)**

  *Load experiments from data files (simple scalar or field)*

- size_t **num_experiments () const**

  *retrieve the number of experiments*

- size_t **num_total_exppoints () const**

  *retrieve the total number of experimental data points over all experiments*

- size_t **num_scalars () const**

  *retrieve the number of scalars (applies to all experiments)*

- size_t **num_fields () const**

  *retrieve the number of fields (applies to all experiments)*

- const RealVector & **config_vars (size_t experiment)**

  *retrieve the vector of configuration variables for the given experiment number*

- const RealVector & **all_data (size_t experiment)**

  *return contiguous vector of all data (scalar, followed by field) for the specified experiment*

- void **per_exp_length (IntVector &per_length) const**

  *return the individual sizes of the experimental data lengths (all function values, scalar and field)*

- const IntVector & **field_lengths (size_t experiment) const**

  *return the field lengths for specified experiment index*
• **Real scalar data** (size_t response, size_t experiment)
  retrieve the data value for the given response, for the given experiment

• **Real scalar sigma_est** (size_t response)
  retrieve the standard deviation value for the given response, estimated over all responses

• **RealVector field_data_view** (size_t response, size_t experiment)
  retrieve a view of the field data for the given response, for the given experiment

• **RealMatrix field_coords_view** (size_t response, size_t experiment)
  retrieve a view of the field data coordinates for the given response, for the given experiment

• **bool variance_type_active** (short variance_type)
  whether the specified variance type (enum value) is present and active

• **bool variance_active()**
  whether any variance type is active

• **Real apply_covariance** (const RealVector &residuals, size_t experiment)
  apply the covariance responses to compute the triple product v' = inv(C) * v for the given experiment

• **void apply_covariance_inv_sqrt** (const RealVector &residuals, size_t experiment, RealVector &weighted_residuals)
  apply inverse sqrt of the covariance to compute weighted residuals

• **void apply_covariance_inv_sqrt** (const RealMatrix &gradients, size_t experiment, RealMatrix &weighted_gradients)
  apply inverse sqrt of the covariance to compute weighted gradients

• **void apply_covariance_inv_sqrt** (const RealSymMatrixArray &hessians, size_t experiment, RealSymMatrixArray &weighted_hessians)
  apply inverse sqrt of the covariance to compute weighted Hessians

• **void get_main_diagonal** (RealVector &diagonal, size_t experiment) const
  return a (copy) vector containing the main diagonal entries of a specified experimental covariance matrix

• **void form_residuals** (const Response &sim_resp, const ShortArray &total_asv, Response &residual_resp)
  form residuals for all experiments, interpolating if necessary

• **void form_residuals** (const Response &sim_resp, size_t exp_num, const ShortArray &total_asv, size_t residual_resp_offset, Response &residual_resp)
  form residuals for an individual experiment, interpolating if necessary

• **void form_residuals_deprecated** (const Response &sim_resp, size_t experiment, RealVector &residuals)
  Remove once BAYES classes are cleaned up to use new Minimizer::data_difference_core functions.

• **void recover_model** (size_t num_pri_fns, RealVector &model_fns) const
  recover original model from the first experiment block in a full set of residuals; works in no interpolation case only (sizes same)

• **bool interpolate_flag()**
  flag for interpolation. If 0, no interpolation. If 1, interpolate.

• **void interpolate_simulation_data** (const Response &sim_resp, size_t exp_num, const ShortArray &total_asv, size_t exp_offset, Response &interp_resp)
  Interpolate simulation data (values, gradients and hessians) onto the coordinates of the experimental data.

• **ShortArray determine_active_request** (const Response &resid_resp, bool interrogate_field_data)
  Perform check on the active request vector to make sure it is amenable to interpolation of simulation data and application of apply covariance.

• **void scale_residuals** (Response &residual_response, ShortArray &total_asv)
Apply the experiment data covariance to the residual data.

- void **build_hessian_of_sum_square_residuals** (const Response &resp, RealSymMatrix &ssr_hessian)
  
  Build the hessian of the ssr from residuals, function gradients and function hessians based on the response’s active set request vector.

- void **build_hessian_of_sum_square_residuals** (const Response &resp, const ShortArray &asrv, RealSymMatrix &ssr_hessian)
  
  Build the hessian of the ssr from residuals, function gradients and function hessians using the passed active set request vector (overrides the response’s request vector).

- void **build_hessian_of_sum_square_residuals_from_response** (const Response &resp, const ShortArray &asrv, int exp_ind, RealSymMatrix &ssr_hessian)
  
  Update the hessian of ssr with the values from the hessian associated with a single experiment.

- void **build_hessian_of_sum_square_residuals_from_function_data** (const RealSymMatrixArray &func_hessians, const RealMatrix &func_gradients, const RealVector &residuals, RealSymMatrix &ssr_hessian, const ShortArray &asrv)
  
  Construct the hessian of the sum of squares of residuals.

**Private Member Functions**

- void **initialize** (const StringArray &variance_types, const SharedResponseData &srd)
  
  Shared body of constructor initialization.

- void **parse_sigma_types** (const StringArray &sigma_types)
  
  Parse user-provided sigma type strings and populate enums.

- void **load_experiment** (size_t exp_index, std::ifstream &scalar_data_stream, size_t num_field_sigma_matrices, size_t num_field_sigma_diagonals, size_t num_field_sigma_scalars, size_t num_field_sigma_none, Response &exp.resp)
  
  Load a single experiment exp_index into exp.resp.

- void **read_scalar_sigma** (std::ifstream &scalar_data_stream, RealVector &sigma_scalars, IntVector &scalar_map_indices)
  
  Read or default populate the scalar sigma.

- RealVector **residuals_view** (const RealVector &residuals, size_t experiment)
  
  Return a view (to allowing updating in place) of the residuals associated with a given experiment, from a vector containing residuals from all experiments.

- RealMatrix **gradients_view** (const RealMatrix &gradients, size_t experiment)
  
  Return a view (to allowing updating in place) of the gradients associated with a given experiment, from a matrix containing gradients from all experiments.

- RealSymMatrixArray **hessians_view** (const RealSymMatrixArray &hessians, size_t experiment)
  
  Return a view (to allowing updating in place) of the hessians associated with a given experiment, from an array containing the hessians from all experiments.

**Private Attributes**

- bool **calibrationDataFlag**
  
  Whether the user specified a calibration data block.

- size_t **numExperiments**
  
  The total number of experiments.

- size_t **numConfigVars**
number of configuration (state) variables to read for each experiment

- UShortVararray varianceTypes
  type of variance specified for each variable, one per response group; empty varianceType indicates none specified by user

- boost::filesystem::path dataPathPrefix
  path to prepend to any data file names

- String scalarDataFilename
  the user-specified scalar data filename

- unsigned short scalarDataFormat
  tabular format of the simple scalar data file; supports TABULAR_NONE, TABULAR_HEADER, TABULAR_EVAL_ID, TABULAR_EXPER_ANNOT

- size_t scalarSigmaPerRow
  number of sigma values to read from each row in simple data file format (calculated from variance types strings

- bool readFieldCoords
  whether to read coordinate data files for fields

- SharedResponseData simulationSRD
  archived shared data for use in sizing fields, total functions (historically we read all functions, including constraints, which might not be correct)

- bool interpolateFlag
  flag for interpolation.

- short outputLevel
  output verbosity level

- RealVector estimated_sigmas
  vector of estimated sigma values to be used in Bayesian calibration

- std::vector< Response > allExperiments
  Vector of numExperiments ExperimentResponses, holding the observed data and error (sigma/covariance) for each experiment.

- std::vector< RealVector > allConfigVars
  Vector of numExperiments configurations at which data were gathered; empty if no configurations specified.

- IntVector experimentLengths
  Length of each experiment.

- IntVector expOffsets
  function index offsets for individual experiment data sets

13.44.1 Detailed Description

Interpolation method for interpolating between experimental and model data. I need to work on inputs/outputs to this method. For now, this assumes interpolation of functional data.

As Brian suggested, this class has the experimental data (coordinates and RealVectorArray interpolated-Results; 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 Member Function Documentation

#### void recover_model ( size_t num_pri_fns, RealVector & best_fns ) const

recover original model from the first experiment block in a full set of residuals; works in no interpolation case only (sizes same)

Add the data back to the residual to recover the model, for use in surrogated-based LSQ where DB lookup will fail (need approx eval DB). best_fns contains primary and secondary responses.

References Dakota::abort_handler(), ExperimentData::allExperiments, Response::function_value(), ExperimentData::interpolateFlag, and Response::num_functions().

Referenced by LeastSq::post_run().

#### void build_hessian_of_sum_square_residuals_from_function_data ( const RealSymMatrixArray & func_hessians, const RealMatrix & func_gradients, const RealVector & residuals, RealSymMatrix & ssr_hessian, const ShortArray & asrv )

Construct the hessian of the sum of squares of residuals.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>func_hessians</td>
<td>A list of matrices containing the Hessians of the function elements in the residual vector</td>
</tr>
<tr>
<td>func_gradients</td>
<td>A matrix containing the gradients of the residual vector</td>
</tr>
<tr>
<td>residuals</td>
<td>A vector of residuals (mismatch between experimental data and the corresponding function values</td>
</tr>
<tr>
<td>asrv</td>
<td>The active set request vector</td>
</tr>
</tbody>
</table>

Referenced by ExperimentData::build_hessian_of_sum_square_residuals_from_response().

#### void parse_sigma_types ( const StringArray & sigma_types ) [private]

parse user-provided sigma type strings and populate enums

Validate user-provided sigma specification. User can specify 0, 1, or num_response_groups sigmas. If specified, sigma types must be the same for all scalar responses.

References Dakota::abort_handler(), SharedResponseData::num_response_groups(), SharedResponseData::num_scalar_responses(), ExperimentData::scalarDataFilename, ExperimentData::scalarSigmaPerRow, ExperimentData::simulationSRD, and ExperimentData::varianceTypes.

Referenced by ExperimentData::initialize().

#### void load_experiment ( size_t exp_index, std::ifstream & scalar_data_stream, size_t num_field_sigma_matrices, size_t num_field_sigma_diagonals, size_t num_field_sigma_scalars, size_t num_field_sigma_none, Response & exp_resp ) [private]

Load a single experiment exp_index into exp_resp.

Load an experiment from a mixture of legacy format data and field data format files

References ExperimentData::dataPathPrefix, Response::field_coords(), Response::field_group_labels(), ExperimentData::field_lengths(), Response::field_lengths(), Response::field_values(), Response::function_labels(), Response::function_value(), SharedResponseData::num_field_response_groups(), ExperimentData::num_fields(), SharedResponseData::num_scalar_responses(), ExperimentData::num_scalars(), ExperimentData::read_scalar_sigma(), ExperimentData::scalarDataFilename, ExperimentData::scalarSigmaPerRow, Response::set_full_covariance(), ExperimentData::simulationSRD, and ExperimentData::varianceTypes.

Referenced by ExperimentData::load_data().

The documentation for this class was generated from the following files:
13.45 ExperimentResponse Class Reference

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

Inheritance diagram for ExperimentResponse:

Response
  ↘
ExperimentResponse

Public Member Functions

- **ExperimentResponse ()**
  default constructor

- **ExperimentResponse (const Variables &vars, const ProblemDescDB &problem_db)**
  standard constructor built from problem description database

- **ExperimentResponse (const SharedResponseData &srd, const ActiveSet &set)**
  alternate constructor that shares a SharedResponseData instance

- **ExperimentResponse (const SharedResponseData &srd)**
  alternate constructor that shares a SharedResponseData instance

- **ExperimentResponse (const ActiveSet &set)**
  alternate constructor using limited data

- **~ExperimentResponse ()**
  destructor

- **void set_scalar_covariance (RealVector &scalars)**
  method to set the covariance matrix defined for ExperimentResponse

- **Real get_scalar_covariance (const int this_response)**
  method to get the covariance matrix defined for ExperimentResponse

- **void set_full_covariance (std::vector< RealMatrix > &matrices, std::vector< RealVector > &diagonals, RealVector &scalars, IntVector matrix_map_indices, IntVector diagonal_map_indices, IntVector scalar_map_indices)**
  method to set the full covariance matrices for ExperimentResponse

- **Real apply_covariance (const RealVector &residual)**
  method to compute the triple product $v' \cdot \text{inv}(C) \cdot v$.

- **void apply_covariance_inv_sqrt (const RealVector &residuals, RealVector &weighted_residuals)**
  method to compute $(v' \cdot \text{inv}(C) \cdot v)^{1/2}$, to compute weighted residual

- **void apply_covariance_inv_sqrt (const RealMatrix &gradients, RealMatrix &weighted_grads)**

- **void apply_covariance_inv_sqrt (const RealSymMatrixArray &hessians, RealSymMatrixArray &weighted_hessians)**

- **void get_covariance_diagonal (RealVector &diagonal) const**
Protected Member Functions

- void copy_rep(Response *source_resp_rep)

  Specialization of copy_rep; pulls base class data as well as derived specific data from the source rep into the this object.

Private Attributes

- ExperimentCovariance expDataCovariance

  sigma terms...

Additional Inherited Members

13.45.1 Detailed Description

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

The ExperimentResponse 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 (ExperimentResponse) actually contains the response data (functionValues, function-Gradients, functionHessians, etc.). The representation is hidden in that an instance of ExperimentResponse 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).

The documentation for this class was generated from the following files:

- ExperimentResponse.hpp
- ExperimentResponse.cpp

13.46 FileReadException Class Reference

base class for Dakota file read exceptions (to allow catching both tabular and general file truncation issues)

Inheritance diagram for FileReadException:

```
FileReadException
↓
ResultsFileError  TabularDataTruncated
```

Public Member Functions

- FileReadException (const std::string &msg)

13.46.1 Detailed Description

base class for Dakota file read exceptions (to allow catching both tabular and general file truncation issues)

The documentation for this class was generated from the following file:

- dakota_global defs.hpp
### 13.47 ForkApplicInterface Class Reference

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

Inheritance diagram for ForkApplicInterface:

![Inheritance Diagram](image)

#### 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 async evaluations

- void join_analysis_process_group (bool new_group)
  
  create (if new_group) and join the process group for async 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

void wait_evaluation (bool block_flag)
  process all available completions within the evaluation process group; if block_flag = true, wait for at least one completion

void 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

void 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

13.47.1 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
13.48  FSUDesignCompExp Class Reference

Wrapper class for the FSUDace QMC/CVT library.

Inheritance diagram for FSUDesignCompExp:

```
    iterator
     |     
    analyzer
     |     
PStudyDACE
     |     
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 getParameterSets (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

13.48.1 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 Tessellation (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.
13.48.2 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_ipv(), 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.

13.48.3 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.
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, FSUDesignComp-
   Exp::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

13.49 FunctionEvalFailure Class Reference

exception class for function evaluation failures
   Inherits runtime_error.

Public Member Functions
   • FunctionEvalFailure (const std::string &msg)

13.49.1 Detailed Description

exception class for function evaluation failures
   The documentation for this class was generated from the following file:
   • dakota.global_defs.hpp

13.50 GaussProcApproximation Class Reference

Derived approximation class for Gaussian Process implementation.
   Inheritance diagram for GaussProcApproximation:

   Approximation
   |                  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 num-Vars 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 NCSU Direct

- 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 component 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 &)
  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 component 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 * GInstance
  pointer to the active object instance used within the static evaluator

Additional Inherited Members

13.50.1 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.

13.50.2 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.
### 13.50.3 Member Function Documentation

```cpp
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().

### 13.50.4 Member Data Documentation

```cpp
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

### 13.51 GetLongOpt Class Reference

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

Inheritance diagram for GetLongOpt:

```
GetLongOpt        CommandLineHandler
                   \---------------------------
                     |                       |
                     v                       v
GetLongOpt          CommandLineHandler
```

#### 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 ()`
13.51. GETLONGOPT CLASS REFERENCE

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

13.51.1 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 "=".
13.51.2 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

13.51.3 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.


13.51.4 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.


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.


13.51.5 Member Function Documentation

**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 string. 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

13.52 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

13.52.1 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.

13.52.2 Member Function Documentation

void create_plots2d ( 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(), NonDLReliability::mean_value(), and NonDLReliability::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

13.53 GridApplicInterface Class Reference

Derived application interface class which spawns simulation codes using grid services such as Condor or Globus.
Inheritance diagram for GridApplicInterface:

```
  Interface
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>ApplicationInterface</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>ProcessApplicInterface</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>SysCallApplicInterface</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>GridApplicInterface</td>
</tr>
</tbody>
</table>
```

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)
CHAPTER 13. CLASS DOCUMENTATION

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

13.53.1 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.

13.53.2 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

13.54 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_async_computeresponse (const ActiveSet &set)
  portion of async_computeresponse() 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
- IntIntPair estimate_partition_bounds (int max_eval_concurrency)
  estimate the minimum and maximum partition sizes that can be utilized by this Model
• void derived_init_communicators (ParLevIter 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 (ParLevIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)
  set active parallel configuration within lowFidelityModel and highFidelityModel
• void derived_free_communicators (ParLevIter pl_iter, int max_eval_concurrency, bool recurse_flag=true)
  deallocate communicator partitions for the HierarchSurrModel (request forwarded to lowFidelityModel and high-
  FidelityModel)
• void serve_run (ParLevIter pl_iter, int max_eval_concurrency)
  Service lowFidelityModel and highFidelityModel job requests received from the master. Completes when a termi-
  nation 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 highFidelity-
  Model)
• 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 corre-
  sponding 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

13.54.1 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.

13.54.2 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, Surrogate-
  Model::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::delta-
  Corr, Model::eval_tag_prefix(), HierarchSurrModel::evalTagPrefix, SurrogateModel::force_rebuild(), Model::hierarchical-
  Tagging, HierarchSurrModel::hierModelEvalCntr, HierarchSurrModel::highFidelityModel, HierarchSurrModel-
  ::highFidRefResponse, HierarchSurrModel::lowFidelityModel, Model::outputLevel, ActiveSet::request_vector(),
  SurrogateModel::response_mapping(), SurrogateModel::responseMode, Response::update(), and HierarchSurr-
  Model::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(), Surrogate-
  Model::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, Active-
const IntResponseMap & derived_synchonize ( ) [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_synchonize() 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_synchonize_nowait(), HierarchSurrModel::highFidelityModel, HierarchSurrModel::highFidRefResponse, HierarchSurrModel::lowFidelityModel, Model::outputLevel, SurrogateModel::rawVarsMap, SurrogateModel::response mapping(), SurrogateModel::responseMode, SurrogateModel::surrResponseMap, Model::synchronize(), and SurrogateModel::truthIdMap.

const IntResponseMap & derived_synchonize_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_synchonize_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::derived_synchonize_nowait(), HierarchSurrModel::highFidelityModel, HierarchSurrModel::highFidRefResponse, HierarchSurrModel::lowFidelityModel, Model::outputLevel, SurrogateModel::rawVarsMap, SurrogateModel::response mapping(), SurrogateModel::responseMode, SurrogateModel::surrResponseMap, Model::synchronize_nowait(), and SurrogateModel::truthIdMap.

   Referenced by HierarchSurrModel::derived_synchonize().

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::hierModelEvalCntnr.

   The documentation for this class was generated from the following files:

   • HierarchSurrModel.hpp
   • HierarchSurrModel.cpp
13.55 Interface Class Reference

Base class for the interface class hierarchy.

Inheritance diagram for Interface:

```
  Interface
  |       |
  v       v
ApplicationInterface ApproximationInterface

  DirectApplicInterface
  |       |
  v       v
MatlabInterface PythonInterface

  ProcessApplicInterface
  |       |
  v       v
ProcessHandleApplicInterface SysCallApplicInterface

  TestDriverInterface
  |       |
  v       v
ParallelDirectApplicInterface

  SerialDirectApplicInterface
```

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 ()**
  - Sends 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コミュニケーションs` (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 `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_lbnds, const RealVector &c_ubnds, const IntVector &di_lbnds, const IntVector &di_ubnds, const RealVector &drl_bnds, const RealVector &dru_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 Incre=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 fine grain EvalCounters to true and initialize counters if needed
void init_evaluation_counters(size_t num_fns)
    initialize fine grained evaluation counters, sizing if needed
void set_evaluation_reference()
 CHAPTER 13. CLASS DOCUMENTATION

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).

- **short outputLevel**
  output verbosity level: \{SILENT, QUIET, NORMAL, VERBOSE, DEBUG\} OUTPUT

- **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 for detailed evaluation report

- **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**
CHAPTER 13. CLASS DOCUMENTATION


text content

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 (used in print_evaluation_summary() and derived direct interface classes); initialized in map() functions due to potential updates after construction

- bool multiProcEvalFlag
  Flag for multiprocessor evaluation partitions (evalComm)

- bool ieDedMasterFlag
  Flag for dedicated master partitioning at the iterator level

- 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 (conval) 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
13.55. INTERFACE CLASS REFERENCE

number of objects sharing interfaceRep

- ASL * asl
  pointer to an AMPL solver library (ASL) object

13.55.1 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.

13.55.2 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).
13.55.3 Member Function Documentation

**Interface operator= ( const Interface & interface_in )**


References Interface::interfaceRep, and Interface::referenceCount.

**void assign_rep ( Interface * interface_rep, bool ref_countincr = 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_countincr:

- ref_countincr = 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_countincr = 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::append_ifaceId, 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::coreMapping, 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().

### 13.55.4 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

### 13.56 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
• 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
• virtual IntIntPair estimate_partition_bounds ()
  estimate the minimum and maximum partition sizes that can be utilized by this Iterator
• 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 &d_index1, const SizetArray &ds_index1, const SizetArray &dr_index1, const ShortArray &c_target2, const ShortArray &d_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.

- **StrStrSize const 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)
13.56. **ITERATOR CLASS REFERENCE**

- **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.
CHAPTER 13. CLASS DOCUMENTATION

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

13.56.1 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.

13.56.2 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.

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, 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.

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.

13.56.3 Member Function Documentation

Iterator operator=(const Iterator & iterator)

assignment operator
  Assignment operator decrements referenceCount for old iteratorRep, assigns new iteratorRep, and increments
  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 PSUADEDesignCompExp.

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 SNNLOptimizer, NonD, SNLLLeastSq, Minimizer, Optimizer, and LeastSq.

References `Iterator::finalize_run()`, and `Iterator::iteratorRep`.

Referenced by Minimizer::finalize_run(), `Iterator::finalize_run()`, NonD::finalizerun(), `Iterator::run()`, and Seq-HybridMetaIterator::run_sequential_adaptive().

```cpp
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().

```cpp
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, NonDPOFDarts, NonDQUESOBayesCalibration, NonDLocalReliability, SeqHybridMetaIterator, NonDAdaptiveSampling, ConcurrentMetaIterator, NonDGPImpSampling, Optimizer, NonDAdaptImpSampling, NonDGlobalReliability, NonDIncrementLHSSampling, 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().

```cpp
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 simultaneously (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(), NCUSOptimizer::initialize(), NLPLPOptimizer::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::summary_OutputFlag.

Referenced by Iterator::run().

void assign_rep ( Iterator * iterator_rep, bool ref_count_inc = true )

replaces existing letter with a new one

Similar to the assignment operator, the assign_rep() function decrements referenceCount for the old iterator-Rep 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_inc:

- ref_count_inc = 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_inc = 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::initialize_fullspace_sampler(), NonDLocalInterval::method_recourse(), NonDLocalReliability::method_recourse(), NonDAdaptiveSampling::NonDAdaptiveSampling(), NonDBayesCalibration::NonDBayesCalibration(), NonDGlobalInterval::NonDGlobalInterval(), NonDGlobalReliability::NonDGlobalReliability(), NonDGPlmSampling::NonDGPlmSampling(), NonDGPSABayesCalibration::NonDGPSABayesCalibration(), NonDLHInterval::NonDLHInterval(), 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().
13.56. **ITERATOR CLASS REFERENCE**

**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().

13.56.4 **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(), CollabHybridMetaIterator::estimate_partition_bounds(), EmbedHybridMetaIterator::estimate_partition_bounds(), ConcurrentMetaIterator::estimate_partition_bounds(), SeqHybridMetaIterator::estimate_partition_bounds(), 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(), NonDGPSAmpSampling::NonDGPSAmpSampling(), NonDGPMSABayesCalibration::NonDGPMSABayesCalibration(), NonDInterval::NonDInterval(), NonDLocalInterval::NonDLocalInterval(), NonDLocalReliability::NonDLocalReliability(), NonDPolynomialChaos::NonDPolynomialChaos(), NonDQuadrature::NonDQuadrature(), NonDSparseGrid::NonDSparseGrid(), NonDStochCollocation::NonDStochCollocation(), NPS-
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(), NonDExpansion::derived_free_communicators(), NonDGlobalInterval::derived_free_communicators(), NonDAAdaptiveSampling::derived_free_communicators(), NonDGPImpSampling::derived_free_communicators(), NonDPolynomialChaos::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(), NonDExpansion::derived_init_communicators(), NonDGPImpSampling::derived_init_communicators(), NonDPolynomialChaos::derived_init_communicators(), NonDLocalReliability::derived_init_communicators(), Iterator::derived_init_communicators(), EfficientSubspaceMethod::derived_set_communicators(), SurrBasedMinimizer::derived_set_communicators(), NonDExpansion::derived_set_communicators(), NonDGlobalInterval::derived_set_communicators(), NonDExpansion::derived_set_communicators(), NonDLocalReliability::derived_set_communicators(), Iterator::derived_set_communicators(), NonDExpansion::derived_set_communicators(), 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(), NonDExpansion::NonDExpansion(), NonDGlobalInterval::NonDGlobalInterval(), NonDExpansion::NonDExpansion(), NonDQuadrature::NonDQuadrature(), NonDSparseGrid::NonDSparseGrid(), 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

### 13.57 IteratorScheduler Class Reference

This class encapsulates scheduling operations for concurrent sub-iteration within an outer level context (e.g., meta-iteration, nested models).
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 construct_sub_iterator** (ProblemDescDB &problem_db, Iterator &sub_iterator, Model &sub_model, const String &method_ptr, const String &method_name, const String &model_ptr)
  
  Instantiate sub_iterator on the current rank if not already constructed

- **IntIntPair configure** (ProblemDescDB &problem_db, Iterator &sub_iterator, Model &sub_model)
  
  Performs sufficient initialization to define partitioning controls (min and max processors per iterator server)

- **IntIntPair configure** (ProblemDescDB &problem_db, const String &method_string, Iterator &sub_iterator, Model &sub_model)
  
  Performs sufficient initialization to define partitioning controls (min and max processors per iterator server)

- **void partition** (int max_iterator_concurrency, IntIntPair &ppi_pr)
  
  Convenience function for initializing iterator communicators, setting parallel configuration attributes, and managing outputs and restart.

- **void init_iterator** (ProblemDescDB &problem_db, Iterator &sub_iterator, Model &sub_model)
  
  Invokes static version of this function with appropriate parallelism level

- **void init_iterator** (ProblemDescDB &problem_db, const String &method_string, Iterator &sub_iterator, Model &sub_model)
  
  Invokes static version of this function with appropriate parallelism level

- **void set_iterator** (Iterator &sub_iterator)
  
  Invokes static version of this function with appropriate parallelism level

- **void run_iterator** (Iterator &sub_iterator)
  
  Invokes static version of this function with appropriate parallelism level

- **void free_iterator** (Iterator &sub_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 Parallel-Level

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 &sub_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 &sub_iterator, Model &sub_model, ParLevLIter pl_iter)
convenience function for allocation of an iterator and (parallel) initialization of its comms

static void init_iterator (ProblemDescDB &problem_db, const String &method_string, Iterator &sub_iterator, Model &sub_model, ParLevLIter pl_iter)
convenience function for lightweight allocation of an iterator and (parallel) initialization of its comms

static void set_iterator (Iterator &sub_iterator, ParLevLIter pl_iter)
convenience function for setting comms prior to running an iterator

static void run_iterator (Iterator &sub_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 &sub_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
13.57. ITERATORSCHEDULER CLASS REFERENCE

- 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::miPLIter) 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)

13.57.1 Detailed Description

This class encapsulates scheduling operations for concurrent sub-iteration within an outer level context (e.g., meta-iteration, nested models).

In time, a Scheduler class hierarchy is envisioned, but for now, this class is not part of a hierarchy.

13.57.2 Constructor & Destructor Documentation

IteratorScheduler ( ParallelLibrary & parallelLib, 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().

13.57.3 Member Function Documentation

void init_iterator ( ProblemDescDB & problemDb, Iterator & sub_iterator, ParLevLIter plIter ) [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().

```c
void init_iterator ( ProblemDescDB & problem_db, Iterator & sub_iterator, Model & sub_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().

```c
void init_iterator ( ProblemDescDB & problem_db, const String & method_string, Iterator & sub_iterator, Model & sub_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 ProblemDescDB::get_iterator(), 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().

```c
void set_iterator ( Iterator & sub_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().

```c
void run_iterator ( Iterator & sub_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_view, static_schedule_iterators(), and IteratorScheduler::run_iterator(), and IteratorScheduler::serve_iterators().
void free_iterator ( Iterator & sub_iterator, ParLevLIter pl_iter ) [static]

Convenience function for deallocating commrs 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().

IntIntPair configure ( ProblemDescDB & problem_db, Iterator & sub_iterator, Model & sub_model )

Performs sufficient initialization to define partitioning controls (min and max processors per iterator server).

This is a convenience function for computing the minimum and maximum partition size prior to concurrent iterator partitioning.

References ProblemDescDB::get_iterator(), IteratorScheduler::schedPCIter, and ParallelLevel::server_communicator_rank().

Referenced by IteratorScheduler::configure(), ConcurrentMetaIterator::derived_init_communicators(), NestedModel::derived_init_communicators(), MetaIterator::estimate_by_name(), and MetaIterator::estimate_by_pointer().

IntIntPair configure ( ProblemDescDB & problem_db, const String & method_string, Iterator & sub_iterator, Model & sub_model )

Performs sufficient initialization to define partitioning controls (min and max processors per iterator server).

This is a convenience function for computing the minimum and maximum partition size prior to concurrent iterator partitioning.

References IteratorScheduler::configure(), ProblemDescDB::get_iterator(), IteratorScheduler::schedPCIter, and ParallelLevel::server_communicator_rank().

IntIntPair configure ( ProblemDescDB & problem_db, Iterator & sub_iterator )

Performs sufficient initialization to define partitioning controls (min and max processors per iterator server).

This is a convenience function for computing the minimum and maximum partition size prior to concurrent iterator partitioning.

References ParallelLibrary::bcast(), Iterator::estimate_partition_bounds(), ProblemDescDB::get_db_method_node(), ProblemDescDB::get_db_model_node(), IteratorScheduler::parallelLib, IteratorScheduler::schedPCIter, ParallelLevel::server_communicator_rank(), ParallelLevel::server_communicator_size(), ProblemDescDB::set_db_method_node(), ProblemDescDB::set_db_model_nodes(), and MPIPackBuffer::size().

void partition ( int max_iterator_concurrency, IntIntPair & ppi_pr )

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(), and IteratorScheduler::update().

Referenced by CollabHybridMetaIterator::derived_init_communicators(), EmbedHybridMetaIterator::derived_init_communicators(), ConcurrentMetaIterator::derived_init_communicators(), SeqHybridMetaIterator::derived_init_communicators(), and NestedModel::derived_init_communicators().
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

13.58 JEGAOptimizer Class Reference

A version of Dakota::Optimizer for instantiation of John Eddy’s Genetic Algorithms (JEGA).

Inheritance diagram for JEGAOptimizer:
13.58. **JEGAOPTIMIZER CLASS REFERENCE**

![Diagram](image)

**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 equivalent 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.
CHAPTER 13. CLASS DOCUMENTATION

• 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, const JEGA::Algorithms::GeneticAlgorithm &theGA, 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, const JEGA::Algorithms::GeneticAlgorithm &theGA, std::multimap<RealRealPair, JEGA::Utilities::Design *> &designSortMap)
  Retrieve 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, const JEGA::Algorithms::GeneticAlgorithm &theGA, std::multimap<RealRealPair, JEGA::Utilities::Design *> &designSortMap)
  Retrieve 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

13.58.1 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).

13.58.2 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

<table>
<thead>
<tr>
<th>problem_db</th>
<th>The Dakota::ProblemDescDB with information on how the algorithm controls should be set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>The Dakota::Model that will be used by this optimizer for problem information, etc.</td>
</tr>
</tbody>
</table>

References JEGAOptimizer::theEvalCreator, ProblemDescDB::get_int(), ProblemDescDB::get_short(), Iterator::iteratedModel, JEGAOptimizer::LoadTheParameterDatabase(), Iterator::maxEvalConcurrency, Iterator::methodName, Iterator::numFinalSolutions, and Iterator::probDescDB.

13.58.3 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 equivalent 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

<table>
<thead>
<tr>
<th>from</th>
<th>The JEGA Design class object from which to extract the variable and response information for Dakota.</th>
</tr>
</thead>
<tbody>
<tr>
<td>vars</td>
<td>The Dakota::Variables object into which to load the design variable values of from.</td>
</tr>
<tr>
<td>resp</td>
<td>The Dakota::Response object into which to load the objective function and constraint values of from.</td>
</tr>
</tbody>
</table>

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

\begin{tabular}{|l|l|}
\hline
\textit{pConfig} & The problem configuration object to load. \\
\hline
\end{tabular}

\textbf{void LoadTheDesignVariables ( JEGA::FrontEnd::ProblemConfig \& \textit{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

\begin{tabular}{|l|l|}
\hline
\textit{pConfig} & The problem configuration object to load. \\
\hline
\end{tabular}

References Model::continuous_lower_bounds().

\textbf{void LoadTheObjectiveFunctions ( JEGA::FrontEnd::ProblemConfig \& \textit{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

\begin{tabular}{|l|l|}
\hline
\textit{pConfig} & The problem configuration object to load. \\
\hline
\end{tabular}

\textbf{void LoadTheConstraints ( JEGA::FrontEnd::ProblemConfig \& \textit{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

\begin{tabular}{|l|l|}
\hline
\textit{pConfig} & The problem configuration object to load. \\
\hline
\end{tabular}

References Dakota::asstring(), Dakota::copy_row_vector(), and Model::nonlinear_ineq_constraint_lower_bounds().

\textbf{void GetBestSolutions ( const JEGA::Utilities::DesignOFSortSet \& \textit{from}, const JEGA::Algorithms::GeneticAlgorithm \& \textit{theGA}, std::multimap< RealRealPair, JEGA::Utilities::Design * > \& \textit{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

\begin{tabular}{|l|l|}
\hline
\textit{from} & The full set of designs returned by the solver. \\
\textit{theGA} & The GA used to generate this set; needed for its weights in the SO case, provided to both for consistency \\
\hline
\end{tabular}
13.58. JEGAOPTIMIZER CLASS REFERENCE

| 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.

```cpp
void GetBestMOSolutions ( const JEGA::Utilities::DesignOFSortSet & from, const JEGA::Algorithms::GeneticAlgorithm & theGA, std::multimap< RealRealPair, JEGA::Utilities::Design * > & designSortMap ) [protected]
```
Retrieves 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.

```cpp
void GetBestSOSolutions ( const JEGA::Utilities::DesignOFSortSet & from, const JEGA::Algorithms::GeneticAlgorithm & theGA, std::multimap< RealRealPair, JEGA::Utilities::Design * > & designSortMap ) [protected]
```
Retrieves 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. References Dakota::abort_handler().

```cpp
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.

```cpp
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::Execute-Algorithm 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().

```cpp
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) const [virtual]

Overridden to return true since JEGA algorithms can return multiple final points.

Returns

ture, 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.

13.58.4 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

13.59 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

13.59.1 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, NL-SSOLLeastSq, and SNLLLeastSq).

13.59.2 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, Minimizer::calibrationDataFlag, Variables::copy(), Model::current_variables(), Minimizer::data_transform_model(), Iterator::iteratedModel, Iterator::method_Name, LeastSq::numLeastSqTerms, Minimizer::numTotalCalibTerms, Minimizer::optimizationFlag, Model::primary_fn_type(), Minimizer::scale_model(), Minimizer::scaleFlag, LeastSq::weight_model(), and LeastSq::weightFlag.

13.59.3 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::calibrationDataFlag, Minimizer::initialize_run(), Iterator::iteratedModel, LeastSq::least-SqInstance, LeastSq::prevLSqInstance, Minimizer::scaleFlag, and Model::update_from_subordinate_model().

References 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).
transform variables back to inbound model, before any potential lookup
Reimplemented from Iterator.
Reimplemented in SNLLLeastSq.
References Dakota::abort_handler(), Response::active_set_request_vector(), Iterator::bestResponseArray, Iterator::bestVariablesArray, Minimizer::calibrationDataFlag, Variables::continuous_variables(), Response::copy(), Minimizer::cvScaleMultipliers, Minimizer::cvScaleOffsets, Minimizer::cvScaleTypes, Minimizer::expData, Response::function_value(), Response::function_values(), Response::function_values_view(), LeastSq::get_confidence_intervals(), ExperimentData::interpolate_flag(), Iterator::iteratedModel, Minimizer::local_recast_retrieve(), Minimizer::modify_s2n(), Minimizer::need_resp_trans_byvars(), LeastSq::numLeastSqTerms, Minimizer::numNonLinearConstraints, Minimizer::numUserPrimaryFns, Minimizer::post_run(), Model::primary_response_fn_weights(), Minimizer::primaryRespScaleFlag, ExperimentData::recover_model(), 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 point-
ers. 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, Minimizer::calibrationDataFlag, LeastSq::confBoundsLower, LeastSq::confBoundsUpper, Model::continuous_variable_labels(), Response::copy(), Model::current_response(), Minimizer::data_difference_core(), Dakota::data_pairs, Model::interface_id(), Iterator::iteratedModel, Dakota::lookup_by_val(), Model::num_functions(), Minimizer::numContinuousVars, Minimizer::numNonLinearConstraints, Minimizer::numTotalCalibTerms, Minimizer::numUserPrimaryFns, Minimizer::original_model(), Model::primary_response_fn_weights(), Minimizer::print_modelResp(), Minimizer::print_residuals(), 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, Minimizer::calibrationDataFlag, Model::compute_response, LeastSq::confBoundsLower, LeastSq::confBoundsUpper, Model::continuous_variables, Response::copy, Model::current_response, Minimizer::data_difference_core, Response::function_gradients, Iterator::iteratedModel, Minimizer::data_difference_core, LeastSq::confBoundsLower, LeastSq::confBoundsUpper, Model::continuous_variables, Response::copy, Model::current_response, Minimizer::data_difference_core, LeastSq::confBoundsLower, LeastSq::confBoundsUpper, Model::continuous_variables, Response::copy, Model::current_response, Minimizer::data_difference_core.

Referenced by LeastSq::post_run, 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::minimizerRecasts, Minimizer::numContinuousVars, LeastSq::numLeastSqTerms, Minimizer::numNonlinearConstraints, Minimizer::numNonlinearIneqConstraints, Iterator::outputLevel, LeastSq::primary_response_weighter, Model::primary_response_fn_weights, Model::primary_response_fn_weights, Minimizer::secondary_response_copier, and Model::subordinate_model.

Referenced by LeastSq::LeastSq.

void primary_response_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

13.60 LibraryEnvironment Class Reference

LibraryEnvironment corresponding to execution as an embedded library.

Inheritance diagram for LibraryEnvironment:

```
Environment

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

13.60.1 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.

13.60.2 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().

13.60.3 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

13.61 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.
13.61.1 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

13.62 MatchesWC Struct Reference

Predicate that returns true when the passed path matches the wildcard 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

13.62.1 Detailed Description

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

The documentation for this struct was generated from the following file:

- WorkdirHelper.hpp

13.63 MatlabInterface Class Reference

Inheritance diagram for MatlabInterface:

```
  Interface
  |    |
  v    v
ApplicationInterface
  |    |
  v    v
DirectApplicInterface
  |    |
  v    v
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 structure 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

13.63.1 Detailed Description

Specialization of DirectApplicInterface to link to Matlab analysis drivers. Includes convenience functions to map data to/from Matlab

13.63.2 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 derived 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_components.
  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

# 13.64 MetaIterator Class Reference

Base class for meta-iterators.

Inheritance diagram for MetaIterator:

```
MetaIterator
<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Iterator</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>MetaIterator</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
| CollabHybridMetaIterator
| ConcurrentMetaIterator
| EmbedHybridMetaIterator
| SeqHybridMetaIterator
```

## 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

13.64.1 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.

13.64.2 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

13.65 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 const 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

  *Return a shallow copy of the original model this Iterator was originally passed, optionally leaving recasts_left on top of it.*

• void **data_transform_model** ()
  
  *Wrap iteratedModel in a **RecastModel** that subtracts provided observed data from the primary response functions
  (variables and secondary responses are unchanged)*

• void genprimary_resp_map (const SharedResponseData &srd, Sizet2DArray &primary_resp_map_indices,
  BoolDequeArray &nonlinear_resp_map) const
  
  *compute the primary response map for a data transform **RecastModel***

• template<typename T>
  
  void expand_array (const SharedResponseData &srd, const T &submodel_array, T &recast_array) const
  
  *expand submodel_array by replicates to populate a pre-sized recast_array*

• 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_resps)
  
  *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 RealSymMatrix-
  Array &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.
- `Real sum_squared_residuals (size_t num_pri_fns, const RealVector &residuals, const RealVector &weights)
  return weighted sum of squared residuals
- `void print_residuals (size_t num_terms, const RealVector &best_terms, const RealVector &weights, size_t num_best, size_t best_index, std::ostream &s)
  print num_terms residuals and misfit for final results
- `void print_model_resp (size_t num_pri_fns, const RealVector &best_fns, size_t num_best, size_t best_index, std::ostream &s)
  print the original user model resp in the case of data transformations
- `void local_recast_retrieve (const Variables &vars, Response &response) const
  infers MOO/NLS solution from the solution of a single-objective optimizer
- `void data_difference_core (const Response &raw_response, Response &residual_response)
  Core of data difference, which doesn’t perform any output.

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
CHAPTER 13. CLASS DOCUMENTATION

- `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 inbound model; always initialize at Minimizer, even if overridden later
- `size_t numIterPrimaryFns`
  number of objective functions or least squares terms in iterator’s view, after transformations; 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
- bool `calibrationDataFlag`
  flag indicating whether user-supplied calibration 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 numTotalCalibTerms`
  number of total calibration terms (sum over experiments of number of experimental data per experiment, including field data)
- `bool applyCovariance`
  whether to scale residuals by covariance
- `bool matrixCovarianceActive`
  whether matrix-valued covariance is present
- `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
CHAPTER 13. CLASS DOCUMENTATION

- 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

- void copy_residuals (const RealVector &fn_vals, RealMatrix &fn_grad, const RealSymMatrixArray &fn_hess, size_t offset, size_t num_fns, Response &residual_response)
  
  Convenience function to map data from passed fn, grad, hess, into subset of residual_response, based on ASV.

- 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 responses 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
13.65. MINIMIZER CLASS REFERENCE

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)

13.65.1 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.

13.65.2 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().

13.65.3 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.

```
void data_transform_model ( ) [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.
References Dakota::abort_handler(), Iterator::activeSet, Minimizer::applyCovariance, Model::assign_rep(), Model::current_response(), Minimizer::expData, Minimizer::gen_primary_resp_map(), ProblemDescDB::get_sizet(), Iterator::iteratedModel, ExperimentData::load_data(), Minimizer::matrixCovarianceActive, Minimizer::minimizerRecasts, ExperimentData::num_total_exppoints(), Minimizer::numContinuousVars, Minimizer::numExperiments, Minimizer::numFunctions, Minimizer::numIterPrimaryFns, Minimizer::numNonlinearConstraints,
**MINIMIZER CLASS REFERENCE**

Minimizer::numNonlinearIneqConstraints, Minimizer::numTotalCalibTerms, Minimizer::numUserPrimaryFns, Iterator::outputLevel, Minimizer::primary_resp_diffencer(), Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), Iterator::probDescDB, ActiveSet::request_vector(), Minimizer::secondary_resp_copier(), Response::shared_data(), Model::subordinate_model(), and ExperimentData::variance_type_active.

Referenced by LeastSq::LeastSq(), and Optimizer::Optimizer().

```cpp
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, Minimizer::minimizerRecasts, 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().

```cpp
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().

```cpp
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().

```cpp
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 necessary

References Minimizer::varsScaleFlag.

Referenced by LeastSq::post_run(), Optimizer::post_run(), SNLLLeastSq::post_run(), and Minimizer::response_scaler_core().

```cpp
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, Active\_Set::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::numUserPrimary\_Fns, Iterator::output\_level, Active\_Set::request\_vector(), Minimizer::response\_Scale\_Multipliers, Minimizer::response\_Scale\_Offsets, Minimizer::response\_Scale\_Types, 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::numUserPrimary\_Fns.

Referenced by Surr\_Based\_Local\_Minimizer::approx\_subprob\_objective\_eval(), Surr\_Based\_Minimizer::augmented\_lagrangian\_merit(), Eff\_Global\_Minimizer::expected\_improvement(), Surr\_Based\_Minimizer::lagrangian\_merit(), Optimizer::objective\_reduction(), Surr\_Based\_Minimizer::penalty\_merit(), COLI\_Optimizer::post\_run(), Surr\_Based\_Minimizer::update\_filter(), and Surr\_Based\_Local\_Minimizer::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 user\_Primary\_Fns

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::optimization\_Flag.

`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.

```cpp
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

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.

```cpp
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(), and Minimizer::original_model().

Referenced by COLINOptimizer::post_run().

```cpp
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(), and Minimizer::original_model().

Referenced by COLINOptimizer::post_run().

```cpp
void local_recast_retrieve ( const Variables & vars, Response & response ) const [protected]
```

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. Always returns best data in the space of the original inbound Model.

References Response::active_set(), Dakota::data_pairs, Model::interface_id(), Iterator::iteratedModel, Dakota::lookup_by_val(), and Response::update().

Referenced by LeastSq::post_run(), and Optimizer::post_run().
void primary_responder 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 Response::active_set.request_vector(), Minimizer::data_difference_core(), Response::function_gradient_view(), Response::function_hessian(), Response::function_labels(), Response::function_value(), Response::function_values(), Minimizer::minimizerInstance, ExperimentData::num_total_exppoints(), Minimizer::numTotalCalibTerms, Minimizer::numUserPrimaryFns, 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_eq_constraint_coeffs(), Model::linear_ineq_constraint_coeffs(), Model::linear_ineq_constraint_upper_bounds(), Minimizer::linearEqScaleMultipliers, Minimizer::linearEqScaleOffsets, Minimizer::linearEqScaleTypes, Minimizer::linearEqScaleMultipliers, Minimizer::linearEqScaleOffsets, Minimizer::linearEqScaleTypes, Model::model_rep(), Minimizer::modify_n2s(), Model::nonlinear_eq_constraint_coeffs(), Model::nonlinear_eq_constraint_targets(), Model::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Minimizer::numContinuousVars, Minimizer::numFunctions, Minimizer::numLinearEqConstraints, Minimizer::numLinearEqConstraints, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearEqConstraints, 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_responder scaler (const Variables & native_vars, 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 responses 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, 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

13.66 MixedVarConstraints Class Reference

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

Inheritance diagram for MixedVarConstraints:

```
    Constraints
     ^
     |       ^
    |       |       
    MixeVarConstraints
```
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

13.66.1 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`).

13.66.2 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

13.67 MixedVariables Class Reference

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

Inheritance diagram for MixedVariables:
13.67. MIXEDVARIABLES CLASS REFERENCE

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

13.67.1 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)).

13.67.2 Constructor & Destructor Documentation

MixedVariables ( const ProblemDescDB &问题_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().
13.67.3 Member Function Documentation

```cpp
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::allDiscrete-IntVars, 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

13.68 Model Class Reference

Base class for the model class hierarchy.

Inheritance diagram for Model:

```
Model
+---- NestedModel
|    RecastModel
|    SingleModel
|    SurrogateModel
|    +---- DataFitSurrModel
|    +---- HierarchSurrModel
```

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, DataFit-
  SurrModel::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 append_approximation (const RealMatrix &samples, 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/ACTIONAL_MODEL/HF_MODEL/TRUTH_MODEL)].
virtual IntIntPair estimate_partition_bounds (int max_eval_concurrency)
    estimate the minimum and maximum partition sizes that can be utilized by this 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 (ParLevIter 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

- virtual bool db_lookup (const Variables &search_vars, const ActiveSet &search_set, Response &found_resp)
  search the eval database (during derivative estimation); derived may need to reimplement due to problem transformations (RecastModel); return true if found in DB

- 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
- `void epistemic_distribution_parameters` (const Pecos::EpistemicDistParams &edp)
  
  ```cpp
  set epistDistParams
  ```

- `Real continuous_probability_density()` const
  
  `compute a multivariate probability density from the marginals for the aleatory random variables within the current active continuous variables (must be uncorrelated)`

- `Real continuous_probability_density`(const RealVector &c_vars) const
  
  `compute a multivariate probability density from the marginals for the aleatory random variables within the provided active continuous variables (must be uncorrelated)`

- `Real continuous_probability_density`(size_t cv_index) const
  
  `compute a univariate probability density for a particular active continuous variable at the current value`

- `Real continuous_probability_density` (Real c_var, size_t cv_index) const
  
  `compute a univariate probability density for a particular active continuous variable at the provided value`

- `Real continuous_probability_density` (Real c_var, unsigned short cv_type, size_t dist_index) const
  
  `compute a univariate probability density for a particular active continuous variable at the provided value`

- `RealRealPair continuous_distribution_bounds`(size_t cv_index) const
  
  `return a set of distribution bounds for a particular index within the active continuous variables`

- `Real continuous_distribution_moment`(size_t cv_index, unsigned short int moment_id) const
  
  `return a distribution moment for a particular index within the active continuous 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 &id\_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 &id\_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 &id\_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)
13.68. MODEL CLASS REFERENCE

- 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

• size_t num_primary_fns () const
  return number of primary functions (total less nonlinear constraints)

• 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

- short primary_fn_type () const
  get the primary response function type (generic, objective, calibration)

- void primary_fn_type (short type)
  set the primary response function type, e.g., when recasting

- 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 Shared-
VariablesData &svd, const SharedResponseData &srd, 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 asyncEvalFlag and evaluationCapacity based on ie_pl settings
• void string_variable_max (const StringSetArray &ssa, size_t offset, Variables &vars)
  set the current value of each string variable offset + i to the longest string value found in the admissible string set ssa[i]
• void string_variable_max (const StringRealMapArray &srma, size_t offset, Variables &vars)
  set the current value of each string variable offset + i to the longest string value found in the admissible string map srma[i]

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
- ModellList modelList
  used to collect sub-models for \texttt{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 IntSet's, each containing the set of allowable integer 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 IntSet's, each containing the set of allowable integer 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)
13.68.1 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.

13.68.2 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 class reference**

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.

### 13.68.3 Member Function Documentation

**Model operator= (const Model & model)**


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().

**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(), Minimizer::original_model(), LeastSq::post_run(), COLINOptimizer::post_run(), Optimizer::primary_resp_reducer(), LeastSq::primary_resp_weighter(), LeastSq::print_results(), 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(), Least-Sq::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.
13.68. MODEL CLASS REFERENCE

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 asynchronous local operations for:

- NestedModels: a subIterator can support message passing parallelism, but not asynchronous local.
- DataFitSurrModels: while asynchronous evaluations 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::DataFitSurrModel(), Model::db_lookup(), Model::estimate_message_lengths(), SurrBasedLocalMinimizer::find_center_approm(), Model::interface_id(), RecastModel::interface_id(), Minimizer::local_recast_retrieve(), SurrBasedLocalMinimizer::minimize_surrogates(), SNLLLeastSq::post_run(), Analyzer::pre_output(), SurrBasedMinimizer::print_results(), LeastSq::print_results(), Optimizer::print_results(), SeqHybridMetaIterator::run_sequential(), Discrepancy-Correction::search_db(), Model::synchronize(), Model::synchronize_nowait(), Analyzer::update_best(), Concurrent-MetaIterator::update_local_results(), SeqHybridMetaIterator::update_local_results(), and NonDLocalReliability::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_idstr ) [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(), Data-
FitSurrModel::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(), N-
LPQLPOptimizer::initialize(), NonDLocalInterval::NonDLocalInterval(), NonDLocalReliability::NonDLocalReliability(),
SOLBase::SOLBase(), Model::subordinate_models(), and SurrBasedLocalMinimizer::SurrBasedLocalMinimizer().

void init_communicators ( ParLevLIter pliter, 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 structured 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(), NonDGlobal-
Interval::derived_init_communicators(), NonDAdaptImpSampling::derived_init_communicators(), NonDAdaptive-
Sampling::derived_init_communicators(), NonDGImpSampling::derived_init_communicators(), NonDPolynomial-
Chaos::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 structured 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.
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(), Model::aleatDistParams, Response::copy(), Variables::copy(), Model::currentResponse, Model::currentVariables, Variables::cv(), Model::discreteDesignSetStringValues, Model::discreteStateSetStringValues, Model::epistDistParams, Model::estimate_message_lengths(), Variables::icv(), Model::interface_id(), Model::messageLengths, Model::modelRep, ParallelLibrary::mpirun_flag(), Model::numFns, Model::parallelLib, MPIPackBuffer::reset(), MPIPackBuffer::size(), and Model::string_variable_max().

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 model-Rep 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(), NonDGPlmpSampling::NonDGPlmpSampling(), 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().

Real continuous_probability_density ( Real c_var, unsigned short cv_type, size_t dist_index ) const

compute a univariate probability density for a particular active continuous variable at the provided value

This overloaded version accepts a distribution index to avoid repeated invocations of find_index().

References Model::aleatDistParams, Model::continuous_probability_density(), and Model::modelRep.
CHAPTER 13. CLASS DOCUMENTATION

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(), DataFitSurrModel::estimate_partition_bounds(), 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
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 async_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::dbCaptureList, Model::dResponseList, Model::deltaList, ActiveSet::derivative_vector(), Dakota::find_index(), Response::function_gradients(), Response::function_values(), Model::icv(), Variables::inactive_continuous_variable_ids(), Model::inactive_continuous_variable_types(), Variables::inactive_continuous_variables(), Model::initialize_h(), Model::initialMapList, Model::intervalType, Model::numFns, Model::outputLevel, ActiveSet::request_vector(), Model::shortStep, and Model::update_response().

Referenced by Model::async_compute_response(), and Model::compute_response().

void update_response ( const Variables & vars, Response & new_response, const ShortArray & fd_grad_asv, const ShortArray & fd_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().

13.68.4 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(), SingleModel::estimate_partition-
    _bounds(), HierarchSurrModel::estimate_partition_bounds(), NestedModel::estimate_partition_bounds(), DataFit-
    SurrModel::estimate_partition_bounds(), Model::problem_description_db(), and NestedModel::update_sub_iterator().

    The documentation for this class was generated from the following files:
    
    • DakotaModel.hpp
    • DakotaModel.cpp

13.69  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

13.69.1 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

13.70 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**()
  
  Destructor.

- 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.
CHAPTER 13. CLASS DOCUMENTATION

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.

13.70.1 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

13.71 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.
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.

13.71.1 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

13.72 NCSUOptimizer Class Reference

Wrapper class for the NCSU DIRECT optimization library.

Inheritance diagram for NCSUOptimizer:

```
  NCSUOptimizer
      ^
     |  
    Minimizer
       ^
      |  
   Optimizer
     ^
    |  
  Iterator
```

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

13.72.1 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:

13.72.2 Constructor & Destructor Documentation

NCSUOptimizer ( ProblemDescDB & problemdb, 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_lbnds, const RealVector & var_ubnds, 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.
13.72.3 Member Function Documentation

```c
int objective_eval ( int * n, double c[], double l[], double u[], int * maxI, int * start, int * maxfunc, double free[], 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.

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, NCSU-Optimizer::ncsudirectInstance, Model::primary
`response`_fn_sense(), NCSUOptimizern::setUpType, Model::
synchronize(), and NCSUOptimizern::userObjectiveEval.

Referenced by NCSUOptimizern::find
`optimum`().

The documentation for this class was generated from the following files:

- NCSUOptimizern.hpp
- NCSUOptimizern.cpp

13.73 NestedModel Class Reference

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

Inheritance diagram for NestedModel:

```
Model
    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)
• intIntPair estimate_partition_bounds (int max_eval_concurrency)
  estimate the minimum and maximum partition sizes that can be utilized by this Model
• 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 optional interface (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
13.73. NESTEDMODEL CLASS REFERENCE

- 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

13.73.1 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.

13.73.2 Member Function Documentation

```c
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::iterator-CommSize, 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-
CHAPTER 13. CLASS DOCUMENTATION

void derived_async_compute_response ( const ActiveSet & set ) [protected], [virtual]

portion of async_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::synchronize().

short local_eval_synchronization ( ) [virtual]

return optionalInterface synchronization setting

Used in setting Model::asyncEvalFlag. subModel synchronization is used for setting asyncEvalFlag within subModel.

Reimplemented from Model.

References Interface::asynch_local_evaluation_concurrency(), Interface::interface_synchronization(), NestedModel::optInterfacePointer, and NestedModel::optionalInterface.

int local_eval_concurrency ( ) [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 [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 ( ParLevIter 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(), IteratorScheduler::configure(), Model::currentVariables, ProblemDescDB::get_db_method_node(), ProblemDescDB::get_db_model_node(), Interface::init_communicators(), IteratorScheduler::init_iterator(), Iterator::is_null(), IteratorScheduler::iterator_message_lengths(), IteratorScheduler::iteratorServerId, Model::messageLengths, IteratorScheduler::messagePass, Model::modelPCIter, IteratorScheduler::numIteratorServers, NestedModel::optInterfacePointer, NestedModel::optionalInterface, ParallelLibrary::parallel_configuration_iterator(), Model::parallelLib, IteratorScheduler::partition(), 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 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 \( \{g_l\} \leq \{g\} \leq \{g_u\} \)
\( \{a_l\} \leq [A]\{S\} \leq \{a_u\} \)
\( \{g\} = \{g_t\} \)
\( [A]\{S\} = \{a_t\} \)

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: \(\{g_l\} \leq \{g\} + [A]\{S\} \leq \{g_u\} \) 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().

13.73.3 Member Data Documentation

Model subModel [private]

the sub-model used in sub-iterator evaluations
CHAPTER 13. CLASS DOCUMENTATION

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::estimate_partition_bounds(), 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

13.74 NIDRProblemDescDB Class Reference

The derived input file database utilizing the new IDR parser.

Inheritance diagram for NIDRProblemDescDB:

```
ProblemDescDB
  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_lit)**
- **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_li)
• 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_nmint)
• 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_ubyte)
- **KWH** (method_augment_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** (model_utype)
- **KWH** (model_augment_utype)
- **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** (resp_utype)
- **KWH** (resp_augment_utype)
- **KWH** (env_int)
- **KWH** (env_start)
- **KWH** (env_str)
- **KWH** (env_strL)
- **KWH** (env_true)
- **KWH** (env_utype)
- **KWH** (env_augment_utype)
- **KWH** (var_Reallb)
- **KWH** (var_ReallUb)
- **KWH** (var_Intlb)
- **KWH** (var_categorical)
- **KWH** (var cauliflower)
- **KWH** (var_dauilbl)
• KWH (var_dauslbl)
• KWH (var_daurlbl)
• KWH (var_ceulbl)
• KWH (var_deulbl)
• 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_newarray)
• KWH (var_newivec)
• KWH (var_newrvec)
• KWH (var_iivec)
• 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

13.74.1 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.

13.74.2 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::tokenizer_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

### 13.75 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 \( r(x) \)

### 13.75.1 Detailed Description

Auxiliary information passed to calcr and calcj via ur.

The documentation for this struct was generated from the following file:

- NL2SOLLeastSq.cpp

### 13.76 NL2SOLLeastSq Class Reference

Wrapper class for the NL2SOL nonlinear least squares library.

Inheritance diagram for NL2SOLLeastSq:
CHAPTER 13. CLASS DOCUMENTATION

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`
  auxiliary 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 dltfde`
finite-diff step size for function differences for H
\(<\text{fd}\_\text{hessiann\_step\_size})

- int \text{mxcal}
  - function-evaluation limit \((\text{max}\_\text{function\_evaluations})\)
- int \text{mxiter}
  - iteration limit \((\text{max}\_\text{iterations})\)
- Real \text{rfctol}
  - relative fn convergence tolerance \((\text{convergence}\_\text{tolerance})\)
- Real \text{afctol}
  - absolute fn convergence tolerance \((\text{absolute}\_\text{conv}\_\text{tol})\)
- Real \text{xctol}
  - \(x\)-convergence tolerance \((\text{x\_conv\_tol})\)
- Real \text{setol}
  - singular convergence tolerance \((\text{singular}\_\text{conv}\_\text{tol})\)
- Real \text{lmmaxs}
  - radius for singular-convergence test \((\text{singular}\_\text{radius})\)
- Real \text{xftol}
  - false-convergence tolerance \((\text{false}\_\text{conv}\_\text{tol})\)
- int \text{covreq}
  - kind of covariance required \((\text{\&c}\ \text{covariance})\):
    \(<\ 1\ \text{or} -1\ ==\ >\ \text{sigma}\^2\ H^{-1}\ J^T J H^{-1} < 2\ \text{or} -2\ ==\ >\ \text{sigma}\^2\ 2\ H^{-1}\ < 3\ \text{or} -3\ ==\ >\ \text{sigma}\^2\ H^{-1}
    \<\ 1\ \text{or} 2\ ==\ >\ \text{use gradient diffs to estimate } H < -1\ \text{or} -2\ ==\ >\ \text{use function diffs to estimate } H < \text{default} = 0\ (\text{no covariance})\)
- int \text{rdreq}
  - whether to compute the regression diagnostic vector
\(<(\text{regression}\_\text{diagnostics})\)
- Real \text{fprec}
  - expected response function precision \((\text{function}\_\text{precision})\)
- Real \text{lmmax0}
  - initial trust-region radius \((\text{initial}\_\text{trust}\_\text{radius})\)

Static Private Attributes

- static \text{NL2SOLLeastSq} * \text{nl2solInstance}
  - pointer to the active object instance used within the static evaluator functions

Additional Inherited Members

13.76.1 Detailed Description

Wrapper class for the NL2SOL nonlinear least squares library.

The \text{NL2SOLLeastSq} class provides a wrapper for NL2SOL (TOMS Algorithm 573), in the updated form of Port Library routines \text{dn}\[fg\][l] from Bell Labs; see \url{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.
13.76.2 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.


Implements LeastSq.

References NL2SOLLeastSq::afctol, NL2SOLLeastSq::auxprt, Iterator::bestResponseArray, Iterator::bestVariablesArray, Minimizer::boundConstraintFlag, NL2SOLLeastSq::calcj(), NL2SOLLeastSq::caler(), Minimizer::calibrationDataFlag, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), Dakota::copy_data(), NL2SOLLeastSq::covreq, NL2SOLLeastSq::delta0, NL2SOLLeastSq::dltfdc, NL2SOLLeastSq::dltfdj, Minimizer::expData, NL2SOLLeastSq::fprec, Model::gradient_type(), ExperimentData::interpolate_flag(), Iterator::iteratedModel, NL2SOLLeastSq::lmax0, NL2SOLLeastSq::lmaxs, NL2SOLLeastSq::mxfcal, NL2SOLLeastSq::mxiter, NL2SOLLeastSq::nxolInstance, Minimizer::numContinuousVars, LeastSq::numLeastSqTerms, Minimizer::numUserPrimaryFns, LeastSq::numLeastSqTerms, Minimizer::numUserPrimaryFns, 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

13.77 NLPQLPOptimizer Class Reference

Wrapper class for the NLPQLP optimization library, Version 2.0.

Inheritance diagram for NLPQLPOptimizer:

```
  NLPQLPOptimizer
  |      |
  |      |    Optimizer
  |      |     |
  |      |     |    Minimizer
  |      |     |     |    Iterator
  |      |     |    |
  |      |    |
  |      |

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 approximation 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>M+AX, N>NMAX, or MNN2<>M+N+N+2. IFAIL = 7 : 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
  
  LACTIVE : The length LACTIVE 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
CHAPTER 13. CLASS DOCUMENTATION

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**

**13.77.1 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

\[
\begin{align*}
\text{minimize} & \quad F(X) \\
\text{subject to} & \quad G(J,X) = 0 , \ J=1,...,ME \\
& \quad G(J,X) \geq 0 , \ J=ME+1,...,M \\
& \quad XL \leq X \leq Xu
\end{align*}
\]

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
13.78 NLSSOLLeastSq Class Reference

Wrapper class for the NLSSOL nonlinear least squares library.

Inheritance diagram for NLSSOLLeastSq:

- Iterator
- Minimizer
- LeastSq
- SOLBase
- 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

13.78.1 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 N- LSSOLLeastSq’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.

13.78.2 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(), ProblemDesc -DB::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

13.79 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."

13.79.1 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:
13.80  NomadOptimizer Class Reference

Wrapper class for NOMAD Optimizer.

Inheritance diagram for NomadOptimizer:

```
  NomadOptimizer
  |     |
  |     |  Optimizer
  |     |  Minimizer
  |     |  Iterator
  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, NOMAD::Parameters &p)
  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
CHAPTER 13. CLASS DOCUMENTATION

Algorithm control parameters passed to NOMAD.

- int maxBlackBoxEvals
- int maxIterations
- Real epsilon
- Real vns
- std::string outputFormat

Output control parameters passed to NOMAD.

- std::string historyFile
- bool displayAll
- int numHops

Parameters needed for categorical neighbor construction.

- BitArray discreteSetIntCat
- BitArray discreteSetRealCat
- RealMatrixArray discreteSetIntAdj
- RealMatrixArray discreteSetRealAdj
- RealMatrixArray discreteSetStrAdj
- RealMatrixArray categoricalAdjacency
- NOMAD::Point initialPoint
- NOMAD::Point upperBound
- NOMAD::Point lowerBound

Parameters: constraintMapIndices

- std::vector<int> constraintMapIndices
  map from Dakota constraint number to Nomad constraint number

Multipliers: constraintMapMultipliers

- std::vector<double> constraintMapMultipliers
  multipliers for constraint transformations

Offsets: constraintMapOffsets

- std::vector<double> constraintMapOffsets
  offsets for constraint transformations

Additional Inherited Members

13.80.1 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.
13.80.2 Constructor & Destructor Documentation

NomadOptimizer ( ProblemDescDB & problem_db, Model & model )

Constructor.

NOMAD Optimizer Constructor

Parameters

<table>
<thead>
<tr>
<th>model</th>
<th>DAKOTA Model object</th>
</tr>
</thead>
</table>

References ProblemDescDB::get_bat(), ProblemDescDB::get_bool(), ProblemDescDB::get_int(), ProblemDescDB::get_real(), ProblemDescDB::get_rma(), ProblemDescDB::get_string(), NomadOptimizer::numHops, NomadOptimizer::outputFormat, Iterator::probDescDB, and NomadOptimizer::randomSeed.

13.80.3 Member Function Documentation

void load_parameters ( Model & model, NOMAD::Parameters & p ) [private]

Convenience function for Parameter loading.

This function takes the Parameters provided by the user in the DAKOTA model.

Parameters

<table>
<thead>
<tr>
<th>model</th>
<th>NOMAD Model object Variables for the stuff that must go in the parameters. Will be filled by calling load_parameters after the constructor to capture model recasts.</th>
</tr>
</thead>
</table>

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::discrete_int_lower_bounds(), Model::discrete_int_sets(), Model::discrete_int_upper_bounds(), Model::discrete_real_lower_bounds(), Model::discrete_real_upper_bounds(), Model::discrete_string_values(), Model::discrete_string_variables(), Model::discrete_set_int_values(), Model::discrete_set_int_variables(), Model::discrete_set_real_values(), Model::discrete_set_real_variables(), NomadOptimizer::initialPoint, Iterator::iteratedModel, NomadOptimizer::lowerBound, Model::nonlinear_eq_constraint_targets(), Model::nonlinear_ineq_constraint_lower_bounds(), Model::nonlinear_ineq_constraint_upper_bounds(), Minimizer::numDiscreteIntVars, Minimizer::numDiscreteRealVars, Minimizer::numDiscreteStringVars, NomadOptimizer::numNomadNonlinearIneqConstraints, Minimizer::numNonlinearEqConstraints, Minimizer::numNonlinearIneqConstraints, NomadOptimizer::numTotalVars, Dakota::set_value_to_index(), and NomadOptimizer::upperBound.

Referenced by NomadOptimizer::find_optimum().

The documentation for this class was generated from the following files:

- NomadOptimizer.hpp
- NomadOptimizer.cpp

13.81 NonD Class Reference

Base class for all nondeterministic 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 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 is supported by Nataf for given variable types

- void transform_correlations ()
  perform correlation warping for variable types supported by Nataf
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_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 \( X \rightarrow Z \rightarrow U \) 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-\(\cdot\)p (PROBABILITIES), z-\(\cdot\)beta (RELIABILITIES), or z-\(\cdot\)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`
CHAPTER 13. CLASS DOCUMENTATION

- **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::miPLIets

### 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 &os) 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.

#### 13.81.1 Detailed Description
Base class for all nondeterministic iterators (the DAKOTA/UQ branch).
The base class for nondeterministic iterators consolidates uncertain variable data and probabilistic utilities for inherited classes.

#### 13.81.2 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().
13.81.  NOND CLASS REFERENCE

Referenced by NonDExpansion::compute_statistics(), NonDGlobalReliability::importance_sampling(), NonDLocalReliability::initialize_class_data(), NonDExpansion::initialize_expansion(), NonDBayesCalibration::initialize_model(), 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 transformation 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_random_variable_types ( short u_space_type )

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(), NonDBayesCalibration::initialize_model(), NonD::initialize_random_variables(), NonDBayesCalibration::NonDBayesCalibration(), NonDIntegration::NonDIntegration(), and NonDReliability::NonDReliability().

void initialize_random_variable_parameters ( )

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(), NonDBayesCalibration::initialize_model(), NonD::initialize_random_variables(), NonDLocalReliability::mean_value(), NonDLocalReliability::mpp_search(), and NonDGlobalReliability::quantify_uncertainty().

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`.

```cpp
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()`.

```cpp
void finalize_run() [inline], [protected], [virtual]
```

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`.

```cpp
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()`.

```cpp
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()`, `NonDAdapImpSampling::print_results()`, `NonDGPImpSampling::print_results()`, `NonDAdaptiveSampling::print_results()`, `NonDPOFDarts::print_results()`, and `NonDSampling::print_statistics()`.

```cpp
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().

```cpp
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().

```cpp
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().

```cpp
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

### 13.82 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 coefficient 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

13.82.1 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.

13.82.2 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.

13.82.3 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

13.83 NonDAdaptiveSampling Class Reference

Class for testing various Adaptively sampling methods using geometric, statistitical, and topological information of the surrogate.

Inheritance diagram for NonDAdaptiveSampling:

```
  Iterator
   
    Analyzer
        
          NonD
              
                NonDSampling
                      
                            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 compute 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 other 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 other 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 other 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 other 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 other 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 other 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_complexes (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_amspace (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 type 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 the type of surrogate used to fit the data. Options are: global_kriging, global_mars, global_neural_network, global_polynomial, global_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

#### 13.83.1 Detailed Description

Class for testing various Adapтивely sampling methods using geometric, statisticstical, 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.

#### 13.83.2 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.
NonDAdaptiveSampling::outputDir, Iterator::outputLevel, NonDAdaptiveSampling::outputValidationData, NonDAdaptiveSampling::parse_options(), Iterator::probDescDB, NonDSampling::randomSeed, ActiveSet::request_values(), 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

### 13.84 NonDBayesCalibration Class Reference

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

Inheritance diagram for NonDBayesCalibration:

```
NonDBayesCalibration
   |       |
   |       |
   |       |
NonDCalibration
   |       |
   |       |
NonDAnalyzer
   |       |
   |       |
   |       |
Iterator
```

#### 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 (ParLevLItet pl_iter)
  derived class contributions to freeing the communicators associated with this Iterator instance
- const Model & algorithm_space_model () const
- void initialize_model ()
  initialize emulator model and probability space transformations

**Protected Attributes**

- short emulatorType
  the emulator type: NO_EMULATOR, GP_EMULATOR, PCE_EMULATOR, or SC_EMULATOR
- Model mcmcModel
  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)
- Iterator stochExpIterator
  NonDPolynomialChaos or NonDStochCollocation instance for defining a PCE/SC-based mcmcModel.
- int numSamples
  number of samples in the chain (e.g. number of MCMC samples)
- int chainCycles
  number of update cycles for MCMC chain (implemented by restarting of short chains)
- int randomSeed
  random seed for MCMC process
- bool standardizedSpace
  flag indicating use of a variable transformation to standardized probability space for the model or emulator
- bool adaptPosteriorRefine
  flag indicating usage of adaptive posterior refinement; currently makes sense for unstructured grids in GP and PCE least squares/CS
- String proposalCovarType
  approach for defining proposal covariance
- RealVector proposalCovarData
  data from user input of proposal covariance
- String proposalCovarFilename
  filename for user-specified proposal covariance
- String proposalCovarInputType
  approach for defining proposal covariance

**Additional Inherited Members**

**13.84.1 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.
13.84.2 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 Response::active_set(), NonDBayesCalibration::adaptPosteriorRefine, Iterator::algorithm_space_model(), Iterator::assign_rep(), Model::assign_rep(), NonD::cdfFlag, NonDBayesCalibration::chainCycles, Model::current_response(), NonDBayesCalibration::emulatorType, ProblemDescDB::get_bool(), ProblemDescDB::get_int(), ProblemDescDB::get_real(), ProblemDescDB::get_rv(), ProblemDescDB::get_string(), ProblemDescDB::get_usa(), ProblemDescDB::get_ushort(), 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(), Iterator::maxEvalConcurrency, Iterator::maxIterations, NonDBayesCalibration::mcmcModel, NonDBayesCalibration::numSamples, Iterator::outputLevel, Iterator::probDescDB, NonDBayesCalibration::proposalCovarType, NonDBayesCalibration::randomSeed, ActiveSet::request_values(), NonD::requested_levels(), NonD::respLevelTarget, NonD::respLevelTargetReduce, NonDBayesCalibration::standardizedSpace, NonDBayesCalibration::stochExpIterator, NonD::transform_model(), and NonD::verify_correlation_support().

13.84.3 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::mcmcModel.

The documentation for this class was generated from the following files:

- NonDBayesCalibration.hpp
- NonDBayesCalibration.cpp

13.85 NonDCalibration Class Reference

Inheritance diagram for NonDCalibration:
Public Member Functions

- NonDCalibration (ProblemDescDB &problem_db, Model &model)
  standard constructor
- NonDCalibration()
  destructor

Protected Member Functions

- 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
- bool calibrationData
  flag indicating whether there is calibration data present
- size_t numExperiments
  number of experiments to read from data file
- size_t numExpConfigVars
  number of columns in data file which are state variables
- StringArray varianceTypesRead
  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

- 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

13.85.1 Detailed Description

This class ...

13.85.2 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::find_state_index(), ProblemDescDB::get_sizet(), Iterator::iteratedModel, NonDCalibration::numExpConfigVars, and Iterator::probDescDB.

The documentation for this class was generated from the following files:

- NonDCalibration.hpp
- NonDCalibration.cpp

13.86 NonDCubature Class Reference

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

Inheritance diagram for NonDCubature:

```
  NonDCubature
  NonDIntegration
  NonD
  Analyzer
  Iterator
```

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

13.86.1 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.

13.86.2 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 non_d_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.

13.86.3 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

13.87 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
- bool calibrateSigma
  flag to indicate if the sigma terms should be calibrated (default true)
- 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 Gelman-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<br boost::uniform_real<double>> priorSamplers
  samplers for the uniform prior PDFs for each variable

Static Private Attributes

- static NonDDREAMBayesCalibration * NonDDREAMInstance
  Pointer to current class instance for use in static callback functions.

Additional Inherited Members

13.87.1 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)
13.87. 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, NonDDREAMBayesCalibration::numCR,
NonDDREAMBayesCalibration::numGenerations, and NonDBayesCalibration::numSamples.

13.87.3 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::calibrateSigma, NonDDREAMBayesCalibration::crossoverChainPairs,
NonDDREAMBayesCalibration::NonDDREAMInstance, NonDDREAMBayesCalibration::numChains, Analyzer::numContinuousVars,
NonDDREAMBayesCalibration::numCR, Analyzer::numFunctions, 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, NonDDREAMBayesCalibration::NonDDREAMInstance,
NonDDREAMBayesCalibration::numChains, NonDDREAMBayesCalibration::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::numGenerator.

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 ExperimentData::all_data(), ExperimentData::apply_covariance(), NonDDREAMBayesCalibration::calibrateSigma, NonDCalibration::calibrationData, Model::compute_response(), Model::continuous_variables(), Dakota::copy_data_partial(), Model::current_response(), NonDCalibration::expData, ExperimentData::form_residuals_deprecated(), Response::function_values(), NonDDREAMBayesCalibration::likelihoodScale, NonDBayesCalibration::mcmcModel, NonDDREAMBayesCalibration::NonDDREAMInstance, ExperimentData::num_total_exppoints(), Analyzer::numContinuousVars, NonDCalibration::numExperiments, Analyzer::numFunctions, Iterator::outputLevel, and ExperimentData::variance_active().

```cpp
void quantify_uncertainty( ) [protected], [virtual]
```

redefined from DakotaNonD

Perform the uncertainty quantification

Implements NonD.

References Dakota::abort_handler(), NonDDREAMBayesCalibration::calibrateSigma, NonDCalibration::calibrationData, Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), NonDBayesCalibration::emulatorType, NonDCalibration::expData, NonD::generate_system_seed(), ExperimentData::get_main_diagonal(), NonDBayesCalibration::initialize_model(), Iterator::iterator_rep(), ExperimentData::load_data(), NonDBayesCalibration::mcmcModel, NonDDREAMBayesCalibration::NonDDREAMInstance, Analyzer::numContinuousVars, Analyzer::numFunctions, NonDBayesCalibration::numSamples, NonDDREAMBayesCalibration::paramMaxs, NonDDREAMBayesCalibration::paramMins, NonDDREAMBayesCalibration::priorDistributions, NonDDREAMBayesCalibration::priorSamplers, NonDBayesCalibration::randomSeed, NonDDREAMBayesCalibration::rnumGenerator, NonDBayesCalibration::standardizedSpace, and NonDBayesCalibration::stochExpIterator.

13.87.4 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

13.88 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
- virtual void append (const RealMatrix &samples, const IntResponseMap &resp_map)
  - append new data to uSpaceModel and update expansion order (PCE only)

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 ()
CHAPTER 13. CLASS DOCUMENTATION

- virtual void increment_order_and_grid()  
  uniformly increment the expansion order and structured/unstructured grid (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 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-based 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**

**13.88.1 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.

**13.88.2 Member Function Documentation**

```cpp
const Model & algorithm_space_model( ) const [inline], [virtual]
```

default definition that gets redefined in selected derived Minimizers

Reimplemented from **Analyzer**.

References **NonDExpansion::uSpaceModel**.

```cpp
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**.

```cpp
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**, **NonExpansion::covarianceControl**, **NonExpansion::respCovariance**, and **NonExpansion::respVariance**.

Referenced by **NonDStochCollocation::compute_covariance_metric**, **NonExpansion::increment_sets**, and **NonExpansion::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::approximations_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::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::methodPCLiter, NonD::miPLIndex, PecosApproximation::moments(), NonD::natafTransform, 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().

13.88.3 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 auxiliary
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(), NonDPolynomialChaos::ratio_samples_to_order(), 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

13.89 NonDGlobalEvidence Class Reference

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

Inheritance diagram for NonDGlobalEvidence:

```
NonDAnalyzer
  NonDAnalyzer::Iterator
  NonDAnalyzer::NonD
    NonD::NonDInterval
      NonDInterval::NonDGlobalInterval
        NonDGlobalInterval::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** ()

- **NonDExpansion.hpp**
- **NonDExpansion.cpp**
post-process the interval computed for a response function

- void post_process_final_results ()
  perform final post-processing

Additional Inherited Members

13.89.1 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

13.90 NonDGlobalInterval Class Reference

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

Inheritance diagram for NonDGlobalInterval:

```
NonDGlobalEvidence

NonDGlobalInterval

NonDInterval

NonD

Analyzer

Iterator

```

Public Member Functions

- NonDGlobalInterval (ProblemDescDB &problem_db, Model &model)
  constructor
- ~NonDGlobalInterval ()
13.90. NONDGLOBALINTERVAL CLASS REFERENCE

- **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**
CHAPTER 13. CLASS DOCUMENTATION

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 **prevFinStar**
  
  stores previous solution value for assessing convergence

- size_t **sblIterNum**
  
  surrogate-based minimization/maximization iteration count
13.91. NONDGLOBALRELIABILITY CLASS REFERENCE

- 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

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

13.90.1 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 approaches 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.

13.90.2 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

13.91 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 improvement 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 function

- 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**

**13.91.1 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
13.92 NonDGlobalSingleInterval Class Reference

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

Inheritance diagram for NonDGlobalSingleInterval:

```
  NonDGlobalSingleInterval
    NonDGlobalInterval
      NonDInterval
        NonD
          Analyzer
            Iterator
```

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

13.92.1 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 approaches 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

13.93 NonDGPImpSampling Class Reference

Class for the Gaussian Process-based Importance Sampling method.

Inheritance diagram for NonDGPImpSampling:

```
NonDGPImpSampling
    NonDSampling
    NonD
    Analyzer
    Iterator
```

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.*
Additional Inherited Members

13.93.1 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).

13.93.2 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 Response::active_set(), Iterator::assign_rep(), Model::assign_rep(), NonD::construct_lhs(), Model::current_response(), ProblemDescDB::get_bool(), ProblemDescDB::get_int(), ProblemDescDB::get_string(), ProblemDescDB::get_ushort(), 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, ActiveSet::request_values(), NonDSampling::rngName, NonDGPImpSampling::sampleRhoOne, NonDSampling::samplingVarsMode, NonDSampling::statsFlag, NonDSampling::vary_pattern(), and NonDSampling::varyPattern.

13.93.3 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(),
CHAPTER 13. CLASS DOCUMENTATION

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

13.94 NonDGPMsABayesCalibration Class Reference

Generates posterior distribution on model parameters given experiment data.

Inheritance diagram for NonDGPMsABayesCalibration:

```
NonDGPMsABayesCalibration
  NonDBayesCalibration
    NonDCalibration
      NonD
        Analyzer
          Iterator
```

Public Member Functions

- **NonDGPMsABayesCalibration** (ProblemDescDB &problem_db, Model &model)
  
  *constructor*

- **~NonDGPMsABayesCalibration** ()
  
  *destructor*

Public Attributes

- int emulatorSamples
number of samples of the simulation to construct the GP

- 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
- unsigned short **approxImportFormat**
  build data import tabular format
- bool **approxImportActiveOnly**
  import active variables only

Protected Member Functions

- void **derived_initCommunicators** (ParLevLIter pl_iter)
  derived class contributions to initializing the communicators associated with this Iterator instance
- void **derived_setCommunicators** (ParLevLIter pl_iter)
  derived class contributions to setting the communicators associated with this Iterator instance
- void **derived_freeCommunicators** (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.

Private Attributes

- Iterator **lhsIter**
  LHS iterator for generating samples for GP.

Static Private Attributes

- static NonDGPMSABayesCalibration * **NonDGPMSAInstance**
  print the final statistics

Additional Inherited Members

13.94.1 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.

13.94.2 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 Dakota::abort_handler(), Iterator::assign_rep(), NonDGPMSABayesCalibration::emulatorSamples, ProblemDescDB::get_string(), NonDGPMSABayesCalibration::lhsIter, NonDBayesCalibration::mcmcModel, Iterator::probDescDB, and NonDBayesCalibration::randomSeed.

13.94.3 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

Implements NonD.

References Iterator::all_responses(), Analyzer::all_samples(), Iterator::all_samples(), NonDGPMSABayesCalibration::approxImportActiveOnly, NonDGPMSABayesCalibration::approxImportFile, NonDGPMSABayesCalibration::approxImportFormat, NonDGPMSABayesCalibration::calibrateSigmaFlag, ExperimentData::config_vars(), Model::continuous_lower_bounds(), Model::continuous_upper_bounds(), Model::continuous_variables(), NonDGPMSABayesCalibration::emulatorSamples, NonD::mPLIndex, NonD::miPLIndex, NonDGPMSABayesCalibration::NonDGPMSAInstance, NonDCalibration::numExperiments, Analyzer::numFunctions, NonDBayesCalibration::numSamples, NonD::numUncertainVars, Iterator::outputLevel, Iterator::run(), and ExperimentData::scalar_data().

13.94.4 Member Data Documentation

NonDGPMSABayesCalibration * NonDGPMSAInstance [static], [private]

print the final statistics

Pointer to current class instance for use in static callback functions

Referenced by NonDGPMSABayesCalibration::quantify_uncertainty().

The documentation for this class was generated from the following files:

- NonDGPMSABayesCalibration.hpp
- NonDGPMSABayesCalibration.cpp

13.95 NonDIncremLHSSampling Class Reference

Performs incremental 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

13.95.1 Detailed Description

Performs incremental 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 incremental 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 incremental 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.

13.95.2 Constructor & Destructor Documentation

NonDIncremLHSSampling ( ProblemDescDB & \textit{problem db}, Model & \textit{model} )

\textbf{constructor}

This constructor is called for a standard letter-envelope iterator instantiation. In this case, \textit{set db_list_nodes} has been called and \textit{probDescDB} can be queried for settings from the method specification.

13.95.3 Member Function Documentation

\textbf{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 \textit{statsFlag} is set.

Implements NonD.

References Dakota::abort_handler(), Model::aleatory_distribution_params(), 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::num-BinomialVars, Analyzer::numContinuousVars, NonD::numExponentialVars, NonD::numFrechetVars, NonD::num-GammaVars, NonD::numGeometricVars, NonD::numGumbelVars, NonD::numHistogramBinVars, NonD::num-HistogramPtIntVars, NonD::numHistogramPtRealVars, NonD::numHistogramPtStringVars, NonD::numHyper-GeomVars, NonD::numLognormalVars, NonD::numLoguniformVars, NonD::numNegBinomialVars, NonD::num-NormalVars, NonD::numPoissonVars, NonDSampling::numSamples, NonD::numTriangularVars, NonD::num-UniformVars, NonD::numWeibullVars, NonDIncremLHSSampling::previousSamples, NonDIncremLHSSampling::rank_sort(), NonDIncremLHSSampling::rawData, NonDSampling::sampleRanks, NonDSampling::sampleRanks-Mode, 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

13.96 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 utility 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

13.96.1 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.

13.96.2 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::numDiscrete-IntVars, 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.

13.96.3 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
Converting 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

13.97 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.*
Additional Inherited Members

13.97.1 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

13.98 NonDLHSEvidence Class Reference

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

Inheritance diagram for NonDLHSEvidence:
CHAPTER 13. CLASS DOCUMENTATION

NonDLHSEvidence
NonDLHSInterval
NonDInterval
NonD
Analyzer
Iterator

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

13.98.1 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

13.99 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

#### 13.99.1 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

### 13.100 NonDLHSSampling Class Reference

Performs LHS and Monte Carlo sampling for uncertainty quantification. Inheritance diagram for NonDLHSSampling:

```
NonDLHSSampling
  NonDSampling
    NonD
      Iterator
      Analyzer
      NonD
      NonDSampling
```

### 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

13.100.1 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.

13.100.2 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.

References Model::num_primary_fns(), NonDLHSSampling::numResponseFunctions, and Model::primary_-fn_type().

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().

13.100.3 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::numResponseFunctions, NonDSampling::numSamples, NonDSampling::statsFlag, NonDLHSSampling::varBasedDecompFlag, and Analyzer::variance_based_decomp().

The documentation for this class was generated from the following files:

- NonDLHSSampling.hpp
- NonDLHSSampling.cpp

13.101 NonDLHSSingleInterval Class Reference

Class for pure interval propagation using LHS.

Inheritance diagram for NonDLHSSingleInterval:

```
  NonDLHSSingleInterval
     |                    
     |                    
  NonDInterval
     |                    
     |                    
  NonDLHSInterval
     |                    
     |                    
  Iterator
   |                    
   |                    
  Analyzer
```


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

13.101.1 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

13.102 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

13.102.1 Detailed Description

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

The NonDLocalEvidence 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

13.103 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_initCommunicators (ParLevLIter pl_iter)**
  derived class contributions to initializing the communicators associated with this Iterator instance
- **void derived_setCommunicators (ParLevLIter pl_iter)**
  derived class contributions to setting the communicators associated with this Iterator instance
- **void derived_freeCommunicators (ParLevLIter pl_iter)**
  derived class contributions to freeing the communicators associated with this Iterator instance
- **void quantifyUncertainty ()**
  Performs a gradient-based optimization to determine interval bounds for an entire function or interval bounds on a particular statistical estimator.
- **unsigned short usesMethod () const**
  return name of active optimizer method
- **void methodRecourse ()**
  perform an MPP optimizer method switch due to a detected conflict

Protected Member Functions

- **virtual void initialize ()**
  perform any required initialization
- **virtual void setCellBounds ()**
  set the optimization variable bounds for each cell
- **virtual void truncateToCellBounds (RealVector &initial_pt)**
  truncate initial_pt to respect current cell lower/upper bounds
- **virtual void postProcessCellResults (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

13.103.1 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 approaches 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
13.104 NonDLocalReliability Class Reference

Class for the reliability methods within DAKOTA/UQ.

Inheritance diagram for NonDLocalReliability:

```
  Iterator
     /  \
    /    \
   /     \
  Analyzer
     /  \
    /    \
   /     \
  NonD
     /  \
    /    \
   /     \
  NonDReliability
```

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_rescue ( )**
  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 $\|\text{norm } u\|^2$.

• 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 $G(u) = \text{response level}$.

• 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 $\|\text{norm } u\|^2 = (\beta - \beta_{\text{bar}})^2$.

• 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 $\beta^* = \beta - \beta_{\text{bar}}$.

• 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
CHAPTER 13. CLASS DOCUMENTATION

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

13.104.1 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 reliability/_probabilities or as the performance measure approach (PMA) for performing the inverse mapping of reliability/probability levels to response levels.

13.104.2 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 a 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 \((\text{norm } u)^2 = (\beta-bar)^2\).

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 \(\beta^* = \beta-star\).

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]

cost-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, Iterator::subIteratorFlag, NonDLocalReliability::taylorOrder, and NonDReliability::uSpaceModel.

Referenced by NonDLocalReliability::mean_value(), and NonDLocalReliability::mpp_search().

void initialize_class_data ( ) [private]

cost-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::prevFnGradDLLev0, NonDLocalReliability::prevFnGradULev0, NonDLocalReliability::prevMPPULev0, NonDLocalReliability::ranVarMeansU, Iterator::subIteratorFlag, Model::update_from_subordinate_model(), and NonDLocalReliability::warmStartFlag.

Referenced by NonDLocalReliability::mpp_search().

```cpp
class NonDLocalReliability {
public:
  void initialize_level_data();
  void initializempp_search_data();
  void updatempp_search_data(const Variables& vars, const Response& resp);
private:
  // Convenience functions for initializing/warm starting MPP search data
  class InitializeMPPSearchData {
  public:
    InitializeMPPSearchData();
    // Initialize the MPP search data for each response function
  private:
    // Initialize level data
    class InitializeLevelData {
    public:
      InitializeLevelData();
      // Initialize level data for each response function prior to level 0
    private:
      // Initialize MPP search data
      class InitializeMPPSearchData {
      public:
        InitializeMPPSearchData();
        // Initialize MPP search data for each response function at a particular z/p/beta level
      private:
        // Update MPP search data
        class UpdateMPPSearchData {
        public:
          UpdateMPPSearchData();
          // Update MPP search data for each response function at a particular z/p/beta level
        private:
          // Convenience functions for updating MPP search data
          class ConvenienceFunctions {
          public:
            ConvenienceFunctions();
            // Convenience function for updating MPP search data
          private:
            // Convenience function for updating MPP search data
          }
        }
      }
    }
  }
}
```

Referenced by NonDLocalReliability::mpp_search().
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(), 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, NonDLocalReliability::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::respFnCount, NonDLocalReliability::signed_norm(), NonDReliability::statCount, NonDLocalReliability::taylorOrder, NonDLocalReliability::update_limit_state_surrogate(), NonDLocalReliability::update_pma_maximize(), NonDReliability::uSpaceModel, NonDLocalReliability::warnStartFlag, and NonDLocalReliability::warningBits.

Referenced by NonDLocalReliability::mpp_search().

```cpp
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, NonD::computedRespLevels, 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::respLevelTargetReduce, NonD::respLevelTarget, NonD::respLevelTargetReduce, NonDReliability::statCount, Iterator::subIteratorFlag, NonD::totalLevelRequests, and NonDLocalReliability::warnStartFlag.

Referenced by NonDLocalReliability::mpp_search().

```cpp
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-
CHAPTER 13. CLASS DOCUMENTATION

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. auxiliary 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, NonD::AdaptImpSampling::final_probability(), NonDReliability::importanceSampler, NonDLocalReliability::integrationOrder, NonDLocalReliability::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

13.105 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

13.105.1 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 approaches 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
13.106 NonDPOFDarts Class Reference

Base class for POF Dart methods within DAKOTA/UQ.

Inheritance diagram for NonDPOFDarts:

```
NonDPOFDarts
   |   
   v   
Analyzer
   |   
Iterator
   |   
NonD
   |   
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 ()
• 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)
• 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 emulatorSamples
• String lipschitzType
• double Q [1220]
• int index
• 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
Additional Inherited Members

13.106.1 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.

The documentation for this class was generated from the following files:

- NonDPOFDarts.hpp
- NonDPOFDarts.cpp

13.107 NonDPolynomialChaos Class Reference

Nonintrusive polynomial chaos expansion approaches to uncertainty quantification.

Inheritance diagram for NonDPolynomialChaos:

```
NonDPolynomialChaos
  |________________________
  |                      |
  |    Analyzer          |
  |______________________|
          |                 |
          |   NonD           |
          |__________________|
                  |             |
                  |  NonDExpansion  |
                          |___________|
                                  |      |
                                  |     NonDPolynomialChaos
```

Public Member Functions

- **NonDPolynomialChaos (ProblemDescDB &problem_db, Model &model)**
  
  _standard constructor_

- **NonDPolynomialChaos (Model &model, short exp_coeffs_approach, const UShortArray &num_int_seq, const RealVector &dim_pref, short u_space_type, bool piecewise_basis, bool use_derivs)**

  _alternate constructor for numerical integration_
• NonDPolynomialChaos (Model &model, short exp_coeffs_app, const UShortArray &exp_order_seq, const RealVector &dim_pref, Real colloc_ratio, int seed, short u_space_type, bool piecewise_basis, bool use_derivs, bool cv_flag)
  alternate constructor for regression
• NonDPolynomialChaos ()
  destructor

Protected Member Functions
• derived_init_communicators (ParLevLIter pl_iter)
  derived class contributions to initializing the communicators associated with this Iterator instance
• derived_set_communicators (ParLevLIter pl_iter)
  derived class contributions to setting the communicators associated with this Iterator instance
• derived_free_communicators (ParLevLIter pl_iter)
  derived class contributions to freeing the communicators associated with this Iterator instance
• resolve_inputs (short &u_space_type, short &data_order)
  perform error checks and mode overrides
• initialize_u_space_model ()
  initialize uSpaceModel polynomial approximations with PCE/SC data
• increment_specification_sequence ()
  increment the input specification sequence (PCE only)
• compute_expansion ()
  form or import an orthogonal polynomial expansion using PCE methods
• append (const RealMatrix &samples, const IntResponseMap &resp_map)
  append new data to uSpaceModel and update expansion order
• increment_order_and_grid ()
  uniformly increment the order of the polynomial chaos expansion and define a consistent grid increment
• print_coefficients (std::ostream &s)
  print the PCE coefficient array for the orthogonal basis
• archive_coefficients ()
  archive the PCE coefficient array for the orthogonal basis

Private Member Functions
• increment_grid_from_order ()
  define a grid increment that is consistent with an advancement in expansion order
• increment_order_from_grid ()
  define an expansion order that is consistent with an advancement in structured/unstructured grid level/density
• int terms_ratio_to_samples (size_t num_exp_terms, Real colloc_ratio)
  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)
  convert number of expansion terms and number of collocation samples to a collocation ratio
• void ratio_samples_to_order (Real colloc_ratio, int num_samples, UShortArray &exp_order)
  convert collocation ratio and number of samples to expansion order
• 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
CHAPTER 13. CLASS DOCUMENTATION

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

13.107.1 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.
13.107.2 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(), Response::active_set(), 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(), Model::current_response(), 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, NonDPolynomialChaos::refineControl, NonDPolynomialChaos::refineType, ActiveSet::request_values(), 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, const UShortArray & num_int_seq, const RealVector & dim_pref, short u_space_type, bool piecewise_basis, bool use_derivs )
alternate constructor for numerical integration

This constructor is used for helper iterator instantiation on the fly that employ numerical integration (quadrature, sparse grid, cubature).

References Response::active_set(), Model::assign_rep(), NonDExpansion::construct_cubature(), NonDExpansion::construct_quadrature(), NonDExpansion::construct_sparse_grid(), Model::current_response(), NonDExpansion::expansionCoeffsApproach, NonDExpansion::initialize(), NonDPolynomialChaos::initialize_u_space_model(), Iterator::iteratedModel, Iterator::outputLevel, ActiveSet::request_values(), NonDPolynomialChaos::resolve_inputs(), NonD::transform_model(), and NonDExpansion::uSpaceModel.

NonDPolynomialChaos ( Model & model, short exp_coeffs_approach, const UShortArray & exp_order_seq, const RealVector & dim_pref, Real colloc_ratio, int seed, short u_space_type, bool piecewise_basis, bool use_derivs, bool cv_flag )
alternate constructor for regression

This constructor is used for helper iterator instantiation on the fly that employ regression (least squares, CS, OLI).

References Response::active_set(), Model::assign_rep(), NonDPolynomialChaos::collocRatio, NonD::construct_lhs(), NonDExpansion::construct_quadrature(), Model::current_response(), NonDIntegration::dimension_preference_to_anisotropic_order(), NonDPolynomialChaos::dimPrefSpec, NonDExpansion::expansionBasisType, NonDPolynomialChaos::expOrderSeqSpec, NonDExpansion::initialize(), NonDPolynomialChaos::initialize_u_space_model(), Iterator::iteratedModel, Analyzer::numContinuousVars, NonDExpansion::numSamplesOnModel, Iterator::outputLevel, NonDPolynomialChaos::randomSeed, ActiveSet::request_values(), NonDPolynomialChaos::resolve_inputs(), NonDPolynomialChaos::sequenceIndex, NonDPolynomialChaos::tensorRegression, NonDPolynomialChaos::terms_ratio_to_samples(), NonD::transform_model(), and NonDExpansion::uSpaceModel.
13.107.3 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(), DataFitSurrModel::total_points(), NonDQuadrature::update(), and NonDExpansion::uSpaceModel.

**void increment order and grid ( )** [protected], [virtual]

uniformly increment the order of the polynomial chaos expansion and define a consistent grid increment
   - Used for uniform refinement of regression-based PCE.
   - Reimplemented from NonDExpansion.

References SharedApproxData::data_rep(), NonDPolynomialChaos::increment_grid_from_order(), SharedPecosApproxData::increment_order(), Model::shared_approximation(), and NonDExpansion::uSpaceModel.

**void increment grid from order ( )** [private]

define a grid increment that is consistent with an advancement in expansion order
   - Used for uniform refinement of regression-based PCE.

References NonDPolynomialChaos::collocRatio, SharedApproxData::data_rep(), SharedPecosApproxData::expansion_order(), NonDExpansion::expansionBasisType, NonDQuadrature::increment_grid(), 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(), DataFitSurrModel::total_points(), NonDQuadrature::update(), and NonDExpansion::uSpaceModel.

Referenced by NonDPolynomialChaos::increment_order_and_grid().

**void increment order from grid ( )** [private]

define an expansion order that is consistent with an advancement in structured/unstructured grid level/density
   - Used for uniform refinement of regression-based PCE.

References NonDPolynomialChaos::collocRatio, SharedApproxData::data_rep(), SharedPecosApproxData::expansion_order(), NonDExpansion::numSamplesOnModel, NonDPolynomialChaos::ratio_samples_to_order(), Model::shared_approximation(), and NonDExpansion::uSpaceModel.

Referenced by NonDPolynomialChaos::append().

The documentation for this class was generated from the following files:

- NonDPolynomialChaos.hpp
- NonDPolynomialChaos.cpp
13.108 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:

```
NonDQuadrature
    
    NonDIntegration
    
    NonD
    
    Analyzer
    
    Iterator

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)`
  - 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

13.108.1 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.

13.108.2 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]

creator

This creator 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(), Problem-
DescDB::get_short(), ProblemDescDB::get_ushort(), Iterator::maxEvalConcurrency, Iterator::maxIterations, Non-
D::natafTransform, NonDQuadrature::nestedRules, NonDIntegration::numIntDriver, Iterator::outputLevel, Iterator-
::probDescDB, NonDQuadrature::reset(), and NonDQuadrature::tpqDriver.

13.108.3 Member Function Documentation

void initialize_grid ( const std::vector < Pecos::BasisPolynomial > & poly_basis ) [protected],
[virtual]

Used in combination with alternate NonDQuadrature creator.

Implements NonDIntegration.

References NonDQuadrature::nestedRules, Analyzer::numContinuousVars, Non-
DQuadrature::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, Non-
DQuadrature::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
Bayesian inference using the QUESO library from UT Austin.

Inheritance diagram for NonDQUESOBayesCalibration:

```
    Iterator
     /    \
    /     \
   /      \
Analyzer
 /        \
/          \
/            \
NonD
 /          \
/            \
/              \
NonDCalibration
 /                \
/                  \
/                    \
NonDBayesCalibration
```

### Public Member Functions
- **NonDQUESOBayesCalibration** (ProblemDescDB &problem_db, Model &model)
  - *standard constructor*
- ∼**NonDQUESOBayesCalibration** ()
  - *destructor*
- **Real prior_density** (const QUESO::GslVector &qv)
  - *compute the prior PDF for a particular MCMC sample*

### Protected Member Functions
- void **quantify_uncertainty** ()
  - *redefined from DakotaNonD*
- void **print_results** (std::ostream &s)
  - *redefined from DakotaAnalyzer*
- void **init_queso_environment** ()
  - *initialize the QUESO FullEnvironment on the Dakota MPIComm*
- void **init_precond_request_value** ()
  - *initialize the ASV value for preconditioned cases*
- void **init_residual_response** ()
  - *initialize a residual response for use in data transformations*
- void **init_queso_solver** ()
  - *define variables, options, likelihood callback, and inverse problem*
- void **precondition_proposal** ()
  - *use derivative information from the emulator to define the proposal covariance (inverse of misfit Hessian)*
• void run_queso_solver ()
  perform the MCMC process
• void run_chain_with_restarting ()
  short term option to restart the MCMC chain with updated proposal density computed from the emulator at a new
  starting point
• void filter_chain (size_t update_cntr, unsigned short batch_size)
  extract batch_size points from the MCMC chain and store final aggregated set within allSamples
• void chain_to_local (unsigned short batch_size, std::map<Real, size_t> &local_best)
  store indices of best batch_size samples from the current MCMC chain within the local_best array
• void local_to_aggregated (unsigned short batch_size, const std::map<Real, size_t> &local_best)
  update bestSamples aggregation using new contributions from the current MCMC chain
• void aggregated_to_all ()
  following aggregation cycles, copy bestSamples to allSamples
• void local_to_all (const std::map<Real, size_t> &local_best)
  in the absence of aggregation cycles, copy local_best to allSamples
• void update_center ()
  update the starting point for a restarted MCMC chain using last point from previous chain
• void update_model ()
  evaluates allSamples on iteratedModel and update the mcmcModel emulator with all{{Samples,Responses}}
• Real assess_emulator_convergence ()
  compute the L2 norm of the change in emulator coefficients
• void init_parameter_domain ()
  initialize the QUESO parameter space, min, max, initial, and domain
• void prior_proposal_covariance ()
  use covariance of prior distribution for setting proposal covariance
• void user_proposal_covariance (const String &input_fmt, const RealVector &cov_data, const String &cov_filename)
  set proposal covariance from user-provided diagonal or matrix
• void set_ip_options ()
  set inverse problem options calIpOptionsValues common to all solvers
• void set_mh_options ()
  set MH-specific inverse problem options calIpMhOptionsValues
• void update_chain_size (unsigned int size)
  update MH-specific inverse problem options calIpMhOptionsValues
• void copy_gsl (const QUESO::GslVector &qv, RealVector &rv)
  local copy gsl utility
• void copy_gsl (const QUESO::GslVector &qv, RealMatrix &rm, int i)
  local copy gsl utility

Static Protected Member Functions

• static double dakotaLikelihoodRoutine (const QUESO::GslVector &paramValues, const QUESO::GslVector
+paramDirection, const void *functionDataPtr, QUESO::GslVector *gradVector, QUESO::GslMatrix *hessian-
Matrix, QUESO::GslVector *hessianEffect)
  Likelihood function for call-back from QUESO to DAKOTA for evaluation.
Protected Attributes

- **String mcmcType**
  
  MCMC type ("dram" or "delayed_rejection" or "adaptive_metropolis" or "metropolis_hastings" or "multilevel", within QUESO)

- **Real likelihoodScale**
  
  scale factor for likelihood; deprecated

- **bool calibrateSigma**
  
  flag to indicated if the sigma terms should be calibrated (default true)

- **short precondRequestValue**
  
  the active set request value to use in proposal preconditioning

- **Response residualResponse**
  
  local Response in which to store computed residuals

- **bool logitTransform**
  
  flag indicating user activation of logit transform option

Private Attributes

- **boost::shared_ptr < QUESO::EnvOptionsValues > envOptionsValues**
  
  options for setting up the QUESO Environment

- **boost::shared_ptr < QUESO::FullEnvironment > quesoEnv**
  
  top-level QUESO Environment

- **boost::shared_ptr < QUESO::VectorSpace < QUESO::GslVector, QUESO::GslMatrix > > paramSpace**
  
  QUESO parameter space based on number of calibrated parameters.

- **boost::shared_ptr < QUESO::BoxSubset < QUESO::GslVector, QUESO::GslMatrix > > paramDomain**
  
  QUESO parameter domain: hypercube based on min/max values.

- **boost::shared_ptr < QUESO::GslVector > paramInitials**
  
  initial parameter values at which to start chain

- **boost::shared_ptr < QUESO::GslMatrix > proposalCovMatrix**
  
  proposal covariance for DRAM

- **boost::shared_ptr < QUESO::SipOptionsValues > callIpOptionsValues**
  
  general inverse problem options

- **boost::shared_ptr < QUESO::MhOptionsValues > callPmOptionsValues**
  
  MH-specific inverse problem options.
• boost::shared_ptr
  < QUESO::GenericScalarFunction
  < QUESO::GslVector,
  QUESO::GslMatrix > > likelihoodFunctionObj

• boost::shared_ptr
  < QUESO::BaseVectorRV
  < QUESO::GslVector,
  QUESO::GslMatrix > > priorRv

• boost::shared_ptr
  < QUESO::GenericVectorRV
  < QUESO::GslVector,
  QUESO::GslMatrix > > postRv

• boost::shared_ptr
  < QUESO::StatisticalInverseProblem
  < QUESO::GslVector,
  QUESO::GslMatrix > > inverseProb

• std::map< Real, QUESO::GslVector > bestSamples
  array for managing aggregation of best MCMC samples across multiple (restarted) chains

• RealVectorArray prevCoeffs
  cache previous expansion coefficients for assessing convergence of emulator refinement process

Static Private Attributes

• static NonDQUESOBayesCalibration * NonDQUESOInstance
  Pointer to current class instance for use in static callback functions.

Additional Inherited Members

13.109.1 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) Cen-
ter at UT Austin. The name QUESO stands for Quantification of Uncertainty for Estimation, Simulation, and
Optimization.

13.109.2 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.

References Dakota::abort_handler(), NonDQUESOBayesCalibration::calibrateSigma, NonDCalibration::calibration-
Data, NonDCalibration::expData, NonDQUESOBayesCalibration::init_queso_environment(), and ExperimentData-
::load_data().
13.109.3 Member Function Documentation

`void quantify_uncertainty()` [protected], [virtual]

redefined from DakotaNonD
   - Perform the uncertainty quantification
   Implements NonD.
   References Dakota::abort_handler(), NonDBayesCalibration::adaptPosteriorRefine, NonDQUESOBayesCalibration::assess_emulator_convergence(), NonDQUESOBayesCalibration::calibrateSigma, Analyzer::compactMode, Iterator::convergenceTol, NonDBayesCalibration::emulatorType, NonDCalibration::expData, NonDQUESOBayesCalibration::init_parameter_domain(), NonDQUESOBayesCalibration::init_precond_request_value(), NonDQUESOBayesCalibration::init_queso_solver(), NonDQUESOBayesCalibration::init_residual_response(), NonDBayesCalibration::initialize_model(), Iterator::maxIterations, NonDQUESOBayesCalibration::NonDQUESOInstance, Analyzer::numContinuousVars, Analyzer::numFunctions, Iterator::outputLevel, NonDQUESOBayesCalibration::paramSpace, NonDQUESOBayesCalibration::prior_proposal_covariance(), NonDBayesCalibration::proposalCovData, NonDBayesCalibration::proposalCovarFileName, NonDBayesCalibration::proposalCovarInputType, NonDBayesCalibration::proposalCovarType, NonDQUESOBayesCalibration::proposalCovMatrix, NonDQUESOBayesCalibration::run_chain_with_restarting(), ExperimentData::scalar_sigma_est(), NonDQUESOBayesCalibration::update_model(), and NonDQUESOBayesCalibration::user_proposal_covariance().

`void prior_proposal_covariance()` [protected]

use covariance of prior distribution for setting proposal covariance
   Must be called after paramMins/paramMaxs set above
   References Model::continuous_distribution_moment(), NonDBayesCalibration::mcmcModel, Analyzer::numContinuousVars, Iterator::outputLevel, NonDQUESOBayesCalibration::proposalCovMatrix, and NonDBayesCalibration::standardizedSpace.
   Referenced by NonDQUESOBayesCalibration::quantify_uncertainty().

`void user_proposal_covariance(const String &input_fmt, const RealVector &cov_data, const String &cov_filename)` [protected]

set proposal covariance from user-provided diagonal or matrix
   This function will convert user-specified cov_type = "diagonal" | "matrix" data from either cov_data or cov_filename and populate a full QUESO::GslMatrix in proposalCovMatrix with the covariance.
   References Analyzer::numContinuousVars, NonDQUESOBayesCalibration::proposalCovMatrix, and NonDBayesCalibration::standardizedSpace.
   Referenced by NonDQUESOBayesCalibration::quantify_uncertainty().

`void set_ip_options()` [protected]

set inverse problem options callIpOptionsValues common to all solvers
   References NonDQUESOBayesCalibration::callIpOptionsValues.
   Referenced by NonDQUESOBayesCalibration::init_queso_solver().

13.109.4 Member Data Documentation

`bool logitTransform` [protected]

flag indicating user activation of logit transform option
this option is useful for preventing rejection or resampling for out-of-bounds samples by transforming bounded
domains to [-inf,inf].
Referenced by NonDQUESOBayesCalibration::set_fh_options().
The documentation for this class was generated from the following files:
• NonDQUESOBayesCalibration.hpp
• NonDQUESOBayesCalibration.cpp

13.110 NonDReliability Class Reference

Base class for the reliability methods within DAKOTA/UQ.
Inheritance diagram for NonDReliability:

```
NonDReliability
  NonD
  Iterator
  NonDGlobalReliability
  NonDLocalReliability
```

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_A-
  PROX
13.11. NONDRELIABILITY CLASS REFERENCE

- **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

13.11.1 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.

13.11.2 Member Function Documentation

```cpp
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
CHAPTER 13. CLASS DOCUMENTATION

13.111 NonDSampling Class Reference

Base class for common code between NonDLHSSampling, NonDIncremLHSSampling, and NonDAdaptImp-Sampling.

Inheritance diagram for NonDSampling:

```
NonDSampling
  NonDAnalyzer
  NonDIterator
  NonDAdaptImpSampling
  NonDAdaptiveSampling
  NonDGPImpSampling
  NonDIncremLHSSampling
  NonDLHSSampling
```

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/computed-RespLevels

- 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 (epistemic 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\_dsv, size\_t &num\_drv) 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\_dauv, 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\_deuv, 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\_drsv) 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, ACTIVE, UNIFORM, or ALL. 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

13.111.1 Detailed Description

Base class for common code between NonDLHSSampling, NonDIncremLHSSampling, and NonDAdaptImp-Sampling.

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.

13.111.2 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::max-EvalConcurrency, 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.

13.111.3 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::...
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::num-
ContinuousVars, Analyzer::numDiscreteIntVars, Analyzer::numDiscreteRealVars, Analyzer::numDiscreteString-
Vars, 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_dauv, 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::numCont-
AleatUncVars, NonD::numDiscIntAleatUncVars, NonD::numDiscRealAleatUncVars, NonD::numDiscStringAleat-
UncVars, 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_deuv, 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::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::adriv(), Model::adsv(), Model::current_variables(), Variables::cv(), Variables::cv_start(), Variables::div(), Variables::drv(), Variables::dsv(), Variables::dsv_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
13.112 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:

```
  NonDSparseGrid
  |                     
  v                     
Iterator
  |                     
  v                     
Analyzer
  |                     
  v                     
NonD
  |                     
  v                     
NonDIntegration
  |                     
  v                     
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 level 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 ()
**CHAPTER 13. CLASS DOCUMENTATION**

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**

**13.112.1 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.
13.112.2 Constructor & Destructor Documentation

NonDSparseGrid ( Model & model, const UShortArray & ssg_level_seq, const RealVector & dim_pref, short exp_coeffs_sln_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::convergence-Tol, 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::ssg-Driver, and NonDSparseGrid::ssgLevelRef.

13.112.3 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

13.113 NonDStochCollocation Class Reference

Nonintrusive stochastic collocation approaches to uncertainty quantification.

Inheritance diagram for NonDStochCollocation:
CHAPTER 13. CLASS DOCUMENTATION

Public Member Functions

- NonDStochCollocation (ProblemDescDB &problem_db, Model &model)
  standard constructor
- NonDStochCollocation (Model &model, short exp_coeffs_approach, const UShortArray &num_int_seq,
  const RealVector &dim_pref, 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

13.113.1 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.
13.113.2 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 Response::active_set(), Model::assign_rep(), ParallelLibrary::command_line_check(), NonDExpansion::construct_expansion_sampler(), NonDExpansion::construct_quadrature(), NonDExpansion::construct_sparse_grid(), Model::current_response(), NonDExpansion::expansionBasisType, NonDExpansion::expansionCoeffsApproach, ProblemDescDB::get(rv), ProblemDescDB::get_short(), ProblemDescDB::get_string(), ProblemDescDB::get_usa(), ProblemDescDB::get_ushort(), NonDExpansion::initialize(), NonDStochCollocation::initialize_u_space_model(), Iterator::iteratedModel, NonDExpansion::nestedRules, NonD::numContDesVars, NonD::numContEpistUncVars, NonD::numContStateVars, NonD::numSamplesOnModel, Iterator::outputLevel, Iterator::parallelLib, NonDExpansion::piecewiseBasis, Iterator::probDescDB, NonDExpansion::refineControl, ActiveSet::request_values(), NonDStochCollocation::resolve_inputs(), NonD::transform_model(), and NonDExpansion::uSpaceModel.

NonDStochCollocation ( Model & model, short exp_coeffs_approach, const UShortArray & num_int_seq, const RealVector & dim_pref, short u_space_type, bool piecewise_basis, bool use_derivs )

alternate constructor

This constructor is used for helper iterator instantiation on the fly.
References Response::active_set(), Model::assign_rep(), NonDExpansion::construct_quadrature(), NonDExpansion::construct_sparse_grid(), Model::current_response(), NonDExpansion::expansionBasisType, NonDExpansion::expansionCoeffsApproach, NonDExpansion::initialize(), NonDStochCollocation::initialize_u_space_model(), Iterator::iteratedModel, Iterator::outputLevel, NonDExpansion::piecewiseBasis, ActiveSet::request_values(), NonDStochCollocation::resolve_inputs(), NonD::transform_model(), and NonDExpansion::uSpaceModel.

13.113.3 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::initializePtU, 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::initializePtU, 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

### 13.114 NonlinearCGOptimizer Class Reference

Inheritance diagram for NonlinearCGOptimizer:

```
   Iterator
     ↓
   Minimizer
     ↓
   Optimizer
     ↓
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 Js_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 absissa 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

13.114.1 Detailed Description

Experimental implementation of nonlinear CG optimization

13.114.2 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

13.115 NPSOLOptimizer Class Reference

Wrapper class for the NPSOL optimization library.

Inheritance diagram for NPSOLOptimizer:

```
   iterator
   
   minimizer
   
   optimizer   solbase

   npsolooptimizer
```

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 *, double *, double *, double *, int &), const int &derivative_level, const Real &conv_tol)

alternating 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

### 13.115.1 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.

### 13.115.2 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_coeffs, const RealVector & nonlin_ineq_lower_bnds, const RealVector & nonlin_ineq_upper_bnds, const RealVector & nonlin_eq_targets, void(*)(int &, int &, double *, double &, int &, int &), int &, int &, int &)

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

### 13.116 OptDartsOptimizer Class Reference

Wrapper class for OptDarts Optimizer.

Inheritance diagram for OptDartsOptimizer:

```
        Iterator
         /    \
        /     \
Minimizer
       /       \n   Optimizer
      /         \nOptDartsOptimizer
```

#### 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_dartsTerminate()

Release memory and exit cleanly.

- void opt_darts_plot_discs_2d(size_t icandidate)
- void opt_darts_plot_hull_2d(size_t icandidate, size_t ifunc)

Convenience function for plotting iterates.

- 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
13.117. OPTIMIZER CLASS REFERENCE

- 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

13.116.1 Detailed Description

Wrapper class for OptDarts Optimizer.

The documentation for this class was generated from the following files:

- OptDartsOptimizer.hpp
- OptDartsOptimizer.cpp

13.117 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_linp, size_t num_lin_eq, size_t num_nlin_ineq, size_t num_nlin_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 transform-
  ation.
• 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

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

13.117.1 Detailed Description

Base class for the optimizer branch of the iterator hierarchy.

The Optimizer class provides common data and functionality for DOTOptimizer, CONMINOptimizer, NPS-
OLOptimizer, SNLLOptimizer, NLPQLPOptimizer, COLINOptimizer, and JEGAOptimizer.
13.117.2 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 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::localRecast_retrieve(), Optimizer::localObjectiveRecast, Minimizer::modify_s2n(), Minimizer::needResp_trans_byvars(), Minimizer::numNonlinearConstraints, Minimizer::numUserPrimaryFns, Minimizer::post_run(), Minimizer::primaryRespScaleFlag, Minimizer::response_modify_s2n(), 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, Minimizer::calibrationDataFlag, Model::current_response(), Minimizer-::data_difference_core(), Dakota::data_pairs, Response::function_values(), Model::interface_id(), Dakota::lookup-\_by_val(), Model::num_functions(), Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, Minimizer-::numTotalCalibTerms, Minimizer::numUserPrimaryFns, Minimizer::optimizationFlag, Minimizer::original_model(), Model::primary_response_fn_weights(), Minimizer::print_model_resp(), and Minimizer::print_residuals().

```cpp
template <typename T>
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(), Minimizer::calibrationDataFlag, Model::current_response(), Minimizer::gnewton_set_recast(), Model::hessian_type(), Iterator::iteratedModel, Minimizer::minimizerRecasts, Minimizer::numContinuousVars, Minimizer::numFunctions, Minimizer::numIterPrimaryFns, Minimizer::numNonlinear-Constraints, Minimizer::numNonlinearIneqConstraints, Optimizer::numObjectiveFns, Minimizer::numTotalCalib-Terms, Minimizer::numUserPrimaryFns, Iterator::outputLevel, Model::primary_fntype(), 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().
```

```cpp
template <typename T>
void primaryresp_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().
```

```cpp
template <typename T, typename V, typename W, typename R, typename R2, typename R3>
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 SGO-PTApplication 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().
```
The documentation for this class was generated from the following files:

- DakotaOptimizer.hpp
- DakotaOptimizer.cpp

### 13.118 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_mpi_run_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`

• unsigned short `tabularFormat`
  Tabular format options; see enum

• 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)

13.118.1 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`.

13.118.2 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(), Output-Manager::mpirunFlag, ProgramOptions::output_file(), OutputManager::outputLevel, ConsoleRedirector::push_back(), Dakota::start_dakota_heartbeat(), ProgramOptions::user_stderr_redirect(), ProgramOptions::user_stdout_redirect(), and OutputManager::worldRank.`

13.118.3 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, Output-Manager::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, OutputManager::tabularDataFStream, and OutputManager::tabularFormat.

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, OutputManager::tabularDataFStream, and OutputManager::tabularFormat.

Referenced by SurrBasedMinimizer::initialize_graphics(), and Iterator::initialize_graphics().

The documentation for this class was generated from the following files:

- OutputManager.hpp
- OutputManager.cpp

13.119 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
13.119.1 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

13.120 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 privileges in order to streamline implementation

13.120.1 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).
13.120.2 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

13.121 ParallelDirectApplicInterface Class Reference

Sample derived interface class for testing parallel simulator plug-ins using assign_rep().

Inheritance diagram for ParallelDirectApplicInterface:

```
    Interface
     |  
     | ApplicationInterface
     |  |  
     |  | DirectApplicInterface
     |  |  |  | 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)
13.122. PARALLELLEVEL CLASS REFERENCE

no-op hides base error; job batching occurs within \texttt{wait\_local\_evaluations()}

- \texttt{void \_wait\_local\_evaluations (Dakota::PRPQueue \&prp\_queue)}
  evaluate the batch of jobs contained in \texttt{prp\_queue}
- \texttt{void test\_local\_evaluations (Dakota::PRPQueue \&prp\_queue)}
  invokes \texttt{wait\_local\_evaluations()} (no special nowait support)
- \texttt{void set\_communicators\_checks (int max\_eval\_concurrency)}
  no-op hides default run-time error checks at \texttt{DirectApplicInterface} level

Private Member Functions

- \texttt{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

13.121.1 Detailed Description

Sample derived interface class for testing parallel simulator plug-ins using \texttt{assign\_rep()}.

The plug-in \texttt{ParallelDirectApplicInterface} resides in namespace \texttt{SIM} and uses a copy of \texttt{textbook()} to perform parallel parameter to response mappings. It is used to demonstrate plugging in a parallel direct analysis driver into \texttt{Dakota} in library mode. Test input files can then use an analysis driver of "plugin\_textbook".

13.121.2 Member Function Documentation

\texttt{void test\_local\_evaluations ( Dakota::PRPQueue \& prp\_queue ) [inline], [protected]}

invokes \texttt{wait\_local\_evaluations()} (no special nowait support)

For use by \texttt{ApplicationInterface::serve\_evaluations\_asynch()}, which can provide a batch processing capability within message passing schedulers (called using chain \texttt{ApplicationInterface::serve\_evaluations()} from \texttt{Model\::serve()} from \texttt{IteratorScheduler::run\_iterator()}).

References \texttt{ParallelDirectApplicInterface::wait\_local\_evaluations()}.

The documentation for this class was generated from the following files:

- \texttt{PluginParallelDirectApplicInterface.hpp}
- \texttt{PluginParallelDirectApplicInterface.cpp}

13.122 ParallelLevel Class Reference

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

Public Member Functions

- \texttt{ParallelLevel ()}
  \texttt{default constructor}
- \texttt{ParallelLevel (const ParallelLevel \&pl)}
  \texttt{copy constructor}
- \texttt{\sim ParallelLevel ()}
CHAPTER 13. CLASS DOCUMENTATION

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

• void clear ()
  deallocate the communicators in this ParallelLevel
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
- int serverId
  server identifier
- 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/ next higher level serverIntraComm
- int hubServerCommRank
  rank in hubServerIntraComm
- int hubServerCommSize
  size of hubServerIntraComm
- MPI_Comm * hubServerInterComms
  intercomm. array on hub processor
Friends

- class `ParallelLibrary`
  
  The `ParallelLibrary` class has special access privileges in order to streamline implementation.

### 13.122.1 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.

### 13.122.2 Member Function Documentation

#### void clear( ) [inline]

deallocate the communicators in this `ParallelLevel`

This appears to be more robust outside of the destructor due to interactions among managed deallocation and default deallocation (e.g., explicitly freeing a communicator and then default deallocating its handle). References `ParallelLevel::commSplitFlag`, `ParallelLevel::dedicatedMasterFlag`, `ParallelLevel::hubServerInterComm`, `ParallelLevel::hubServerInterComms`, `ParallelLevel::hubServerIntraComm`, `ParallelLevel::idlePartition`, `ParallelLevel::numServers`, `ParallelLevel::serverId`, and `ParallelLevel::serverIntraComm`.

The documentation for this class was generated from the following file:

- `ParallelLibrary.hpp`

### 13.123 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 async_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 async_local_analysis_concurrency, short default_config, short analysis_scheduling, bool peer_dynamic_avail)`
  
  split an analysis communicator into evaluation communicators
split an evaluation communicator into 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 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 filename

- 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 \texttt{send} (int &send\_int, int dest, int tag, const \	exttt{ParallelLevel} &parent\_pl, const \	exttt{ParallelLevel} &child\_pl)
  
  \begin{itemize}
  \item blocking integer send at the current communication level
  \end{itemize}

• void \texttt{isend} (MPI\_PackBuffer &send\_buff, int dest, int tag, MPI\_Request &send\_req, const \	exttt{ParallelLevel} &parent\_pl, const \	exttt{ParallelLevel} &child\_pl)

  \begin{itemize}
  \item nonblocking buffer send at the current communication level
  \end{itemize}

• void \texttt{isend} (int &send\_int, int dest, int tag, MPI\_Request &send\_req, const \	exttt{ParallelLevel} &parent\_pl, const \	exttt{ParallelLevel} &child\_pl)

  \begin{itemize}
  \item nonblocking integer send at the current communication level
  \end{itemize}

• void \texttt{recv} (MPI\_UnpackBuffer &recv\_buff, int source, int tag, MPI\_Status &status, const \	exttt{ParallelLevel} &parent\_pl, const \	exttt{ParallelLevel} &child\_pl)

  \begin{itemize}
  \item blocking buffer receive at the current communication level
  \end{itemize}

• void \texttt{recv} (int &recv\_int, int source, int tag, MPI\_Status &status, const \	exttt{ParallelLevel} &parent\_pl, const \	exttt{ParallelLevel} &child\_pl)

  \begin{itemize}
  \item blocking integer receive at the current communication level
  \end{itemize}

• void \texttt{irecv} (MPI\_UnpackBuffer &recv\_buff, int source, int tag, MPI\_Request &recv\_req, const \	exttt{ParallelLevel} &parent\_pl, const \	exttt{ParallelLevel} &child\_pl)

  \begin{itemize}
  \item nonblocking buffer receive at the current communication level
  \end{itemize}

• void \texttt{irecv} (int &recv\_int, int source, int tag, MPI\_Request &recv\_req, const \	exttt{ParallelLevel} &parent\_pl, const \	exttt{ParallelLevel} &child\_pl)

  \begin{itemize}
  \item nonblocking integer receive at the current communication level
  \end{itemize}

• void \texttt{check\_mi\_index} (size\_t &index) const

  \begin{itemize}
  \item process NPOS default and perform error checks
  \end{itemize}

• void \texttt{send\_mi} (int &send\_int, int dest, int tag, size\_t index=NPOS)

  \begin{itemize}
  \item blocking send at the metaiterator-iterator communication level
  \end{itemize}

• void \texttt{isend\_mi} (int &send\_int, int dest, int tag, MPI\_Request &send\_req, size\_t index=NPOS)

  \begin{itemize}
  \item nonblocking send at the metaiterator-iterator communication level
  \end{itemize}

• void \texttt{recv\_mi} (int &recv\_int, int source, int tag, MPI\_Status &status, size\_t index=NPOS)

  \begin{itemize}
  \item blocking receive at the metaiterator-iterator communication level
  \end{itemize}

• void \texttt{irecv\_mi} (int &recv\_int, int source, int tag, MPI\_Request &recv\_req, size\_t index=NPOS)

  \begin{itemize}
  \item nonblocking receive at the metaiterator-iterator communication level
  \end{itemize}

• void \texttt{send\_mi} (MPI\_PackBuffer &send\_buff, int dest, int tag, size\_t index=NPOS)

  \begin{itemize}
  \item blocking send at the metaiterator-iterator communication level
  \end{itemize}

• void \texttt{isend\_mi} (MPI\_PackBuffer &send\_buff, int dest, int tag, MPI\_Request &send\_req, size\_t index=NPOS)

  \begin{itemize}
  \item nonblocking send at the metaiterator-iterator communication level
  \end{itemize}

• void \texttt{recv\_mi} (MPI\_UnpackBuffer &recv\_buff, int source, int tag, MPI\_Status &status, size\_t index=NPOS)

  \begin{itemize}
  \item blocking receive at the metaiterator-iterator communication level
  \end{itemize}

• void \texttt{irecv\_mi} (MPI\_UnpackBuffer &recv\_buff, int source, int tag, MPI\_Request &recv\_req, size\_t index=NPOS)

  \begin{itemize}
  \item nonblocking receive at the metaiterator-iterator communication level
  \end{itemize}

• void \texttt{send\_ie} (int &send\_int, int dest, int tag)

  \begin{itemize}
  \item blocking send at the iterator-evaluation communication level
  \end{itemize}

• void \texttt{isend\_ie} (int &send\_int, int dest, int tag, MPI\_Request &send\_req)

  \begin{itemize}
  \item nonblocking send at the iterator-evaluation communication level
  \end{itemize}

• void \texttt{recv\_ie} (int &recv\_int, int source, int tag, MPI\_Status &status)
blocking receive at the iterator-evaluation communication level

- \textbf{void irecv}\_ie (int &recv\_int, int source, int tag, MPI\_Request &recv\_req)

nonblocking receive at the iterator-evaluation communication level

- \textbf{void send}\_ie (MPIPackBuffer &send\_buff, int dest, int tag)

blocking send at the iterator-evaluation communication level

- \textbf{void isend}\_ie (MPIPackBuffer &send\_buff, int dest, int tag, MPI\_Request &send\_req)

nonblocking send at the iterator-evaluation communication level

- \textbf{void recv}\_ie (MPIUnpackBuffer &recv\_buff, int source, int tag, MPI\_Status &status)

blocking receive at the iterator-evaluation communication level

- \textbf{void irecv}\_ie (MPIUnpackBuffer &recv\_buff, int source, int tag, MPI\_Request &recv\_req)

nonblocking receive at the iterator-evaluation communication level

- \textbf{void send}\_ea (int &send\_int, int dest, int tag)

blocking send at the evaluation-analysis communication level

- \textbf{void isend}\_ea (int &send\_int, int dest, int tag, MPI\_Request &send\_req)

nonblocking send at the evaluation-analysis communication level

- \textbf{void recv}\_ea (int &recv\_int, int source, int tag, MPI\_Status &status)

blocking receive at the evaluation-analysis communication level

- \textbf{void irecv}\_ea (int &recv\_int, int source, int tag, MPI\_Request &recv\_req)

nonblocking receive at the evaluation-analysis communication level

- \textbf{void bcast} (int &data, const ParallelLevel &pl)

  broadcast an integer across the server\textit{IntraComm} of a ParallelLevel

- \textbf{void bcast} (short &data, const ParallelLevel &pl)

  broadcast an integer across the server\textit{IntraComm} of a ParallelLevel

- \textbf{void bcast} (MPIPackBuffer &send\_buff, const ParallelLevel &pl)

  broadcast a MPI\textit{PackBuffer} across the server\textit{IntraComm} of a ParallelLevel

- \textbf{void bcast} (MPIUnpackBuffer &recv\_buff, const ParallelLevel &pl)

  broadcast a MPI\textit{UnpackBuffer} across the server\textit{IntraComm} of a ParallelLevel

- \textbf{void bcast}\_hs (int &data, const ParallelLevel &pl)

  broadcast an integer across the hub\textit{Server\textit{IntraComm}} of a ParallelLevel

- \textbf{void bcast}\_hs (MPIPackBuffer &send\_buff, const ParallelLevel &pl)

  broadcast a MPI\textit{PackBuffer} across the hub\textit{Server\textit{IntraComm}} of a ParallelLevel

- \textbf{void bcast}\_hs (MPIUnpackBuffer &recv\_buff, const ParallelLevel &pl)

  broadcast a MPI\textit{UnpackBuffer} across the hub\textit{Server\textit{IntraComm}} of a ParallelLevel

- \textbf{void bcast}\_w (int &data)

  broadcast an integer across MPI\textit{COMM\_WORLD}

- \textbf{void bcast}\_i (int &data, size\_t index=\_NPOS)

  broadcast an integer across an iterator communicator

- \textbf{void bcast}\_i (short &data, size\_t index=\_NPOS)

  broadcast a short integer across an iterator communicator

- \textbf{void bcast}\_e (int &data)

  broadcast an integer across an evaluation communicator

- \textbf{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 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

13.123.1 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.

13.123.2 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().

13.123.3 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, ParallelLibrary::outputManager, ParallelLibrary::programOptions, OutputManager::tabularDataFile, and OutputManager::tabularDataFlag.

Referenced by Environment::construct(), and IteratorScheduler::partition().

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 (ParLevLLIter mi_pl_iter ) [inline]

add a new node to parallelConfigurations and increment currPCIter; limit miPLIter 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::miPLIter, 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, ParallelLevel::outputManager, ParallelLibrary::parallelLevels, ParallelLevel::procsPerServer, ParallelLevel::serverCommRank, ParallelLevel::serverCommSize, ParallelLevel::serverId, ParallelLevel::serverIntraComm, ParallelLevel::serverMasterFlag, ParallelLibrary::startMPTime, 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 Meta-Iterators 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

13.124 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, unsigned short tabular_format) const**
  write a ParamResponsePair object in tabular format (all variables active/inactive) to an std::ostream

- **void write.tabular.labels (std::ostream &s, unsigned short tabular_format) 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 & variables () const**
  return the parameters object

- **void variables (const Variables &vars)**
  set the parameters object
• const Response & response () const
  return the response object
• void response (const Response &resp)
  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 prpVariables
  the set of parameters for the function evaluation
• Response prpResponse
  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

13.124.1 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.
CHAPTER 13. CLASS DOCUMENTATION

13.124.2 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.

13.124.3 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::prpResponse, and ParamResponsePair::prpVariables.

**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::prpResponse, and ParamResponsePair::prpVariables.

13.124.4 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
13.125  ParamStudy Class Reference

Class for vector, list, centered, and multidimensional parameter studies.

Inheritance diagram for ParamStudy:

```
<table>
<thead>
<tr>
<th>Iterator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyzer</td>
</tr>
<tr>
<td>PStudyDACE</td>
</tr>
<tr>
<td>ParamStudy</td>
</tr>
</tbody>
</table>
```

**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 continuous/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, unsigned short tabular_format, bool active_only)
  - *load list of points from data file and distribute among listCVPoints, listDIVPoints, listDSVPoints, and listDRVPoints*
CHAPTER 13. CLASS DOCUMENTATION

- 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)
13.125.  PARAMSTUDY CLASS REFERENCE

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 a 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
CHAPTER 13. CLASS DOCUMENTATION

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 ParameterStudy (a specification option)

- `IntVector contStepsPerVariable`
  number of offsets in the plus and the minus direction for each continuous variable in a centered ParameterStudy

- `IntVector discIntStepsPerVariable`
  number of offsets in the plus and the minus direction for each discrete integer variable in a centered ParameterStudy

- `IntVector discStringStepsPerVariable`
  number of offsets in the plus and the minus direction for each discrete string variable in a centered ParameterStudy

- `IntVector discRealStepsPerVariable`
  number of offsets in the plus and the minus direction for each discrete real variable in a centered ParameterStudy

- `UShortArray contVarPartitions`
  number of partitions for each continuous variable in a multidim ParameterStudy

- `UShortArray discIntVarPartitions`
  number of partitions for each discrete integer variable in a multidim ParameterStudy

- `UShortArray discStringVarPartitions`
  number of partitions for each discrete string variable in a multidim ParameterStudy

- `UShortArray discRealVarPartitions`
  number of partitions for each discrete real variable in a multidim ParameterStudy

Additional Inherited Members

13.125.1 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.

13.125.2 Member Function Documentation

```cpp
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().

```cpp
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::disc_set_string_values(), Iterator::iteratedModel, Iterator::methodName, Analyzer::post_run(), PStudyDACE::pStudyDACE_SensGlobal, and Iterator::subIteratorFlag.

```cpp
bool load_distribute_points ( const String & points_filename, unsigned short tabular_format, 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::disc_int_lower_bounds(), Model::disc_int_sets(), Model::disc_int_upper_bounds(), Model::disc_set_int_values(), Model::disc_set_real_values(), Model::disc_set_string_values(), Iterator::iteratedModel, ParamStudy::listCVPoints, ParamStudy::listDIVPoints, ParamStudy::listDSVPoints, ParamStudy::listDRVPoints, 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 listDRV-Points

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::listDSVPoints, Anchorer::numContinuousVars, Anchorer::numDiscreteIntVars, Anchorer::numDiscreteRealVars, Anchorer::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

13.126 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

13.126.1 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

13.127 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

13.127.1 Detailed Description

wrapper to delegate to the ParamResponsePair hash_value function

The documentation for this struct was generated from the following file:

- PRPMultiIndex.hpp
13.128  PecosApproximation Class Reference

Derived approximation class for global basis polynomials.

Inheritance diagram for PecosApproximation:

```
Approximation
   ^
  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 num-Vars 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

13.128.1 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.

13.128.2 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]
finalizes 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

13.129 ProblemDescDB Class Reference

The database containing information parsed from the DAKOTA input file.
Inheritance diagram for ProblemDescDB:

```
    ProblemDescDB
     ^
     |  NIDRProblemDescDB
```

Public Member Functions

- **ProblemDescDB ()**
  *default constructor*
- **ProblemDescDB (ParallelLibrary &parallel_lib)**
  *standard constructor*
- **ProblemDescDB (const ProblemDescDB &db)**
  *copy constructor*
- **~ProblemDescDB ()**
**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 RealMatrixArray & get_rma (const String &entry_name) const
  get a RealMatrixArray out of the database based on an identifier string
• 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_s (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 (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 overrides and concurrency levels

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 overrides and concurrency levels

bool is_null () const
  function to check dbRep (does this envelope contain a letter)
CHAPTER 13. CLASS DOCUMENTATION

Static Public Member Functions

• static int min_procs_per_level (int min_procs_per_server, int pps_spec, int num_serv_spec)
  compute minimum partition size for a parallel level based on lower level overrides

• static int max_procs_per_level (int max_procs_per_server, int pps_spec, int num_serv_spec, short sched_spec, int asynch_local_conc, bool peer_dynamic_avail, int max_concurrency)
  compute maximum partition size for a parallel level based on lower level 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 in iteratorList, or instantiate a new one

• const Iterator & get_iterator (Model &model)
  retrieve an existing Iterator, if it exists in iteratorList, or instantiate a new one

• const Iterator & get_iterator (const String &method_name, Model &model)
  retrieve an existing Iterator, if it exists in iteratorByNameList, 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 (short type, 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
• IteratorList iteratorByNameList
  list of iterator objects, one for each lightweight instantiation by name
• 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
  
  \textit{Iterator} requires access to \texttt{get\_model()}

• class SeqHybridMetaIterator

  \texttt{SeqHybridMetaIterator} requires access to \texttt{get\_model()}

• class CollabHybridMetaIterator

  \texttt{CollabHybridMetaIterator} requires access to \texttt{get\_model()}

• class ConcurrentMetaIterator

  \texttt{ConcurrentMetaIterator} requires access to \texttt{get\_model()}

• class SurrBasedLocalMinimizer

  \texttt{SurrBasedLocalMinimizer} requires access to \texttt{get\_iterator()}

• class SurrBasedGlobalMinimizer

  \texttt{SurrBasedGlobalMinimizer} requires access to \texttt{get\_iterator()}

\subsection{Detailed Description}

The database containing information parsed from the DAKOTA input file.

The \texttt{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 (\texttt{DataEnvironment}, \texttt{DataMethod}, \texttt{DataVariables}, \texttt{DataInterface}, or \texttt{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.

\subsection{Constructor & Destructor Documentation}

\texttt{ProblemDescDB ( )}

default constructor

  The default constructor: \texttt{dbRep} is \texttt{NULL} in this case. This makes it necessary to check for \texttt{NULL} in the copy constructor, assignment operator, and destructor.

\texttt{ProblemDescDB ( ParallelLibrary & parallel\_lib )}

standard constructor

  This is the envelope constructor which uses \texttt{problem\_db} to build a fully populated \texttt{db} object. It only needs to extract enough data to properly execute \texttt{get\_db(problem\_db)}, since the constructor overloaded with \texttt{Base-Constructor} builds the actual base class data inherited by the derived classes.

  References Dakota::abort\_handler(), ProblemDescDB::dbRep, and ProblemDescDB::get\_db().

\texttt{ProblemDescDB ( const ProblemDescDB & db )}

copy constructor

  Copy constructor manages sharing of \texttt{dbRep} and incrementing of \texttt{referenceCount}.

  References ProblemDescDB::dbRep, and ProblemDescDB::referenceCount.

\texttt{~ProblemDescDB ( )}

destructor

  Destructor decrements \texttt{referenceCount} and only deletes \texttt{dbRep} when \texttt{referenceCount} reaches \texttt{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).

13.129.3 Member Function Documentation

ProblemDescDB operator= (const ProblemDescDB & db)

assignment operator


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 world-Rank 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-
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

13.130 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:

```
  Interface
    ApplicationInterface
      ProcessApplicInterface
        ProcessHandleApplicInterface
          ForkApplicInterface
        SysCallApplicInterface
          SpawnApplicInterface
        GridApplicInterface
```

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<std::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
define filenames to create evaluation process
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
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)

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.

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::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::specifiedResultsFileName, and ApplicationInterface::suppressOutput.

Referenced by ProcessApplicInterface::read_results_files().

```cpp
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().

```cpp
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(), SysCallApplicInterface::spawn_input_filter_to_shell(), and SysCallApplicInterface::spawn_output_filter_to_shell().

```cpp
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 ProcessApplicInterface::useWorkdir.

Referenced by SpawnApplicInterface::create_analysis_process(), ForkApplicInterface::create_analysis_process(), SysCallApplicInterface::spawn_analysis_to_shell(), SysCallApplicInterface::spawn_evaluation_to_shell(), SysCallApplicInterface::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

### 13.131 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:
CHAPTER 13. CLASS DOCUMENTATION

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)  
  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
13.131. **PROCESSHANDLEAPPLICINTERFACE CLASS REFERENCE**

- 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_async** ()
  
  *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().*

13.131.1 **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).

13.131.2 **Constructor & Destructor Documentation**

**ProcessHandleApplicInterface** ( const ProblemDescDB & problem_db ) [inline]

constructor

argList sized 3 for [driver name, input file, output file]
13.131.3 Member Function Documentation

```cpp
int asynchronous_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().

```cpp
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().

```cpp
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().

```cpp
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::asynchLocalAnalysisServerId, ApplicationInterface::asyncLocalAnalysisConcurrency, ApplicationInterface::asyncLocalAnalysisFlag, 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::iFilterName, ProcessHandleApplicInterface::join_evaluation_process_group(), ApplicationInterface::master_dynamic_schedule_analyses(), ProcessApplicInterface::multipleParamsFiles, ApplicationInterface::numAnalysisDrivers, ApplicationInterface::numAnalysisServers, ProcessHandleApplicInterface::oFilterName, ApplicationInterface::parallelLib, ProcessApplicInterface::paramsFileName, ProcessApplicInterface::programNames, ProcessApplicInterface::resultsFileName, ProcessHandleApplicInterface::serve_analyses_synch(), ApplicationInterface::serve_analyses_synch(), ApplicationInterface::suppressOutput, and ProcessApplicInterface::synchronous_local_analyses().

```cpp
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().

```cpp
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().

```cpp
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().

```cpp
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

13.132 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 restart (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
• unsigned int pre_run_output_format() const
  tabular format for pre-run output
• unsigned int post_run_input_format() const
  tabular format for post-run input
• 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`
CHAPTER 13. CLASS DOCUMENTATION

13.132.1 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()

13.132.2 Member Function Documentation

void split_filenames ( const char *filenames, std::string &input_filename, std::string &output_filename )

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
13.133  

**PStudyDACE Class Reference**

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

Inheritance diagram for PStudyDACE:

```
    |-------------------|
    |       Iterator    |
    |                   |
    |       Analyzer    |
    |                   |
    |-------------------|
    |    PStudyDACE     |
    |                   |
    | DDACEDesignCompExp|
    | FSUDesignCompExp  |
    | ParamStudy        |
    | PSUADEDesignCompExp|
```

**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**
CHAPTER 13. CLASS DOCUMENTATION

quality measure
• double hMeas
  quality measure
• double tauMeas
  quality measure

Additional Inherited Members

13.133.1 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.

13.133.2 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
13.134  **PSUADEDesignCompExp Class Reference**

Wrapper class for the PSUADE library.

Inheritance diagram for PSUADEDesignCompExp:

```
PSUADEDesignCompExp
    |                  
    |                   
    |  Iterator         
    |                   
    v                   
Analyzer
    |                   
    |                   
    v                   
PStudyDACE
    |                   
    |                   
    v                   
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::allVariables() and Iterator::allResponses()
- size_t numDACERuns
  - counter for number of executions for this object
- bool varyPattern
  - flag for generating a sequence of seed values within multiple getParameterSets() 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

13.134.1 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.

13.134.2 Constructor & Destructor Documentation

PSUADEDesignCompExp ( ProblemDescDB & probDescDB, 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 PSUADE-DesignCompExp::numSamples.

13.134.3 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 inte-
grated 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

13.135 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** (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)
**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

### 13.135.1 Detailed Description

Specialization of DirectApplicInterface to link to Python analysis drivers. Includes convenience functions to map data to/from Python.

### 13.135.2 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

### 13.136 QuesoJointPdf< V, M > Class Template Reference

**Dakota** specialization of QUESO generic joint PDF.
Inherits BaseJointPdf< V, M >.

**Public Member Functions**

- **QuesoJointPdf** (const char *prefix, const QUESO::VectorSet< V, M > &domainSet, NonDQUESOBayesCalibration *nond_queso_ptr)
  Default constructor.
- **virtual ~QuesoJointPdf ()**
CHAPTER 13. CLASS DOCUMENTATION

Destructor:

- double actualValue (const V &domainVector, const V *domainDirection, V *gradVector, M *hessianMatrix, V *hessianEffect) const
  
  Actual value of the PDF (scalar function).

- double lnValue (const V &domainVector, const V *domainDirection, V *gradVector, M *hessianMatrix, V *hessianEffect) const
  
  Logarithm of the value of the function.

- double computeLogOfNormalizationFactor (unsigned int numSamples, bool m_logOfNormalizationFactor) const
  
  Computes the logarithm of the normalization factor.

Private Attributes

- NonDQUESOBayesCalibration * NonDQUESOInstance

13.136.1 Detailed Description
template<class V, class M> class Dakota::QuesoJointPdf< V, M >

Dakota specialization of QUESO generic joint PDF.

13.136.2 Constructor & Destructor Documentation

QuesoJointPdf ( const char *prefix, const QUESO::VectorSet< V, M > &domainSet, NonDQUESOBayesCalibration * nond_queso_ptr )

Default constructor.

Instantiates an object of the class, i.e. a scalar function, given a prefix and its domain.

The documentation for this class was generated from the following file:

- NonDQUESOBayesCalibration.cpp

13.137 QuesoVectorRV< V, M > Class Template Reference

Dakota specialization of QUESO vector-valued random variable.

Inherits BaseVectorRV< V, M >.

Public Member Functions

- QuesoVectorRV (const char *prefix, const QUESO::VectorSet< V, M > &imageSet, NonDQUESOBayesCalibration *nond_queso_ptr)
  
  Default constructor.

- virtual ~QuesoVectorRV ()
  
  Virtual destructor.

- void print (std::ostream &os) const
  
  TODO: Prints the vector RV (required pure virtual).
13.137.1 Detailed Description

template<class V, class M> class Dakota::QuesoVectorRV< V, M >

Dakota specialization of QUESO vector-valued random variable.

13.137.2 Constructor & Destructor Documentation

QuesoVectorRV ( const char * prefix, const QUESO::VectorSet< V, M > & imageSet, NonDQUESOBayesCalibration * nond_queso_ptr )

Default constructor.

Constructs a generic queso vector RV, given a prefix and the image set of the vector RV.

The documentation for this class was generated from the following file:

- NonDQUESOBayesCalibration.cpp

13.138 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:

```
Model
|
v
RecastModel
```

Public Member Functions


  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 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 &sub_model_vars, const Variables &recast_vars, const Response &sub_model Resp, Response &recast 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 sub-Model

- 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)

- IntIntPair estimate_partition_bounds (int max_eval_concurrency)
  estimate the minimum and maximum partition sizes that can be utilized by this *Model*

- 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

- bool db_lookup (const Variables &search_vars, const ActiveSet &search_set, Response &found_resp)
  RecastModel may need to map variables, ass before DB lookup, or responses after lookup.

Private Member Functions

- void initialize_data_from_submodel ()
  code shared among constructors to initialize base class data from submodel

- void update_from_submodel ()
  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 $N_{sm} \times N_r$ 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.
CHAPTER 13. CLASS DOCUMENTATION

- 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

  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

13.138.1 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.
13.138.2 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(), Response::copy(), Constraints::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 userDefinedConstraints. 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(), Response::copy(), Constraints::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().
13.138.3 Member Function Documentation

```cpp
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.


Referenced by EffGlobalMinimizer::minimize_surrogates_on_model(), NonDLocalReliability::mpp_search(), NonDGlobalReliability::optimize_gaussian_process(), NonDLocalInterval::quantify_uncertainty(), NonDGlobalInterval::quantify_uncertainty(), and Minimizer::scale_model().

```cpp
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().

```cpp
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.

```cpp
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::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::
13.139 RelaxedVarConstraints Class Reference

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

Inheritance diagram for RelaxedVarConstraints:

```
Constraints
    |
    V
RelaxedVarConstraints
```

Public Member Functions

- `RelaxedVarConstraints (const SharedVariablesData &svd)`
CHAPTER 13. CLASS DOCUMENTATION

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

13.139.1 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).

13.139.2 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

13.140 RelaxedVariables Class Reference

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

Inheritance diagram for RelaxedVariables:

```
        Variables
           |
           ▼
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

13.140.1 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)).

13.140.2 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().
13.140.3 Member Function Documentation

```cpp
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

13.141 Response Class Reference

Container class for response functions and their derivatives. `Response` provides the enveloper base class.

Inheritance diagram for `Response`:

```
Response
    |__________ExperimentResponse
    |            |__________SimulationResponse
```

Public Member Functions

- **`Response()`**
  
  *default constructor*

- **`Response(short type, const Variables &vars, const ProblemDescDB &problem_db)`**
  
  *standard constructor built from problem description database*

- **`Response(const SharedResponseData &srd, const ActiveSet &set)`**
  
  *alternate constructor that shares response data*

- **`Response(short type, const ActiveSet &set)`**
  
  *alternate constructor using limited data without sharing*

- **`Response(const SharedResponseData &srd)`**
  
  *alternate constructor using limited data (explicit disallows implicit type conversion)*

- **`Response(const Response &response)`**
  
  *copy constructor*

- **`virtual ~Response()`**
  
  *destructor*

- **`Response operator=(const Response &response)`**
  
  *assignment operator*

- **`const SharedResponseData & shared_data()`**
  
  *return sharedRespData*
- `SharedResponseData & shared_data ()`
  return sharedRespData
- `size_t num_functions () const`
  return the number of response functions
- `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 and verify consistent number of response functions
- `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 and reshape functionGradients/functionHessians if needed
- `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
- `RealVector function_values_view () const`
  return all function values as a view for accessing the function values vector from a const response
- `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 (int i) const`
  return the i-th function gradient as a const Real*
- `RealVector function_gradient_view (int i)`
  return the i-th function gradient as a SerialDenseVector view (shallow copy) for updating in place
- `RealVector function_gradient_view (int i) const`
  return the i-th function gradient as a SerialDenseVector view (shallow copy) for accessing a column vector from a const matrix
- `RealVector function_gradient_copy (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
• RealMatrix \texttt{function\_gradients\_view} () const
  
  \begin{verbatim}
  return all function gradients as a view for updating in place
  \end{verbatim}

• void \texttt{function\_gradient} (const RealVector &function\_grad, int i)
  
  \begin{verbatim}
  set a function gradient
  \end{verbatim}

• void \texttt{function\_gradients} (const RealMatrix &function\_grads)
  
  \begin{verbatim}
  set all function gradients
  \end{verbatim}

• const RealSymMatrix & \texttt{function\_hessian} (size\_t i) const
  
  \begin{verbatim}
  return the i-th function Hessian
  \end{verbatim}

• RealSymMatrix \texttt{function\_hessian\_view} (size\_t i)
  
  \begin{verbatim}
  return the i-th function Hessian as a Teuchos::View (shallow copy) for updating in place
  \end{verbatim}

• RealSymMatrix \texttt{function\_hessian\_view} (size\_t i) const
  
  \begin{verbatim}
  return the i-th function Hessian as a Teuchos::View (shallow copy) for accessing the i-th matrix within a const matrix array
  \end{verbatim}

• const RealSymMatrixArray & \texttt{function\_hessians} () const
  
  \begin{verbatim}
  return all function Hessians
  \end{verbatim}

• RealSymMatrixArray \texttt{function\_hessians\_view} ()
  
  \begin{verbatim}
  return all function Hessians as Teuchos::Views (shallow copies) for updating in place
  \end{verbatim}

• RealSymMatrixArray \texttt{function\_hessians\_view} () const
  
  \begin{verbatim}
  return all function Hessians as Teuchos::Views (shallow copies) for updating in place
  \end{verbatim}

• void \texttt{function\_hessian} (const RealSymMatrix &function\_hessian, size\_t i)
  
  \begin{verbatim}
  set a function Hessian
  \end{verbatim}

• void \texttt{function\_hessians} (const RealSymMatrixArray &function\_hessians)
  
  \begin{verbatim}
  set all function Hessians
  \end{verbatim}

• RealVector \texttt{field\_values\_view} (size\_t i) const
  
  \begin{verbatim}
  return const field values
  \end{verbatim}

• RealVector \texttt{field\_values\_view} (size\_t i)
  
  \begin{verbatim}
  return a "view" of a field value for updating in place
  \end{verbatim}

• void \texttt{field\_values} (const RealVector &field\_val, size\_t i)
  
  \begin{verbatim}
  set a field value
  \end{verbatim}

• RealMatrix \texttt{field\_coords\_view} (size\_t i)
  
  \begin{verbatim}
  return a "view" of a field value's coordinates
  \end{verbatim}

• const RealMatrix \texttt{field\_coords\_view} (size\_t i) const
  
  \begin{verbatim}
  return a const "view" of a field value's coordinates
  \end{verbatim}

• void \texttt{field\_coords} (const RealMatrix &field\_coords, size\_t i)
  
  \begin{verbatim}
  set a field value's coordinates
  \end{verbatim}

• RealMatrix \texttt{field\_gradients\_view} (size\_t i) const
  
  \begin{verbatim}
  return a view of the gradients of each field element
  \end{verbatim}

• RealSymMatrixArray \texttt{field\_hessians\_view} (size\_t i) const
  
  \begin{verbatim}
  return a view of the hessians of each field element
  \end{verbatim}

• const IntVector & \texttt{field\_lengths} () const
  
  \begin{verbatim}
  return the field lengths from sharedRespData
  \end{verbatim}

• void \texttt{field\_lengths} (const IntVector &field\_lens)
  
  \begin{verbatim}
  set the field lengths within sharedRespData
  \end{verbatim}
13.141. RESPONSE CLASS REFERENCE

- **const IntVector & num_coords_per_field () const**
  
  return the num_coords_per_field from sharedRespData

- **void set_coord_values (const RealMatrix &coord_values, const size_t i)**
  
  set the coordinate values per field

- **const RealMatrix & get_coord_values (const size_t i) const**
  
  return the coordinate values per field

- **const StringArray & function_labels () const**
  
  return the fine-grained (unrolled) response function identifier strings from sharedRespData

- **void function_labels (const StringArray &labels)**
  
  set the fine-grained (unrolled) response function identifier strings within sharedRespData

- **const StringArray & field_group_labels ()**
  
  return the user-provided field group labels instead of the unrolled labels available through function_labels()

- **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 (bool deep_srd=false) const**
  
  return a deep response copy of the contained responseRep for use in history mechanisms (SharedResponseData uses a shallow copy by default)

- **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 Response &response)**
Used in place of operator= when only results data updates are desired (functionValues/functionGradients/function-Hessians 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()**

  Resets all response data to zero

- **void reset_inactive()**

  Resets all inactive response data to zero

- **bool is_null()** const

  Function to check responseRep (does this handle contain a body)

- **virtual void set_scalar_covariance** (RealVector &scalars)

  Method to set the covariance matrix defined for ExperimentResponse

- **virtual Real get_scalar_covariance** (const int this_response)

  Method to get the covariance matrix defined for ExperimentResponse

- **virtual void set_full_covariance** (std::vector<RealMatrix> &matrices, std::vector<RealVector> &diagonals, RealVector &scalars, IntVector matrix_map_indices, IntVector diagonal_map_indices, IntVector scalar_map_indices)

  Method to set the full covariance matrices for ExperimentResponse

- **virtual Real apply_covariance** (const RealVector &residuals)

  Method to compute the triple product $v' \cdot \text{inv}(C) \cdot v$.

- **virtual void apply_covariance_inv_sqrt** (const RealVector &residuals, RealVector &weighted_residuals)

  Method to compute $(v' \cdot \text{inv}(C)^{-1/2})$, to compute weighted residual

- **virtual void apply_covariance_inv_sqrt** (const RealMatrix &gradients, RealMatrix &weighted_hessians)

- **virtual void apply_covariance_inv_sqrt** (const RealSymMatrixArray &hessians, RealSymMatrixArray &weighted_hessians)

- **virtual void get_covariance_diagonal** (RealVector &diagonal) const
Protected Member Functions

- **Response** (BaseConstructor, const Variables &vars, 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)

- **Response** (BaseConstructor, const SharedResponseData &srd, const ActiveSet &set)
  
  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

- **Response** (BaseConstructor, const ActiveSet &set)
  
  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

- **Response** (BaseConstructor, const SharedResponseData &srd)
  
  constructor initializes the base class part of letter classes (BaseConstructor overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

- virtual void copy_rep (Response *source_resp_rep)
  
  Implementation of data copy for Response letters (specialized by some derived letter types): pulls base class data from source_resp_rep into the this object.

Protected Attributes

- **SharedResponseData sharedRespData**
  
  reference-counted instance of shared response data: id’s, labels

- **RealVector functionValues**
  
  abstract set of response functions

- **RealMatrix functionGradients**
  
  first derivatives of the response functions

- **RealSymMatrixArray functionHessians**
  
  second derivatives of the response functions

- **IntRealMatrixMap fieldCoords**
  
  coordinates for the field values

- **ActiveSet responseActiveSet**
  
  copy of the ActiveSet used by the Model to generate a Response instance

Private Member Functions

- 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 Response from an archive<class Archive>
• template<class Archive>
  void load_rep (Archive &ar, const unsigned int version)
    read a Response letter object from an archive
• template<class Archive>
  void save (Archive &ar, const unsigned int version) const
    write a Response to an archive
• template<class Archive>
  void save_rep (Archive &ar, const unsigned int version) const
    write a Response letter object to an archive
• BOOST_SERIALIZATION_SPLIT_MEMBER () Response *get_response(short type
  Used by standard envelope constructor to instantiate a new letter class.
• Response * get_response (const SharedResponseData &srd, const ActiveSet &set) const
  Used by alternate envelope constructor to instantiate a new letter class.
• Response * get_response (short type, const ActiveSet &set) const
  Used by alternate envelope constructor to instantiate a new letter class.
• Response * get_response (const SharedResponseData &srd) const
  Used by copy() to instantiate a new letter class.
• Response * get_response (short type) const
  Used by read functions to instantiate a new letter class.
• void read_annotated_rep (std::istream &s)
  read a letter object in annotated format from an std::istream
• void write_annotated_rep (std::ostream &s) const
  write a letter object in annotated format to an std::ostream
• void read_rep (MPIUnpackBuffer &s)
  read a letter object from a packed MPI buffer
• void write_rep (MPIPackBuffer &s) const
  write a letter object to a packed MPI buffer
• void shape_rep (const ActiveSet &set, bool initialize=true)
  resizes the representation’s containers
• void reshape_rep (size_t num_fns, size_t num_params, bool grad_flag, bool hess_flag)
  resizes the representation’s containers

Private Attributes
  • const Variables & vars
  • const Variables const
    ProblemDescDB &problem_db const
  • Response * responseRep
    pointer to the body (handle-body idiom)
  • int referenceCount
    number of handle objects sharing responseRep
13.142. RESTARTWRITER CLASS REFERENCE

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

13.141.1 Detailed Description

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

The Response class is a container class for an abstract set of functions (functionValues) and their first (function-Gradients) 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). For memory efficiency, it employs the "letter-envelope idiom" approach to reference counting and representation sharing (see Coplien "Advanced C++"), for which the base Response class serves as the envelope and one of its derived classes serves as the letter.

13.141.2 Member Data Documentation

RealMatrix functionGradients [protected]

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 Response::field_gradients_view(), Response::function_gradient(), Response::function_gradient_copy(), Response::function_gradient_view(), Response::function_gradients(), and Response::function_gradients_view().

The documentation for this class was generated from the following file:

- DakotaResponse.hpp

13.142 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);

13.142.1 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

13.143 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 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)
13.143.1 Detailed Description

Class: ResultsDBAny Description: A map-based container to store DAKOTA Iterator results in underlying boost::any, with optional metadata

13.143.2 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

13.144 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.
13.145. RESULTSFILEERROR CLASS REFERENCE

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

13.144.1 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.

13.144.2 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

13.145 ResultsFileError Class Reference

exception throw for other results file read error
Inheritance diagram for ResultsFileError:

```
FileReadException
     ^
   ResultsFileError
```

Public Member Functions

- ResultsFileError (const std::string &msg)
CHAPTER 13. CLASS DOCUMENTATION

13.145.1 Detailed Description

exception throw for other results file read error

The documentation for this class was generated from the following file:

- dakota_global_defs.hpp

13.146 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 &methodName, const std::string &methodID)`
  
  explicitly increment the iterator results ID, init to 1 if needed

- `size_t get_id (const std::string &methodName, const std::string &methodID)`
  
  get (possibly creating) a unique iterator results ID for the passed name

- `size_t get_id (const std::string &methodName, const std::string &methodID) 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
13.146.1 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

13.147 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 is 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.

13.147.1 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

### 13.148 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 mapResp_prob`
- `std::string mapResp_rel`
- `std::string mapResp_genrel`
- `std::string mapProbResp`
- `std::string mapRelResp`
- `std::string mapGenrelResp`
- `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`
\begin{itemize}
\item \texttt{std::string pce\_coeff\_labels}
\item \texttt{std::string cv\_labels}
\item \texttt{std::string div\_labels}
\item \texttt{std::string dsv\_labels}
\item \texttt{std::string drv\_labels}
\item \texttt{std::string fn\_labels}
\end{itemize}

13.148.1 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:

\begin{itemize}
\item ResultsManager.hpp
\end{itemize}

13.149 RichExtrapVerification Class Reference

Class for Richardson extrapolation for code and solution verification.

Inheritance diagram for RichExtrapVerification:

\begin{center}
\begin{tikzpicture}
\node (RichExtrapVerification) {RichExtrapVerification};
\node (Verification) [above of=RichExtrapVerification] {Verification};
\node (Analyzer) [above of=Verification] {Analyzer};
\node (Iterator) [above of=Analyzer] {Iterator};
\path (Iterator) -- (Analyzer);
\path (Analyzer) -- (Verification);
\path (Verification) -- (RichExtrapVerification);
\end{tikzpicture}
\end{center}

Public Member Functions

\begin{itemize}
\item \texttt{RichExtrapVerification (ProblemDescDB \&problem\_db, Model \&model)} \textit{constructor}
\item \texttt{\~ RichExtrapVerification ()} \textit{destructor}
\item \texttt{void perform\_verification ()}
\hspace{1em} Redefines the \texttt{core\_run()} virtual function for the Verification branch.
\item \texttt{void print\_results (std::ostream \&s)}
\hspace{1em} print the final iterator results
\end{itemize}

Private Member Functions

\begin{itemize}
\item \texttt{void estimate\_order ()}
\hspace{1em} perform a single estimation of convOrder using extrapolation()
\item \texttt{void converge\_order ()}
\end{itemize}
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**

**13.149.1 Detailed Description**

Class for Richardson extrapolation for code and solution verification.

The `RichExtrapVerification` class contains several algorithms for performing Richardson extrapolation.

**13.149.2 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().

```cpp
void estimate_order( ) [private]
```

perform a single estimation of convOrder using extrapolation()

This algorithm executes a single refinement triple and returns convergence order estimates.


Referenced by RichExtrapVerification::perform_verification().

```cpp
void converge_order( ) [private]
```

iterate using extrapolation() until convOrder stabilizes

This algorithm continues to refine until the convergence order estimate converges.


Referenced by RichExtrapVerification::perform_verification().

```cpp
void converge_qoi( ) [private]
```

iterate using extrapolation() until QOIs stabilize

This algorithm continues to refine until the discretization error lies within a prescribed tolerance.


Referenced by RichExtrapVerification::perform_verification().

The documentation for this class was generated from the following files:

- RichExtrapVerification.hpp
- RichExtrapVerification.cpp

### 13.150 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 engine

13.150.1 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

13.151 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
13.151. SENSANALYSISGLOBAL CLASS REFERENCE

- 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 indicating whether correlations have been computed

Static Private Attributes

- static RealArray rawData = RealArray()
  
  array to hold temporary data before sort

13.151.1 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 PStudyD-ACE branch. Currently, the utility functions provide global sensitivity analysis through correlation calculations (e.g. simple, partial, rank, raw) as well as variance-based decomposition.

13.151.2 Member Function Documentation

void compute_correlations ( const VariablesArray & vars, 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, 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
13.152 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:

```
+-----------------+        +---+        +---+
| Iterator        |        | MetaIterator|<-----|
|                 |        |             |     |
+-----------------+        +---+        +---+
|                 |        | 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
- IntIntPair estimate_partition_bounds ()
  estimate the minimum and maximum partition sizes that can be utilized by this Iterator
- 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

13.152.1 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.

13.152.2 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::messagePass, SeqHybridMetaIterator::methodList, Iterator::methodPCIter, IteratorScheduler::miPLIndex, Iterator::num_final_solutions(), IteratorScheduler::numIteratorsJobs, 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

13.153 SerialDirectApplicInterface Class Reference

Sample derived interface class for testing serial simulator plug-ins using assign_rep().  

Inheritance diagram for SerialDirectApplicInterface:

```
Interface  
ApplicationInterface  
DirectApplicInterface  
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

13.153.1 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".

13.153.2 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

13.154 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 \textit{pre\_combine} (short corr\_type)
  aggregate the shared approximation data from current and saved states
• virtual void \textit{post\_combine} (short corr\_type)
  clean up saved storage after aggregation
• void \textit{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)
• \texttt{SharedApproxData * data\_rep} () const
  returns dataRep for access to derived class member functions that are not mapped to the top \texttt{SharedApproxData} level

\textbf{Protected Member Functions}

• \texttt{SharedApproxData (BaseConstructor, ProblemDescDB &problem\_db, size\_t num\_vars)}
  constructor initializes the base class part of letter classes (\texttt{BaseConstructor} overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)
• \texttt{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 (\texttt{BaseConstructor} overloading avoids infinite recursion in the derived class constructors - Coplien, p. 139)

\textbf{Protected Attributes}

• size\_t num\_Vars
  number of variables in the approximation
• String approx\_Type
  approximation type identifier
• short build\_Data\_Order
  order of the data used for surrogate construction, in ActiveSet request vector 3-bit format.
• short output\_Level
  output verbosity level: \{SILENT, QUIET, NORMAL, VERBOSE, DEBUG\}_{OUTPUT}
• \texttt{RealVector approxC\_Lower\_Bnds}
  approximation continuous lower bounds (used by 3D graphics and Surfpack KrigingModel)
• \texttt{RealVector approxC\_Upper\_Bnds}
  approximation continuous upper bounds (used by 3D graphics and Surfpack KrigingModel)
• \texttt{IntVector approxD\_L\_Bnds}
  approximation continuous lower bounds
• \texttt{IntVector approxD\_U\_Bnds}
  approximation continuous upper bounds
• \texttt{RealVector approxD\_RL\_Bnds}
  approximation continuous lower bounds
• \texttt{RealVector approxD\_RU\_Bnds}
  approximation continuous upper bounds
13.154. SHAREDPREXDATA CLASS REFERENCE

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 VPSApproximation
- class SurfpackApproximation
- class PecosApproximation

13.154.1 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.

13.154.2 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().

13.154.3 Member Function Documentation

SharedApproxData operator= (const SharedApproxData & shared_data)
assignment operator
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.
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().

13.154.4 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. auxiliary 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.
References SharedSurfpackApproxData::add_sd_to_srfdata(), TaylorApproximation::build(), TaylorApproximation::gradient(), TaylorApproximation::hessian(), TaylorApproximation::min_coefficients(), Approximation::min_points(), Approximation::recommended_points(), SharedApproxData::SharedApproxData(), SharedPecosApproxData::SharedPecosApproxData(), SurfpackApproximation::SurfpackApproximation(), SurfpackApproximation::surrogates_to_srfdata(), TANA3Approximation::TANA3Approximation(), and TaylorApproximation::value().
The documentation for this class was generated from the following files:

- SharedApproxData.hpp
- SharedApproxData.cpp

13.155 SharedPecosApproxData Class Reference

Derived approximation class for global basis polynomials.

Inheritance diagram for SharedPecosApproxData:

```
            SharedApproxData
               |
               v
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

13.155.1 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

13.156 SharedResponseData Class Reference

Container class encapsulating variables data that can be shared among a set of Response instances.

Public Member Functions

- SharedResponseData ()
  default constructor
- SharedResponseData (const ProblemDescDB &problem_db)
  standard constructor
- SharedResponseData (const ActiveSet &set)
  alternate on-the-fly constructor (explicit disallows its use for implicit type conversion so that instantiations of Response(set) are invalid)
- SharedResponseData (const SharedResponseData &srd)
  copy constructor
- ~SharedResponseData ()
  destructor
- SharedResponseData & operator= (const SharedResponseData &srd)
  assignment operator
- bool operator== (const SharedResponseData &other)
  experimental operator== for use in unit testing
- size_t num_scalar_responses () const
  number of scalar responses
- size_t num_field_response_groups () const
  number of field response groups
- size_t num_response_groups () const
  total number of response groups (scalars + field groups)
- size_t num_field_functions () const
  total number of field functions (1-norm of fieldRespGroupLengths)
- size_t num_functions () const
  total number of response functions (scalars + 1-norm of fieldRespGroupLengths)
- const IntVector & field_lengths () const
  index of field lengths for field data
• void field_lengths (const IntVector &field_lengths)
  set field lengths (if experiment different from simulation)
• const IntVector & num_coords_per_field () const
  dimensions of each function
• 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
• StringArray & function_labels ()
  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 StringArray & field_group_labels ()
  return the coarse (per-group) field response labels
• void field_group_labels (const StringArray &field_labels)
  set the coarse field group labels (must agree with number fields)
• const String & responses_id () const
  return the response identifier
• short response_type () const
  return the response type: \{BASE,SIMULATION,EXPERIMENT\}_RESPONSE
• void response_type (short type)
  set the response type: \{BASE,SIMULATION,EXPERIMENT\}_RESPONSE
• short primary_fn_type () const
  get the primary function type (generic, objective, calibration)
• void primary_fn_type (short type)
  set the primary function type (generic, objective, calibration)
• SharedResponseData copy () const
  create a deep copy of the current object and return by value
• void reshape (size_t num_fns)
  reshape the data, disconnecting a shared rep if necessary
• bool is_null () const
  return true if empty handle with null representation
• long reference_count () const
  how many handles (including this) are sharing this representation (body); for debugging/testing only
• template<class Archive>
  void serialize (Archive &ar, const unsigned int version)

Private Member Functions

• template<class Archive>
  void serialize (Archive &ar, const unsigned int version)

  serialize through the pointer, which requires object tracking: write and read are symmetric for this class
Private Attributes

- boost::shared_ptr<SharedResponseDataRep> srdRep
  
  pointer to the body (handle-body idiom)

Friends

- class boost::serialization::access
  
  allow boost access to serialize this class

13.156.1 Detailed Description

Container class encapsulating variables data that can be shared among a set of Response instances.

An array of Response objects (e.g., Analyzer::allResponse) contains repeated configuration data (id’s, labels, counts). SharedResponseData employs a handle-body idiom to allow this shared data to be managed in a single object with many references to it, one per Response object in the array. This allows scaling to larger sample sets.

13.156.2 Member Function Documentation

SharedResponseData 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 Response set.

References SharedResponseData::srdRep.

Referenced by ExperimentData::initialize(), and ExperimentData::load_data().

The documentation for this class was generated from the following files:

- SharedResponseData.hpp
- SharedResponseData.cpp

13.157 SharedResponseDataRep Class Reference

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

Public Member Functions

- ~SharedResponseDataRep ()
  
  destructor must be public for shared_ptr

- template<class Archive>
  
  void serialize (Archive &ar, const unsigned int version)

Private Member Functions

- SharedResponseDataRep ()
  
  default constructor

- SharedResponseDataRep (const ProblemDescDB &problem_db)
  
  standard constructor
The representation of a \texttt{SharedResponseData} instance. This representation, or body, may be shared by multiple \texttt{SharedResponseData} handle instances.

The \texttt{SharedResponseData/SharedResponseDataRep} pairs utilize a handle-body idiom (Coplien, Advanced C++).
13.157.2 Member Function Documentation

void copy_rep ( SharedResponseDataRep * srd_rep ) [private]

copy the data from srd_rep to the current representation

Deep copies are used when recasting changes the nature of a Response set.

References SharedResponseDataRep::fieldLabels, SharedResponseDataRep::fieldRespGroupLengths, SharedResponseDataRep::functionLabels, SharedResponseDataRep::numCoordsPerField, SharedResponseDataRep::numScalarResponses, SharedResponseDataRep::primaryFnType, SharedResponseDataRep::responsesId, and SharedResponseDataRep::responseType.

The documentation for this class was generated from the following files:

- SharedResponseData.hpp
- SharedResponseData.cpp

13.158 SharedSurfpackApproxData Class Reference

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

Inheritance diagram for SharedSurfpackApproxData:

```
SharedSurfpackApproxData
    SharedApproxData
```

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
- class VPSApproximation

Additional Inherited Members

13.158.1 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 adaptive regression splines (MARS).

13.158.2 Constructor & Destructor Documentation

SharedSurfpackApproxData ( const String & approx_type, const UShortArray & approx_order, size_t num_vers, 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

13.159 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_relaxed_di=BitArray(), const BitArray &all_relaxed_dr=BitArray())
  medium weight constructor providing detailed variable counts
- SharedVariablesData (const std::pair< short, short > &view, const SizetArray &vars_comps_totals, const BitArray &all_relaxed_di=BitArray(), const BitArray &all_relaxed_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_dsv, 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 \{\text{continuous,discrete integer,discrete real}\} \{\text{design,aleatory uncertain,epistemic uncertain,ate}

const SizetArray & active_components_totals () const
return active variable type counts for \{\text{continuous,discrete integer,discrete real}\} \{\text{design,aleatory uncertain,epistemic uncertain,ate}

const SizetArray & inactive_components_totals () const
return inactive variable type counts for \{\text{continuous,discrete integer,discrete real}\} \{\text{design,aleatory uncertain,epistemic uncertain,ate}

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
  retrieve 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 discrete 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
• template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

Private Member Functions
• template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

    serialize through the pointer, which requires object tracking: write and read are symmetric for this class

Private Attributes
• boost::shared_ptr
  < SharedVariablesDataRep > svdRep

    pointer to the body (handle-body idiom)

Friends
• class boost::serialization::access

    allow boost access to serialize this class
13.159.1 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.

13.159.2 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 SharedVariablesData::svdRep.

The documentation for this class was generated from the following files:

- SharedVariablesData.hpp
- SharedVariablesData.cpp

13.160 SharedVariablesDataRep Class Reference

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

Public Member Functions

- ~SharedVariablesDataRep ()
  destructor must be public for shared_ptr

- template<class Archive>
  void save (Archive &ar, const unsigned int version) const

- template<class Archive>
  void load (Archive &ar, const unsigned int version)

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

- 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_driv, size_t &num_drsv, size_t &num_drdrv) 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_dsv, 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 &numDrv) 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,Dl,DS,DR\}V

- void initialize_inactive_start_counts ()
  initialize i\{c,di,ds,dr\}vStart and numI\{D,Dl,DS,DR\}V

- void initialize_active_components ()
  initialize activeVarsCompsTotals given \{c,di,dr\}vStart and num\{C,Dl,DR\}V

- void initialize_inactive_components ()
  initialize inactiveVarsCompsTotals given i\{c,di,dr\}vStart and numI\{C,Dl,DR\}V

- size_t vc_lookup (unsigned short key) const
  retrieve the count within variablesComponents corresponding to key
• **void** `copy_rep` (SharedVariablesDataRep *svd_rep)
  
  copy the data from `svd_rep` to the current representation

• **template**<class Archive >
  
  **void** `save` (Archive &ar, const unsigned int version) const
  
  serialize the core shared variables data

• **template**<class Archive >
  
  **void** `load` (Archive &ar, const unsigned int version)
  
  load the core shared variables data and restore class state

• **BOOST_SERIALIZATION_SPLIT_MEMBER** () String variablesId
  
  variables identifier string from the input file

**Private Attributes**

• 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

**Friends**

- class `SharedVariablesData`
- class `boost::serialization::access`  
  allow boost access to serialize this class
13.160. **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++).

13.160.2 **Member Function Documentation**

```cpp
void copy_rep ( SharedVariablesDataRep * svd_rep ) [private]
```

copy the data from svd_rep to the current representation

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::numDSV, SharedVariablesDataRep::numICV, SharedVariablesDataRep::numIDIV, SharedVariablesDataRep::numIDSV, SharedVariablesDataRep::variablesComponents, SharedVariablesDataRep::variablesCompsTotals, and SharedVariablesDataRep::variablesView.

13.160.3 **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(), SharedVariablesDataRep::copy_rep(), 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 SharedVariablesDataRep::copy_rep(), 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 SharedVariablesDataRep::copy_rep(), 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 SharedVariablesDataRep::copy_rep(), and SharedVariablesDataRep::initialize_all_ids().
The documentation for this class was generated from the following files:

- SharedVariablesData.hpp
- SharedVariablesData.cpp

13.161 SimulationResponse Class Reference

Container class for response functions and their derivatives. SimulationResponse provides the body class.
Inheritance diagram for SimulationResponse:

```
Response

SimulationResponse
```

Public Member Functions

- **SimulationResponse ()**
  default constructor
-
- **SimulationResponse (const Variables &vars, const ProblemDescDB &problem_db)**
  standard constructor built from problem description database
-
- **SimulationResponse (const SharedResponseData &srd, const ActiveSet &set)**
  alternate constructor that shares a SharedResponseData instance
-
- **SimulationResponse (const SharedResponseData &srd)**
  alternate constructor that shares a SharedResponseData instance
-
- **SimulationResponse (const ActiveSet &set)**
  alternate constructor using limited data
-
- **~SimulationResponse ()**
  destructor
Additional Inherited Members

13.161.1 Detailed Description

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

The `SimulationResponse` 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 (`SimulationResponse`) actually contains the response data (functionValues, function-Gradients, functionHessians, etc.). The representation is hidden in that an instance of `SimulationResponse` 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).

The documentation for this class was generated from the following files:
- SimulationResponse.hpp
- SimulationResponse.cpp

13.162 SingleModel Class Reference

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

Inheritance diagram for SingleModel:

```
Model
    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_async_compute_response (const ActiveSet &set)`
  `portion of async_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 userDefined-
  Interface)
• IntIntPair estimate_partition_bounds (int max_eval_concurrency)
  estimate the minimum and maximum partition sizes that can be utilized by this Model
• 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 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

13.162.1 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.

13.162.2 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

13.163 SNLLBase Class Reference

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

Inheritance diagram for SNLLBase:

```
SNLLBase
   \-------
  |       |
  |       |
+-------+-------+
|       |       |
|       |       |
SNLLLeastSq SNLLOptimizer
```

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, size_t offset)
  convenience function for copying local_fn_grads to grad; used by constraint evaluator functions

• void copy_con_hess (const RealSymMatrixArray &local_fn_hessians, OPTPP::OptppArray< RealSymMatrix > &hess, size_t offset)
  convenience function for copying local_fn_hessians to hess; used by constraint evaluator functions

• void snll_preInstantiate (bool bound_constr_flag, int num_constr)
  convenience function for setting OPT++ options prior to the method instantiation

• void snll_postInstantiate (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_level, 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`

**13.163.1 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

**13.164 SNLLLeastSq Class Reference**

Wrapper class for the OPT++ optimization library.

Inheritance diagram for SNLLLeastSq:

```
    Iterator
     ↓
Minimizer
     ↓
Snake
     ↓
LeastSq  SNLLBase
     ↓
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_resid ()
  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, Real-SymMatrix &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 * prevSnllLeastSqInstance
  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
13.164. SNLLLEASTSQ CLASS REFERENCE

- 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

13.164.1 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.

13.164.2 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,
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, $f(x) = \sum (T_i - T_{bar,i})^2$ 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::snllLSqInstance, 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::snllLSqInstance.

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 active.

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-
13.165. SNLLOptimizer Class Reference

Wrapper class for the OPT++ optimization library.

Inheritance diagram for SNLLOptimizer:

```
Iterator

Minimizer

Optimizer  SNLLBase
```

**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 &nlm_ineq_l_bnds, const RealVector &nlm_ineq_u_bnds, const RealVector &nlm_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::NLF * 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

13.165.1 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`, and `search_scheme_size` are set using OPT++’s `setMaxIter()`, `setMaxEval()`, `setConvTol()`, `setMaxStep()`, `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.

13.165.2 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, 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(), SNLLOptimizer::optbcnewton, SNLLOptimizer::optbcqnnewton, SNLLOptimizer::optcg, SNLLOptimizer::optfdnewton, SNLLOptimizer::optlbfgs, SNLLOptimizer::optnewton, SNLLOptimizer::optnips, SNLLOptimizer::optpds, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, Iterator::outputLevel, Iterator::probDescDB, SNLLOptimizer::searchStrat, SNLLOptimizer::nlpConstraint, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, SNLLOptimizer::optbcfdnewton, SNLLOptimizer::optfdnips, SNLLOptimizer::optlbfgs, SNLLOptimizer::optnewton, SNLLOptimizer::optpds, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, Iterator::outputLevel, Iterator::probDescDB, SNLLOptimizer::nlpConstraint, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, SNLLOptimizer::optbcfdnewton, SNLLOptimizer::optfdnips, SNLLOptimizer::optlbfgs, SNLLOptimizer::optnewton, SNLLOptimizer::optpds, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, Iterator::outputLevel, Iterator::probDescDB, SNLLOptimizer::nlpConstraint, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, SNLLOptimizer::optbcfdnewton, SNLLOptimizer::optfdnips, SNLLOptimizer::optlbfgs, SNLLOptimizer::optnewton, SNLLOptimizer::optpds, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, Iterator::outputLevel, Iterator::probDescDB.

**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(), SNLLOptimizer::optbcnewton, SNLLOptimizer::optbcqnnewton, SNLLOptimizer::optcg, SNLLOptimizer::optfdnewton, SNLLOptimizer::optlbfgs, SNLLOptimizer::optnewton, SNLLOptimizer::optpds, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, Iterator::outputLevel, Iterator::probDescDB, SNLLOptimizer::searchStrat, SNLLOptimizer::nlpConstraint, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, SNLLOptimizer::optbcfdnewton, SNLLOptimizer::optfdnips, SNLLOptimizer::optlbfgs, SNLLOptimizer::optnewton, SNLLOptimizer::optpds, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, Iterator::outputLevel, Iterator::probDescDB.
13.165. SNLLOPTIMIZER CLASS REFERENCE

Name, SNLLOptimizer::nlf1, SNLLOptimizer::nlf1_evaluator(), SNLLOptimizer::nlf1Con, SNLLOptimizer::nlfConstraint, SNLLOptimizer::nlfObjective, SNLLOptimizer::nlfConstraint, Minimizer::numConstraints, Minimizer::numContinuousVars, Minimizer::numNonlinearConstraints, SNLLOptimizer::optbcqnewton, SNLLOptimizer::optlbfgs, SNLLOptimizer::optqnips, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, SNLLOptimizer::theOptimizer, and Minimizer::vendorNumericalGradFlag.

SNLLOptimizer ( const RealVector & initial pt, const RealVector & var lbnds, const RealVector & var ubnds, const RealMatrix & lin ineq coeffs, const RealVector & lin ineq lbnds, const RealVector & lin ineq ubnds, const RealMatrix & lin eq coeffs, const RealVector & lin eq tgts, const RealVector & nln ineq lbnds, const RealVector & nln ineq ubnds, 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, double &f, RealVector &grad f, 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::optqnips, SNLLOptimizer::optqnewton, SNLLOptimizer::optqnips, 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 Minimizer::vendorNumericalGradFlag.

13.165.3 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).
13.166 SOLBase Class Reference

Base class for Stanford SOL software.

Inheritance diagram for SOLBase:

```
SOLBase
   \--------
   |        |
NLSSOLLeastSq NPSOLOptimizer
```

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 &dss)**
  *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: Lagrange 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
  

- 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*

**13.166.1 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

**13.167 SpawnApplicInterface Class Reference**

Derived application interface class which spawns simulation codes using spawnvp.

Inheritance diagram for SpawnApplicInterface:

```
  Interface
    ApplicationInterface
      ProcessApplicInterface
        ProcessHandleApplicInterface
          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

13.167.1 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

13.168 SurfpackApproximation Class Reference

Derived approximation class for Surfpack approximation classes. Interface between Surfpack and Dakota.
Inheritance diagram for SurfpackApproximation:

```
Approximation

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)

- Real value (const RealVector &c_vars)
  
  Return the value of the Surfpack surface for a given parameter vector x.

- const RealVector & gradient (const RealVector &c_vars)
  
  retrieve the approximate function gradient for a given parameter vector x

- const RealSymMatrix & hessian (const RealVector &c_vars)
  
  retrieve the approximate function Hessian for a given parameter vector x

- Real prediction_variance (const RealVector &c_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

13.168.1 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 adaptive regression splines (MARS).

13.168.2 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, Shared-ApproxData::outputLevel, and Approximation::sharedDataRep.

13.168.3 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::approxDLowerBnds, SharedApproxData::approxDUpperBnds, 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().

const RealSymMatrix & hessian (const RealVector & c 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, SurfpackApproximation::model, and Approximation::sharedDataRep.

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::surrogates_to_surf_data().

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:
13.169 **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:

```
  Iterator
     |_________________
     |                
     |               
     Minimizer
     |             
     |            
SurrBasedMinimizer
     |             
     |            
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

13.169.1 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

13.170 SurrBasedLocalMinimizer Class Reference

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

Inheritance diagram for SurrBasedLocalMinimizer:

```
Iterator
    ↓
Minimizer
    ↓
SurrBasedMinimizer
    ↓
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
13.170. SURRBASEDLOCALMINIMIZER CLASS REFERENCE

Additional Inherited Members

13.170.1 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.

13.170.2 Member Function Documentation

```cpp
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(), Response::active_set(), iterator::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(), local::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::origNonlinIneqLowerBounds(), SurrBasedMinimizer::origNonlinIneqUpperBounds(), ParallelLibrary::output_manager(), iterator::parallelLib(), SurrBasedLocalMinimizer::recastSubProb(), SurrBasedLocalMinimizer::release_constraints(), ActiveSet::request_values(), SurrBasedLocalMinimizer::reset(), iterator::response_results(), SurrBasedLocalMinimizer::response_center_approx(), SurrBasedLocalMinimizer::response_center_true(), SurrBasedLocalMinimizer::response_star_approx(), SurrBasedLocalMinimizer::response_star_truth(), iterator::run(), iterator::sampling_scheme(), SurrBasedMinimizer::sbIterNum(), SurrBasedLocalMinimizer::sbmlInstance(), 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::trust_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_yars, 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, SurrBasedLocalMinimizer::constraintViolation(), 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(), SurrBasedLocalMinimizer::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_yars_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, SurrBasedLocalMinimizer::constraintViolation(), 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::constraintViolation(), 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::sblmInstance.

Referenced by SurrBasedLocalMinimizer::tr_ratio_check().

```cpp
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::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::origNonlinEqUpperBnds, SurrBasedMinimizer::origNonlinIneqLowerBnds, SurrBasedMinimizer::origNonlinIneqUpperBnds, Model::primary_response_fn_sense(), Model::primary_response_fn_weights(), and SurrBasedLocalMinimizer::sblmInstance.

Referenced by SurrBasedLocalMinimizer::SurrBasedLocalMinimizer().

```cpp
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().

```cpp
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().

```cpp
void hom_constraint_eval ( int & mode, int & ncnln, int & n, int & nrowj, int * needc, double * tau_and_x, double * c, double * jac, 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

### 13.171 SurrBasedMinimizer Class Reference

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

Inheritance diagram for SurrBasedMinimizer:

```
  +--- SurrBasedMinimizer
  |    +--- Minimizer
  |    |    +--- Iterator
  |    |    |    +--- EffGlobalMinimizer
  |    |    |    +--- SurrBasedGlobalMinimizer
  |    |    |    +--- SurrBasedLocalMinimizer
```

#### 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_eqtgt)
  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_eqtgt, 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_eqtgt)
  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_eqtgt, 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

#### 13.171.1 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.

#### 13.171.2 Member Function Documentation

```cpp
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().

```cpp
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::constraintViolation(), 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_jineq_l_bnds, const RealVector & nln_jineq_u_bnds, const RealVector & nln_eq_lfts ) [protected]
compute a Lagrangian function from a set of function values

The Lagrangian function computation sums the objective function and the Lagrange multiplier 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_lagrangianMerit (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 multiplier terms for
inequality/equality constraints, and quadratic penalty terms for inequality/equality constraints. This implemen-
tation 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::numNonlinear-
EqConstraints, 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 penaltyMerit (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 $cv = g+^T h+$. 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::numNonlinear-
IneqConstraints, Minimizer::numUserPrimaryFns, SurrBasedMinimizer::origNonlinEqTargets, SurrBasedMinimizer-
::origNonlinIneqLowerBnds, and SurrBasedMinimizer::origNonlinIneqUpperBnds.

Referenced by SurrBasedLocalMinimizer::hard_convergence_check(), EffGlobalMinimizer::minimize_surrogates-
on_model(), SurrBasedMinimizer::penaltyMerit(), 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

13.172 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 SharedResponseData &srd, 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::actual\_Model 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::miPLIers`) 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 referenceDILBnds
  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

13.172.1 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.

13.172.2 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(), Variables::discrete_real_variables(), Constraints::discrete_real_upper_bounds(), Model::discrete_real_upper_bounds(), 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::referenceDRLVars, Dakota::strbegins(), Model::subordinate_model(), Model::surrogateType, Model::truth_model(), Surrogate-
Model::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().

### 13.172.3 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

### 13.173 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

13.173.1 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.

13.173.2 Member Function Documentation

void wait_local_evaluations ( PRPQueue & prp_queue ) [inline], [protected], [virtual]

Check for completion of active async jobs (tracked with sysCallSet). Wait for at least one completion and complete all jobs that have returned. This satisfies 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 async 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

13.174 TabularDataTruncated Class Reference

exception thrown when data read truncated

Inheritance diagram for TabularDataTruncated:

```
FileReadException

TabularDataTruncated
```

Public Member Functions

- **TabularDataTruncated** (const std::string &msg)

13.174.1 Detailed Description

exception thrown when data read truncated

The documentation for this class was generated from the following file:

- dakota_global_defs.hpp

13.175 TANA3Approximation Class Reference

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

Inheritance diagram for TANA3Approximation:

```
Approximation

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

13.175.1 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.

13.175.2 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

13.176 TaylorApproximation Class Reference

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

Inheritance diagram for TaylorApproximation:

```
Approximation
  \--- TaylorApproximation
```

Public Member Functions

- `TaylorApproximation()`
  
  default constructor

- `TaylorApproximation(ProblemDescDB &problem_db, const SharedApproxData &shared_data)`
  
  standard constructor

- `TaylorApproximation(const SharedApproxData &shared_data)`
Protected Member Functions

- int min_coefficients () const
  
  return the minimum number of samples (unknowns) required to build the derived class approximation type in num-Vars 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

13.176.1 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 \( \text{grad}(x-x_c)'(x-x_c) \) for first- and second-order, and plus \( (x-x_c)'\text{Hess}(x_c)(x-x_c)/2 \) for second-order.

13.176.2 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

13.177 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 damped\_oscillator ()
  1d-6d that returns field values (ode solution)
- 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 ()
CHAPTER 13. CLASS DOCUMENTATION

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 > &dlw, 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

13.177.1 Detailed Description

Specialization of DirectApplicInterface to embed algebraic test function drivers directly in Dakota

13.177.2 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(), TestDriverInterface::damped_oscillator(), 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::analysis-
DriverIndex, Dakota::dc_ptr_int, DirectApplicInterface::directFnASV, Interface::fnLabels, DirectApplicInterface-
::fnVals, Dakota::mc_ptr_int, ApplicationInterface::multiProcAnalysisFlag, DirectApplicInterface::numACV, Direct-
ApplicInterface::numADIV, DirectApplicInterface::numADR, DirectApplicInterface::numFns, DirectApplicInterface-
::xC, DirectApplicInterface::xCLabels, DirectApplicInterface::xDI, DirectApplicInterface::xDILabels, DirectApplic-
Interface::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

13.178 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,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 &urlpost) 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

13.178.1 Detailed Description

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

13.178.2 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

13.179 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)

13.179.1 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...
13.179.2 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

13.180 Var_ichk 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

13.180.1 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

13.181 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
13.181.1 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

13.182 Variables Class Reference

Base class for the variables class hierarchy.

Inheritance diagram for Variables:

```
Variables
  MixedVariables
  RelaxedVariables
```

Public Member Functions

- **Variables ()**
  - default constructor
- **Variables (const ProblemDescDB &problem_db)**
  - standard constructor (explicit disallows its use for implicit type conversion)
- **Variables (const SharedVariablesData &svd)**
  - alternate constructor for instantiations on the fly (explicit disallows its use for implicit type conversion)
- **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 ostream
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 string variable labels
- `void inactive_discrete_string_variable_labels (StringMultiArrayConstView ids_vars)` set the inactive discrete string variable labels
- `StringMultiArrayConstView inactive_discrete_real_variable_labels () const` return the inactive discrete real variable labels
- `void inactive_discrete_real_variable_labels (StringMultiArrayConstView idr_vars)` set the inactive discrete real 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
- `SizeMatMultiArrayConstView 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
13.182. VARIABLES CLASS REFERENCE

- **Variable MultiArrayConstView**
  all_continuous_variable_jds () const
  return all continuous variable position identifiers

- **Variables copy**
  (bool deep_svd=false) const
  a deep variables copy for use in history mechanisms (SharedVariablesData uses a shallow copy by default)

- **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) const**
  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 \texttt{continuousVars} \\
  the active continuous variables array view
• IntVector \texttt{discreteIntVars} \\
  the active discrete integer variables array view
• RealVector \texttt{discreteRealVars} \\
  the active discrete real variables array view
• RealVector \texttt{inactiveContinuousVars} \\
  the inactive continuous variables array view
• IntVector \texttt{inactiveDiscreteIntVars} \\
  the inactive discrete integer variables array view
• RealVector \texttt{inactiveDiscreteRealVars} \\
  the inactive discrete real variables array view

**Private Member Functions**

• \texttt{Variables *} \texttt{get\_variables} (const \texttt{ProblemDescDB} \&\texttt{problem\_db}) \\
  \textit{Used by the standard envelope constructor to instantiate the correct letter class.}
• \texttt{Variables *} \texttt{get\_variables} (const \texttt{SharedVariablesData} \&\texttt{svd}) const \\
  \textit{Used by the alternate envelope constructors, by read functions, and by \texttt{copy()}} to instantiate a new letter class.
• short method \texttt{map} (short \texttt{view\_spec, bool relaxed}) const \\
  \texttt{infer domain from method selection}
• short method \texttt{domain} (const \texttt{ProblemDescDB} \&\texttt{problem\_db}) const \\
  \texttt{infer domain from method selection}
• short method \texttt{view} (const \texttt{ProblemDescDB} \&\texttt{problem\_db}) const \\
  \texttt{infer view from method selection}
• short response \texttt{view} (const \texttt{ProblemDescDB} \&\texttt{problem\_db}) const \\
  \texttt{infer view from type of response data set}
• void \texttt{check\_view\_compatibility} () \\
  \texttt{perform sanity checks on view\_first and view\_second after update}
• \texttt{template}\langle\texttt{class Archive} \rangle \\
  void \texttt{load} (\texttt{Archive} \&\texttt{ar, const unsigned int version}) \\
  \texttt{read a Variables object from an archive}
• \texttt{template}\langle\texttt{class Archive} \rangle \\
  void \texttt{save} (\texttt{Archive} \&\texttt{ar, const unsigned int version}) const \\
  \texttt{write a Variables object to an archive}
• \texttt{BOOST\_SERIALIZATION\_SPLIT\_MEMBER} () \texttt{Variables *}\texttt{variablesRep} \\
  \texttt{pointer to the letter (initialized only for the envelope)}

**Private Attributes**

• int \texttt{referenceCount} \\
  \texttt{number of objects sharing variables\_rep}
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

13.182.1 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.

13.182.2 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

13.183 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

13.183.1 Detailed Description

Base class for managing common aspects of verification studies.

The Verification base class manages common data and functions, such as those involving ...

13.183.2 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.
13.184 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

13.184.1 Detailed Description

structure for validating integer uncertain variable labels, bounds, values

The documentation for this struct was generated from the following file:

- NIDRProblemDescDB.cpp

13.185 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

13.185.1 Detailed Description

structure for validating real uncertain variable labels, bounds, values

The documentation for this struct was generated from the following file:

- NIDRProblemDescDB.cpp

Reimplemented in RichExtrapVerification.
References Analyzer::print_result().
Referenced by RichExtrapVerification::print_result().
The documentation for this class was generated from the following files:

- DakotaVerification.hpp
- DakotaVerification.cpp
13.186 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

13.186.1 Detailed Description

structure for validating string uncertain variable labels, bounds, values

The documentation for this struct was generated from the following file:

- NIDRProblemDescDB.cpp

13.187 VPSApproximation Class Reference

Derived approximation class for VPS implementation.

Inheritance diagram for VPSApproximation:

```
Approximation
```

```
VPSApproximation
```

Public Member Functions

- VPSApproximation ()  
  *default constructor*
- VPSApproximation (const ProblemDescDB &problem_db, const SharedApproxData &shared_data)  
  *standard constructor (to call VPS from an input deck)*
- VPSApproximation (const SharedApproxData &shared_data)  
  *Alternate constructor (to call VPS from another method like POF-darts)*
- ~VPSApproximation ()  
  *destructor*
- bool VPS_execute ()
- void VPS_create_containers ()
- void VPS_retrieve_neighbors (size_t ipoint, bool update_point_neighbors)
- void VPS_adjust_extend_neighbors_of_all_points ()
- void VPS_extend_neighbors (size_t ipoint)
VPS APPROXIMATION CLASS REFERENCE

- **void VPS_build_local_surrogate(size_t cell_index)**
- **double VPS_evaluate_surrogate(double *x)**
- **void VPS_destroy_global_containers()**
- **void retrieve_permutations(size_t &m, size_t **&perm, size_t num_dim, size_t upper_bound, bool force_sum_constraint, size_t sum_constraint)**
- **void build_radial_basis_function(size_t icell)**
- **void VPS_LS_retrieve_weights(size_t cell_index)**
- **double evaluate_basis_function(double *x, size_t icell, size_t ibasis)**
- **int constrained_LeastSquare(size_t n, size_t m, double **H, double *w, double *f)**
- **double vec_dot_vec(size_t n, double *vec_a, double *vec_b)**
- **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)**
- **void printMatrix(size_t m, size_t n, double **M)**
- **void initiate_random_number_generator(unsigned long x)**
- **double generate_a_random_number()**
- **size_t retrieve_closest_cell(double *x)**
- **bool trim_line_using_Hyperplane(size_t num_dim, double *st, double *end, double *qH, double *nH)**
- **double f_test(double *x)**
- **void generate_poisson_disk_sample(double r)**
- **void generate_MC_sample()**
- **void isocontouring(std::string file_name, bool plot_test_function, bool plot_surrogate, std::vector<double> contours)**
- **void isocontouring_solid(std::string file_name, bool plot_test_function, bool plot_surrogate, std::vector<double> contours)**
- **void plot_neighbors()**

**Protected Member Functions**

- **int min_coefficients() const**
  
  return the minimum number of samples (unknowns) required to build the derived class approximation type in num-Vars 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 predicted 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
CHAPTER 13. CLASS DOCUMENTATION

Private Types

- enum subsurrogate { LS, GP }
- enum subsurrogate_basis { polynomial, radial }
- enum testfunction { SmoothHerbie, Herbie, UnitSphere }

Private Member Functions

- void VPSmodel_build ()
  
  Function to compute coefficients governing the VPS surrogates.
- void VPSmodel_apply (const RealVector &new_x, bool variance_flag, bool gradients_flag)
  
  Function returns a response value using the VPS surface.

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.
- size_t numObs
  
  The number of observations on which the GP surface is built.
- int surrogateOrder
  
  The order of the polynomial in each Voronoi cell.
- subsurrogate vps_subsurrogate
- subsurrogate_basis vps_subsurrogate_basis
- testfunction vps_test_function
- double Q [1220]
- int index
- double cc
- double c
- double zc
- double zx
- double zy
- size_t qlen
- size_t n_dim
- double _xmin
- double _xmax
- double _diag
- size_t num_inserted_points
- double ** sample_points
- double * fval
- size_t ** sample_neighbors
13.187. VPSAPPROXIMATION CLASS REFERENCE

- size_t **_vps_ext_neighbors
- size_t _vps_order
- size_t _num_GMRES
- size_t * _num_cell_basis_functions
- double * _sample_vsize
- double * _vps_dfar
- double *** _sample_basis
- double _max_vsize
- double _disc_min_grad
- double _f_min
- double _f_max
- size_t ***_vps_t
- double ** _vps_w
- SharedApproxData sharedData
- std::vector< Approximation > gpApproximations
- Variables gpEvalVars

Static Private Attributes

- static VPSApproximation * VPSinstance  
  pointer to the active object instance used within the static evaluator

Additional Inherited Members

13.187.1 Detailed Description

Derived approximation class for VPS implementation.

The VPSApproximation class provides a set of piecewise surrogate approximations each of which is valid within a Voronoi cell.

13.187.2 Member Function Documentation

void VPSmodel_apply ( const RealVector & new_x, bool variance flag, bool gradients flag ) [private]

Function returns a response value using the VPS surface.
  The response value is computed at the design point specified by the RealVector function argument.
  References VPSApproximation::approxValue.
  Referenced by VPSApproximation::gradient(), VPSApproximation::prediction_variance(), and VPSApproximation::value().

13.187.3 Member Data Documentation

VPSApproximation * VPSinstance [static],[private]

pointer to the active object instance used within the static evaluator
default constructor
  The documentation for this class was generated from the following files:
  - VPSApproximation.hpp
  - VPSApproximation.cpp
13.188 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 SPWD.

- 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 &wildcard)
  
  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 wildcard 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_items, 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)
  Recursive 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=.

13.188.1 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.

13.188.2 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::startup-
  PWD.
  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 & wildcard ) [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 wildcard as path to reduce wstring conversions

Input: path_with_wc; Output: search_dir, wildcard

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]

Recursivce 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().

```cpp
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().

```cpp
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().

```cpp
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 ()
CHAPTER 14. FILE DOCUMENTATION

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
14.3. DAKOTA_TABULAR_IO.HPP FILE REFERENCE

- 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 facilitate 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
- String format_name (unsigned short tabular_format)
  Translate tabular_format into a user-friendly name.
- void print_expected_format (std::ostream &s, unsigned short tabular_format, size_t num_rows, size_t num_cols)
  Describe the expected data file format based on passed parameters.
- void print_unexpected_data (std::ostream &s, const String &filename, const String &context_message, unsigned short tabular_format)
  Print a warning if there’s extra data in the file.
- 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 &response, const std::string &counter_label, unsigned short tabular_format)

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, unsigned short tabular_format)

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, unsigned short tabular_format)

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, unsigned short tabular_format)

read and discard header line from the stream

• size_t **read_leading_columns** (std::istream &input_stream, unsigned short tabular_format)

read leading columns [ int eval_id [ String iface_id ] ]

• void **read_data_tabular** (const std::string &input_filename, const std::string &context_message, RealVectorArray &input_data, size_t num_entries, unsigned short tabular_format)

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, unsigned short tabular_format, 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, unsigned short tabular_format, 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, unsigned short tabular_format, 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, unsigned short tabular_format, 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, unsigned short tabular_format, bool active_only, Variables vars)

read specified input data file into arrays with sizes specified by the passed vc_totals array; used in ParamStudy
14.4 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.

14.5 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_AS ()
- 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::method_Name, Environment::mpi_manager(), MPIManager::mpirun_flag(), ParallelLibrary::mpirun_flag(), DataVariablesRep::numContinuousDesVars, DataResponsesRep::numNonlinearIneqConstraints, DataResponsesRep::numObjective-Functions, parallel_interface_plugin(), Environment::parallel_library(), serial_interface_plugin(), and ParallelLibrary::world_rank().

Referenced by main().

```cpp
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().

```cpp
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().

```cpp
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_text_book” 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().
14.7. LIBRARY_MODE.CPP FILE REFERENCE

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_g_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:
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.
14.10. **RESTART_UTIL.CPP FILE REFERENCE**

### 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\_usage (std::ostream &s)**

  *Print restart utility help message*

- **void print\_restart (StringArray pos\_args, String print\_dest)**

  *Print a restart file*

- **void print\_restart\_pdb (StringArray pos\_args, String print\_dest)**

  *Print a restart file (PDB format)*

- **void print\_restart\_tabular (StringArray pos\_args, String print\_dest, unsigned short tabular\_format)**

  *Print a restart file (tabular format)*

- **void read\_neutral (StringArray pos\_args)**

  *Read a restart file (neutral file format)*

- **void repair\_restart (StringArray pos\_args, String identifier\_type)**

  *Repair a restart file by removing corrupted evaluations*

- **void concatenate\_restart (StringArray pos\_args)**
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 (\texttt{print\_restart()}, \texttt{print\_restart\_tabular()}, \texttt{read\_neutral()}, \texttt{repair\_restart}(), or \texttt{concatenate\_restart}()).

References Dakota::concatenate\_restart(), Dakota::print\_restart, Dakota::print\_restart\_pdb, Dakota::print\_\_restart\_tabular, Dakota::print\_usage, Dakota::read\_neutral, and Dakota::repair\_restart.
Index

~Approximation
Dakota::Approximation, 355

~Constraints
Dakota::Constraints, 398

~DataFitSurrModel
Dakota::DataFitSurrModel, 408

~EffGlobalMinimizer
Dakota::EffGlobalMinimizer, 474

~Environment
Dakota::Environment, 483

~Interface
Dakota::Interface, 527

~Iterator
Dakota::Iterator, 537

~Model
Dakota::Model, 602

~NonDAdaptiveSampling
Dakota::NonDAdaptiveSampling, 666

~ProblemDescDB
Dakota::ProblemDescDB, 807

~SharedApproxData
Dakota::SharedApproxData, 872

_initPts
Dakota::JEGAOptimizer, 554

_model
Dakota::JEGAOptimizer::Evaluator, 491

A
Dakota::CONMINOptimizer, 389
APPSEvalMgr, 363
Dakota::APPSEvalMgr, 365
APPSOptimizer, 365
abort_handler.t
Dakota, 151
abort_mode
Dakota, 156
accepts_multiple_points
Dakota::JEGAOptimizer, 553
ActiveSet, 327
activeVarsCompsTotals
Dakota::SharedVariablesDataRep, 891
actualModel
Dakota::DataFitSurrModel, 415
add_anchor_to_surfdata
Dakota::SurfpackApproximation, 913
add_datapoint
Dakota::Graphics, 514
Dakota::OutputManager, 768
algorithm_space_model
Dakota::Analyzer, 334
Dakota::EffGlobalMinimizer, 474
Dakota::Minimizer, 572
Dakota::NonDBayesCalibration, 668
Dakota::NonDExpansion, 681
Dakota::NonDGlobalInterval, 687
Dakota::NonDReliability, 741
allContinuousIds
Dakota::SharedVariablesDataRep, 892
Analyzer, 329
anisotropic_order_to_dimension_preference
Dakota::NonDIntegration, 701
append_approximation
Dakota::ApproximationInterface, 361
Dakota::DataFitSurrModel, 411, 412
ApplicationInterface, 336
approx_subprob_constraint_eval
Dakota::SurrogateLocalMinimizer, 921
approx_subprob_objective_eval
Dakota::SurrogateLocalMinimizer, 921
approxBuilds
Dakota::SurrogateModel, 930
Approximation, 350
Dakota::Approximation, 354, 355
ApproximationInterface, 357
array_insert
Dakota::ResultsDBAny, 852
assess_reconstruction
Dakota::EfficientSubspaceMethod, 477
assign_rep
Dakota::Interface, 528
Dakota::Iterator, 540
Dakota::Model, 607
asstring Dakota, 153
asynchronous_local_analyses Dakota::ProcessHandleApplicInterface, 817
asynchronous_local_evaluations Dakota::ApplicationInterface, 348
asynchronous_local_evaluations_nowait Dakota::ApplicationInterface, 349
augmented_lagrangian_merit Dakota::SurrBasedMinimizer, 925
autotag_files Dakota::ProcessApplicInterface, 812

B Dakota::CONMINOptimizer, 388
BaseConstructor, 368
brent_minimize Dakota::NonlinearCGOptimizer, 756
build Dakota::Approximation, 355
Dakota::PecosApproximation, 798
Dakota::SurfpackApproximation, 912
Dakota::TANA3Approximation, 936
Dakota::TaylorApproximation, 937
build_approximation Dakota::ApproximationInterface, 361
Dakota::DataFitSurrModel, 410
build_global Dakota::DataFitSurrModel, 413
build_hessian_of_sum_square_residuals_from_function_data Dakota::ExperimentData, 497
build_local_multipoint Dakota::DataFitSurrModel, 414
build_views Dakota::Constraints, 399
Dakota::Variables, 955
buildDataOrder Dakota::SharedApproxData, 873

C Dakota::CONMINOptimizer, 388
CAUVLbl Dakota, 319
CEUVLbl Dakota, 320
COLINAApplication, 369
CONMINOptimizer, 372
Dakota::COLINOptimizer, 374

CT Dakota::CONMINOptimizer, 388
callback_data, 368
callback_function Dakota::NIDRProblemDescDB, 634
corr Dakota::ProblemDescDB, 808
corr_wait Dakota::ProcessHandleApplicInterface, 816
clear Dakota::ParallelLevel, 776
clear_all Dakota::Approximation, 357
clear_current Dakota::Approximation, 356
Dakota::TANA3Approximation, 936
Clone Dakota::JEGAOptimizer::Evaluator, 490
colin_cache_lookup Dakota::COLINOptimizer, 375
colin_request_to_dakota_request Dakota::COLINApplication, 371
CollabHybridMetaIterator, 376
collect_evaluation_impl Dakota::COLINApplication, 371
CommandLineHandler, 377
CommandShell, 379
compute Dakota::DiscrepancyCorrection, 466
compute_correlations Dakota::SensAnalysisGlobal, 863
compute_covariance_metric Dakota::NonDExpansion, 681
Dakota::NonDStochCollocation, 753
INDEX

compute_final_statistics_metric
    Dakota::NonDExpansion, 681
    Dakota::NonDStochCollocation, 753
compute_statistics
    Dakota::NonDExpansion, 682
concatenate_restart
    Dakota, 155
ConcurrentMetaIterator, 380
configure
    Dakota::IteratorScheduler, 547
conminInfo
    Dakota::CONMINOptimizer, 386
ConsoleRedirector, 390
constraint0_evaluator
    Dakota::SNLLLOptimizer, 906
constraint1_evaluator
    Dakota::SNLLLOptimizer, 906
constraint1_evaluator_gn
    Dakota::SNLLLeastSq, 900
constraint2_evaluator
    Dakota::SNLLLOptimizer, 906
constraint2_evaluator_gn
    Dakota::SNLLLeastSq, 900
constraint_violation
    Dakota::COLINOptimizer, 375
    Dakota::SurrBasedMinimizer, 926
constraintMappingIndices
    Dakota::CONMINOptimizer, 387
    Dakota::DOTOptimizer, 470
constraintMappingMultipliers
    Dakota::CONMINOptimizer, 387
    Dakota::DOTOptimizer, 470
constraintMappingOffsets
    Dakota::CONMINOptimizer, 387
    Dakota::DOTOptimizer, 470
constraintValues
    Dakota::CONMINOptimizer, 387
    Dakota::DOTOptimizer, 470
Constraints, 391
    Dakota::Constraints, 397, 398
contains
    Dakota, 155
continuous_probability_density
    Dakota::Model, 607
converge_order
    Dakota::RichExtrapVerification, 860
converge_qoi
    Dakota::RichExtrapVerification, 860
copy
    Dakota::Constraints, 398
    Dakota::SharedResponseData, 878
    Dakota::SharedVariablesData, 887
copy_data
    Dakota::DDACEDesignCompExp, 459
copy_items
    Dakota::WorkdirHelper, 965
copy_rep
    Dakota::SharedResponseDataRep, 880
    Dakota::SharedVariablesDataRep, 891
core_run
    Dakota::Iterator, 538
    Dakota::LeastSq, 556
    Dakota::NonD, 656
    Dakota::Optimizer, 764
    Dakota::PStudyDACE, 824
    Dakota::SurrBasedMinimizer, 924
    Dakota::Verification, 956
create_command_arguments
    Dakota::ProcessHandleApplicInterface, 817
create_directory
    Dakota::WorkdirHelper, 965
create_evaluation_process
    Dakota::ProcessHandleApplicInterface, 816
create_plots_2d
    Dakota::Graphics, 514
create_tabular_datastream
    Dakota::OutputManager, 769
CreateEvaluator
    Dakota::JEGAOptimizer::EvaluatorCreator, 492
cv_index_map
    Dakota::NestedModel, 627

DAUIVLbl
    Dakota, 319
DAURVLbl
    Dakota, 320
DAUSVLbl
    Dakota, 320
DDACEDesignCompExp, 456
    Dakota::DDACEDesignCompExp, 458
DEUIVLbl
    Dakota, 320
DEURVLbl
    Dakota, 320
DEUSVLbl
    Dakota, 320
DF
    Dakota::CONMINOptimizer, 389
    DOTOptimizer, 467
DakFuncs0
Dakota, 156
Dakota, 71
  abort_handler_t, 151
  abort_mode, 156
  asstring, 153
  CAUVLbl, 319
  CEUVLbl, 320
  concatenate_restart, 155
  contains, 155
  DAUVLbl, 319
  DAURVLbl, 320
  DAUSVLbl, 319
  DEUVLbl, 320
  DEURVLbl, 320
  DEUSVLbl, 320
  DakFuncs0, 156
  DesignAndStateLabelsCheck, 321
  DiscSetLbl, 320
  FIELD_NAMES, 156
  flush, 150
  get_pathext, 155
  getRmax, 152
  getdist, 152
  id_vars_exact_compare, 154
  kw_1, 156, 242
  kw_10, 158, 244
  kw_100, 179, 260
  kw_101, 179, 260
  kw_102, 179, 261
  kw_103, 179, 261
  kw_104, 180, 261
  kw_105, 180, 261
  kw_106, 180, 261
  kw_107, 180, 262
  kw_108, 181, 262
  kw_109, 181, 262
  kw_11, 158, 244
  kw_110, 181, 262
  kw_111, 181, 262
  kw_112, 181, 263
  kw_113, 182, 263
  kw_114, 182, 263
  kw_115, 182, 263
  kw_116, 182, 263
  kw_117, 182, 264
  kw_118, 183, 264
  kw_119, 183, 264
  kw_12, 158, 244
  kw_120, 183, 264
  kw_121, 183, 264
  kw_122, 183, 265
  kw_123, 183, 265
  kw_124, 184, 265
  kw_125, 184, 265
  kw_126, 184, 265
  kw_127, 184, 265
  kw_128, 185, 266
  kw_129, 185, 266
  kw_13, 159, 244
  kw_130, 185, 266
  kw_131, 185, 266
  kw_132, 185, 266
  kw_133, 185, 266
  kw_134, 186, 267
  kw_135, 186, 267
  kw_136, 186, 267
  kw_137, 186, 267
  kw_138, 186, 267
  kw_139, 187, 268
  kw_14, 159, 244
  kw_140, 187, 268
  kw_141, 187, 268
  kw_142, 187, 268
  kw_143, 187, 268
  kw_144, 187, 269
  kw_145, 188, 269
  kw_146, 188, 269
  kw_147, 188, 269
  kw_148, 188, 269
  kw_149, 188, 269
  kw_15, 159, 245
  kw_150, 189, 270
  kw_151, 189, 270
  kw_152, 189, 270
  kw_153, 189, 270
  kw_154, 189, 270
  kw_155, 190, 270
  kw_156, 190, 271
  kw_157, 190, 271
  kw_158, 190, 271
  kw_159, 190, 271
  kw_16, 159, 245
  kw_160, 191, 271
  kw_161, 191, 272
  kw_162, 191, 272
  kw_163, 191, 272
  kw_164, 191, 272
kw_341, 232, 306
kw_342, 232, 306
kw_343, 232, 306
kw_344, 233, 306
kw_345, 233, 306
kw_346, 233, 306
kw_347, 233, 307
kw_348, 233, 307
kw_349, 233, 307
kw_350, 234, 307
kw_351, 234, 308
kw_352, 234, 308
kw_353, 235, 308
kw_354, 235, 308
kw_355, 235, 309
kw_356, 235, 309
kw_357, 236, 309
kw_358, 236, 309
kw_359, 236, 310
kw_36, 163, 248
kw_360, 236, 310
kw_361, 237, 310
kw_362, 237, 310
kw_363, 237, 311
kw_364, 237, 311
kw_365, 238, 311
kw_366, 238, 311
kw_367, 238, 311
kw_368, 238, 312
kw_369, 239, 312
kw_37, 164, 249
kw_370, 239, 312
kw_371, 239, 312
kw_372, 239, 313
kw_373, 239, 313
kw_374, 240, 313
kw_375, 240, 313
kw_376, 240, 314
kw_377, 241, 314
kw_378, 241, 314
kw_379, 241, 314
kw_38, 164, 249
kw_380, 241, 314
kw_381, 315
kw_382, 242, 315
kw_383, 315
kw_384, 315
kw_385, 316
kw_386, 316
kw_387, 316
kw_388, 316
kw_389, 316
kw_39, 164, 249
kw_390, 317
kw_391, 317
kw_392, 317
kw_393, 317
kw_394, 318
kw_395, 318
kw_396, 318
kw_397, 318
kw_399, 319
kw_4, 157, 243
kw_40, 164, 249
kw_41, 165, 249
kw_42, 165, 249
kw_43, 165, 250
kw_44, 166, 250
kw_45, 166, 250
kw_46, 166, 250
kw_47, 166, 251
kw_48, 166, 251
kw_49, 167, 251
kw_5, 157, 243
kw_50, 167, 251
kw_51, 167, 251
kw_52, 167, 252
kw_53, 167, 252
kw_54, 168, 252
kw_55, 168, 252
kw_56, 168, 252
kw_57, 168, 252
kw_58, 168, 253
kw_59, 169, 253
kw_6, 157, 243
kw_60, 169, 253
kw_61, 169, 253
kw_62, 169, 253
kw_63, 169, 253
kw_64, 170, 254
kw_65, 170, 254
kw_66, 170, 254
kw_67, 170, 254
kw_68, 170, 254
kw_69, 171, 255
kw_7, 157, 243
kw_70, 171, 255
kw_71, 171, 255
kw_72, 171, 255
kw_73, 171, 255
kw_74, 172, 255
kw_75, 172, 256
kw_76, 172, 256
kw_77, 172, 256
kw_78, 173, 256
kw_79, 173, 256
kw_80, 173, 256
kw_81, 173, 257
kw_82, 174, 257
kw_83, 174, 257
kw_84, 174, 257
kw_85, 174, 257
kw_86, 174, 257
kw_87, 175, 258
kw_88, 175, 258
kw_89, 175, 258
kw_9, 158, 243
kw_90, 175, 258
kw_91, 176, 258
kw_92, 177, 259
kw_93, 177, 259
kw_94, 177, 259
kw_95, 177, 259
kw_96, 178, 259
kw_97, 178, 260
kw_98, 178, 260
kw_99, 179, 260
lookup_by_val, 154
mindist, 152
mindistindx, 152
mpi_debug_hold, 151
NUMBER_OF_FIELDS, 156
operator==, 151, 153
PRPMultiIndexCache, 150
PRPMultiIndexQueue, 150
perform_analysis, 153
print_restart, 154
print_restart_pdb, 154
print_restart_tabular, 155
read_neutral, 155
register_signal_handlers, 150
repair_restart, 155
SUBMETHOD_COLLABORATIVE, 150
SCI_FIELD_NAMES, 324
set_compare, 153
start_dakota_heartbeat, 153
start_grid_computing, 152
stop_grid_computing, 153
symmetric_eigenvalue_decomposition, 152
VLUncertainInt, 321
VLUncertainReal, 321
VLUncertainStr, 322
var_mp_ebound, 324
var_mp_check_cau, 322
var_mp_check_ceu, 323
var_mp_check_cv, 322
var_mp_check_dau, 323
var_mp_check_daui, 323
var_mp_check_daur, 323
var_mp_check_daus, 323
var_mp_check_deui, 323
var_mp_check_deur, 324
var_mp_check_deus, 323
var_mp_check_dset, 322
var_mp_drange, 324
write_ordered, 151, 152
Dakota::GetLongOpt
   MandatoryValue, 512
   OptionalValue, 512
   Valueless, 512
Dakota::APPSEvalMgr
   APPSEvalMgr, 365
   isReadyForWork, 365
   recv, 365
   submit, 365
Dakota::APPSOptimizer
   find_optimum, 367
   initialize_variables_and_constraints, 367
   set_apps_parameters, 367
Dakota::ActiveSet
   derivVarsVector, 329
   requestVector, 329
Dakota::Analyzer
   algorithm_space_model, 334
   evaluate_parameter_sets, 334
   initialize_run, 333
   num_samples, 333
   post_run, 333
   pre_output, 334
   print_results, 334
   print_sobol_indices, 335
   read_variables_responses, 335
   sample_to_variables, 333
   variables_to_sample, 335
   variance_based_decomp, 335
Dakota::ApplicationInterface
    asynchronous_local_evaluations, 348
    asynchronous_local_evaluations_nowait, 349
    duplication_detect, 346
    init_communicators_checks, 345
    init_default_asv, 346
    init_serial, 343
    map, 343
    master_dynamic_schedule_analyses, 345
    master_dynamic_schedule_evaluations, 346
    master_dynamic_schedule_evaluations_nowait, 348
    peer_dynamic_schedule_evaluations, 347
    peer_dynamic_schedule_evaluations_nowait, 348
    peer_static_schedule_evaluations, 347
    serve_analyses_synch, 346
    serve_evaluations, 344
    serve_evaluations_asynch, 350
    serve_evaluations_asynch_peer, 350
    serve_evaluations_synch, 349
    serve_evaluations_synch_peer, 349
    set_communicators_checks, 345
    stop_evaluation_servers, 345
    synch, 344
    synch_nowait, 344
    synchronous_local_evaluations, 348
Dakota::Approximation
    ~Approximation, 355
    Approximation, 354, 355
    build, 355
    clear_all, 357
    clear_current, 356
    finalize, 356
    get_approx, 357
    operator=, 355
    pop, 356
    rebuild, 356
    restore, 356
Dakota::ApproximationInterface
    append_approximation, 361
    build_approximation, 361
    functionSurfaces, 362
    pop_approximation, 362
    read_challenge_points, 362
    rebuild_approximation, 362
    restore_approximation, 362
    update_approximation, 360, 361
Dakota::COLINApplication
    colin_request_to_dakota_request, 371
    collect_evaluation_impl, 371
    dakota_response_to_colin_response, 371
    evaluation_available, 370
    map_domain, 371
    perform_evaluation_impl, 371
    set_problem, 370
    spawn_evaluation_impl, 370
Dakota::COLINOptimizer
    COLINOptimizer, 374
    colin_cache_lookup, 375
    constraintViolation, 375
    find_optimum, 374
    post_run, 375
    returns_multiple_points, 374
    set_rng, 375
    set_solver_parameters, 375
    solver_setup, 374
Dakota::CONMINOptimizer
    A, 389
    B, 388
    C, 388
    CT, 388
    conminInfo, 386
    constraintMappingIndices, 387
    constraintMappingMultipliers, 387
    constraintMappingOffsets, 387
    constraintValues, 387
    DF, 389
    G1, 388
    G2, 388
    IC, 389
    ISC, 389
    MS1, 389
    N1, 387
    N2, 387
    N3, 387
    N4, 388
    N5, 388
    printControl, 386
    S, 388
    SCAL, 389
Dakota::CommandLineHandler
    output_helper, 378
Dakota::CommandShell
    flush, 380
    operator<<, 380
Dakota::ConcurrentMetaIterator
    pre_run, 382
    print_results, 383
Dakota::Constraints
~Constraints, 398
build views, 399
Constraints, 397, 398
copy, 398
get constraints, 400
manage linear constraints, 399
operator=, 398
reshape, 399
shape, 399
Dakota::DDACEDesignCompExp
copy data, 459
DDACEDesignCompExp, 458
num samples, 459
post run, 458
pre run, 458
resolve samples symbols, 459
Dakota::DOTOptimizer
constraintMappingIndices, 470
constraintMappingMultipliers, 470
constraintMappingOffsets, 470
constraintValues, 470
dotFDSSinfo, 469
dotInfo, 469
dotMethod, 469
intCntlParmArray, 470
printControl, 469
realCntlParmArray, 469
Dakota::DataFitSurrModel
~DataFitSurrModel, 408
actualModel, 415
append approximation, 411, 412
build approximation, 410
build global, 413
build local multipoint, 414
derived async compute response, 409
derived compute response, 408
derived init communicators, 412
derived synchronize, 409
derived synchronize nowait, 409
evaluation id, 413
export point, 413
import points, 413
initialize export, 413
manage data recastings, 413
update actual model, 414
update approximation, 410, 411
update from actual model, 414
Dakota::DirectApplicInterface
derived map ac, 463
init communicators checks, 463
set communicators checks, 463
synchronous local analysis, 463
Dakota::DiscrepancyCorrection
compute, 466
Dakota::EffGlobalMinimizer
~EffGlobalMinimizer, 474
algorithm space model, 474
get best sample, 474
Dakota::EfficientSubspaceMethod
assess reconstruction, 477
derived init communicators, 477
map xi to x, 478
reduced space uq, 477
uncertain vars to subspace, 478
Dakota::Environment
~Environment, 483
Environment, 482, 483
exit mode, 484
get environment, 484
operator=, 484
parse, 484
Dakota::ExperimentData
load experiment, 497
parse sigma types, 497
recover model, 497
Dakota::FSUDesignCompExp
enforce input rules, 505
FSUDesignCompExp, 504
num samples, 504
post run, 504
pre run, 504
Dakota::GaussProcApproximation
GPmodel apply, 510
GaussProcApproximation, 509
trendOrder, 510
Dakota::GetLongOpt
enroll, 512
GetLongOpt, 512
OptType, 512
parse, 512
retrieve, 512
usage, 513
Dakota::Graphics
add datapoint, 514
create plots 2d, 514
new dataset, 514
Dakota::GridApplicInterface
synchronous local analysis, 516
INDEX

Dakota::HierarchSurrModel
  derived_async_compute_response, 519
  derived_compute_response, 519
  derived_synchronize, 520
  derived_synchronize_nowait, 520
  evaluation_id, 520

Dakota::Interface
  ~Interface, 527
  assign_rep, 528
  eval_tag_prefix, 528
  get_interface, 529
  Interface, 527
  operator=, 528
  rawResponseMap, 529
  response_mapping, 528

Dakota::Iterator
  ~Iterator, 537
  assign_rep, 540
  core_run, 538
  eval_tag_prefix, 540
  finalize_run, 538
  get_iterator, 540, 541
  initialize_graphics, 539
  initialize_run, 537
  Iterator, 536, 537
  maxEvalConcurrency, 542
  operator=, 537
  post_run, 538
  pre_run, 538
  print_results, 539
  probDescDB, 541
  run, 539
  uses_method, 539

Dakota::IteratorScheduler
  configure, 547
  free_iterator, 546
  init_iterator, 545, 546
  IteratorScheduler, 545
  master_dynamic_schedule_iterators, 548
  partition, 547
  run_iterator, 546
  schedule_iterators, 547
  serve_iterators, 548
  set_iterator, 546

Dakota::JEGAOptimizer
  _initPts, 554
  accepts_multiple_points, 553
  find_optimum, 553
  GetBestMOSolutions, 553
  GetBestSOSolutions, 553
  GetBestSolutions, 552
  initial_points, 554
  JEGAOptimizer, 550
  LoadAlgorithmConfig, 551
  LoadDakotaResponses, 551
  LoadProblemConfig, 551
  LoadTheConstraints, 552
  LoadTheDesignVariables, 552
  LoadTheObjectiveFunctions, 552
  LoadTheParameterDatabase, 551
  returns_multiple_points, 553
  ToDoubleMatrix, 553

Dakota::JEGAOptimizer::Driver
  DestroyAlgorithm, 472
  Driver, 471
  ExtractAllData, 471
  PerformIterations, 471

Dakota::JEGAOptimizer::Evaluator
  _model, 491
  Clone, 490
  Description, 488
  Evaluate, 489, 490
  Evaluator, 487, 488
  GetDescription, 490
  GetName, 490
  GetNumberOfLinearConstraints, 489
  GetNumberOfNonLinearConstraints, 489
  Name, 488
  RecordResponses, 489
  SeparateVariables, 489

Dakota::JEGAOptimizer::EvaluatorCreator
  CreateEvaluator, 492
  EvaluatorCreator, 491

Dakota::LeastSq
  core_run, 556
  finalize_run, 557
  get_confidence_intervals, 557
  initialize_run, 556
  LeastSq, 556
  post_run, 557
  primary_resp_weighter, 558
  print_results, 557
  weight_model, 558

Dakota::LibraryEnvironment
  filtered_interface_list, 560
  filtered_model_list, 560
  LibraryEnvironment, 559

Dakota::MatlabInterface
Dakota::MetaIterator
post_run, 564

Dakota::Minimizer
algorithm_space_model, 572
data_transform_model, 572
finalize_run, 572
gnewton_set_recast, 573
initialize_run, 571
initialize_scaling, 576
lin_coeffs_modify_n2s, 577
local_recast_retrieve, 575
Minimizer, 571
modify_n2s, 576
modify_s2n, 573
need_resp_trans_byvars, 573
objective, 574
objective_gradient, 574
objective_hessian, 575
post_run, 572
primary_resp_differencer, 575
resize_best_resp_array, 575
resize_best_vars_array, 575
response_modify_n2s, 577
response_modify_s2n, 574
scale_model, 573
secondary_resp_copier, 573
secondary_resp_scaler, 576
variables_scaler, 576

Dakota::MixedVarConstraints
MixedVarConstraints, 578

Dakota::MixedVariables
MixedVariables, 579
read_tabular, 580

Dakota::Model
~Model, 602
assign_rep, 607
continuous_probability_density, 607
derivative_concurrency, 607
derived_interface, 604
estimate_derivatives, 608
estimate_message_lengths, 607
eval_tag_prefix, 605
evaluation_cache, 605
FDstep1, 608
FDstep2, 608
fdGradStepSize, 610
fdHessByFnStepSize, 610
get_model, 608
init_communicators, 606
init_serial, 606
initialize_h, 608
interface_id, 605
local_eval_concurrency, 605
local_eval_synchronization, 605
manage_asv, 610
Model, 602
operator=, 603
probDescDB, 610
subordinate_iterator, 603
subordinate_model, 603
subordinate_models, 606
surrogate_model, 603
synchronize_derivatives, 609
truth_model, 604
update_from_subordinate_model, 604
update_quasi_hessians, 609
update_response, 609

Dakota::NCSUOptimizer
NCSUOptimizer, 618
objective_eval, 619

Dakota::NIDRProblemDescDB
check_driver, 634
derived_parse_inputs, 634
make_variable_defaults, 634

Dakota::NL2SOLLeastSq
minimize_residuals, 638

Dakota::NLSSOLLeastSq
NLSSOLLeastSq, 644

Dakota::NPSOLOptimizer
NPSOLOptimizer, 758

Dakota::NestedModel
ccv_index_map, 628
cdiv_index_map, 628
cdrv_index_map, 628
cdsv_index_map, 628
cv_index_map, 627
derived_async_compute_response, 626
derived_compute_response, 625
derived_init_communicators, 627
derived_master_overload, 626
derived_synchronize, 626
div_index_map, 627
drv_index_map, 628
dsv_index_map, 627
evaluation_id, 627
local_eval_concurrency, 626
local_eval_synchronization, 626
response_mapping, 629
subModel, 629
Dakota::NomadOptimizer
load_parameters, 647
NomadOptimizer, 647
Dakota::NomadOptimizer::Evaluator
eval_x, 486
Evaluator, 485
Dakota::NonD
core_run, 656
finalize_run, 656
initialize_final_statistics, 656
initialize_random_variable_parameters, 655
initialize_random_variable_types, 655
initialize_random_variables, 654, 655
initialize_run, 655
print_distribution_map, 657
print_distribution_mappings, 656
set_u_to_x_mapping, 657
vars_u_to_x_mapping, 656
vars_x_to_u_mapping, 657
Dakota::NonDAdaptImpSampling
initialize, 660, 661
NonDAdaptImpSampling, 660
Dakota::NonDAdaptiveSampling
~NonDAdaptiveSampling, 666
NonDAdaptiveSampling, 665
Dakota::NonDBayesCalibration
algorithm_space_model, 668
NonDBayesCalibration, 668
Dakota::NonDCalibration
NonDCalibration, 670
Dakota::NonDCubature
increment_grid_preference, 672
increment_reference, 672
NonDCubature, 671
num_samples, 672
sampling_reset, 672
Dakota::NonDDREAMBayesCalibration
likelihoodScale, 676
NonDDREAMBayesCalibration, 675
prior_density, 675
prior_sample, 675
problem_size, 675
problem_value, 675
quantify_uncertainty, 676
sample_likelihood, 675
Dakota::NonDExpansion
algorithm_space_model, 681
compute_covariance_metric, 681
compute_final_statistics_metric, 681
compute_statistics, 682
increment_specification_sequence, 681
useDerivs, 682
Dakota::NonDGPImpSampling
NonDGPImpSampling, 693
quantify_uncertainty, 693
Dakota::NonDGPMSABayesCalibration
NonDGPMSABayesCalibration, 696
NonDGPMSAInstance, 696
quantify_uncertainty, 696
Dakota::NonDGlobalInterval
algorithm_space_model, 687
Dakota::NonDIncremLHSSampling
NonDIncremLHSSampling, 698
quantify_uncertainty, 698
Dakota::NonDIntegration
anisotropic_order_to_dimension_preference, 701
check_variables, 701
dimension_preference_to_anisotropic_order, 701
NonDIntegration, 700
Dakota::NonDLHSSampling
NonDLHSSampling, 707
quantify_uncertainty, 708
Dakota::NonDLocalReliability
dg_ds_eval, 721
dp2_dbeta_factor, 722
initial_taylor_series, 719
initialize_class_data, 719
initialize_level_data, 720
initialize_mpp_search_data, 720
PMA2_constraint_eval, 719
PMA_constraint_eval, 718
PMA_objective_eval, 718
probability, 722
RIA_constraint_eval, 718
RIA_objective_eval, 718
update_level_data, 721
update_mpp_search_data, 720
Dakota::NonDPolynomialChaos
increment_grid_from_order, 730
increment_order_and_grid, 730
increment_order_from_grid, 730
increment_specification_sequence, 730
NonDPolynomialChaos, 729
Dakota::NonDQUESOBayesCalibration
logitTransform, 739
NonDQUESTOBayesCalibration, 738
prior_proposal_covariance, 739
quantify_uncertainty, 739
set_ip_options, 739
user_proposal_covariance, 739
Dakota::NonDQuadrature
initialize_grid, 734
NonDQuadrature, 733, 734
num_samples, 734
sampling_reset, 734
Dakota::NonDReliability
algorithm_space_model, 741
Dakota::NonDSampling
get_parameter_sets, 746, 747
mode_counts, 748
NonDSampling, 745, 746
num_samples, 746
sampling_reset, 746
variables_to_sample, 747
view_aleatory_uncertain_counts, 747
view_design_counts, 747
view_epistemic_uncertain_counts, 747
view_uncertain_counts, 748
Dakota::NonDSparseGrid
NonDSparseGrid, 751
num_samples, 751
sampling_reset, 751
Dakota::NonDStochCollocation
compute_covariance_metric, 753
compute_final_statistics_metric, 753
NonDStochCollocation, 753
Dakota::NonlinearCGOptimizer
brent_minimize, 756
Dakota::Optimizer
core_run, 764
finalize_run, 764
initialize_run, 764
objective_reduction, 765
post_run, 764
primary.resp.reducer, 765
print_results, 764
reduce_model, 765
Dakota::OutputManager
add_datapoint, 768
create_tabular_datastream, 769
OutputManager, 768
pop_output_tag, 768
Dakota::PSUADEDesignCompExp
enforce_input_rules, 827
num_samples, 827
PSUADEDesignCompExp, 826
post_run, 827
pre_run, 826
Dakota::PStudyDACE
core_run, 824
print_results, 824
volumetric_quality, 824
Dakota::ParallelConfiguration
mi_parallel_level, 772
mi_parallel_level_iterator, 772
Dakota::ParallelLevel
clear, 776
Dakota::ParallelLibrary
increment_parallel_configuration, 784
init_communicators, 785
init_mpi_comm, 784
ParallelLibrary, 783
push_output_tag, 784
resolve_inputs, 785
terminate_modelcenter, 784
Dakota::ParamResponsePair
evalInterfaceIds, 788
ParamResponsePair, 788
read, 788
write, 788
Dakota::ParamStudy
distribute_list_of_points, 793
load_distribute_points, 793
post_run, 793
pre_run, 792
Dakota::PecosApproximation
build, 798
finalize, 799
pop, 798
rebuild, 798
restore, 799
Dakota::ProblemDescDB
∼ProblemDescDB, 807
check_and_broadcast, 808
check_input, 808
get_db, 809
operator=, 808
parse_inputs, 808
post_process, 809
ProblemDescDB, 807
Dakota::ProcessApplicInterface
autotag_files, 812
file_cleanup, 812
prepare_process_environment, 813
reset_process_environment, 813
synchronous_local_analyses, 813
Dakota::ProcessHandleApplicInterface
asynchronous_local_analyses, 817
cHECK_wait, 816
create_command_arguments, 817
create_evaluation_process, 816
init_communicators_checks, 816
ProcessHandleApplicInterface, 815
serve_analyses_asynch, 817
set_communicators_checks, 816
synchronous_local_analysis, 816
Dakota::ProgramOptions
split_filenames, 822
Dakota::PythonInterface
derived_map_ac, 829
python_convert_int, 829
Dakota::QuesoJointPdf
QuesoJointPdf, 830
Dakota::QuesoVectorRV
QuesoVectorRV, 831
Dakota::RecastModel
derived_compute_response, 838
eval_tag_prefix, 838
initialize, 838
RecastModel, 837
update_from_sub_model, 838
Dakota::RelaxedVarConstraints
RelaxedVarConstraints, 840
Dakota::RelaxedVariables
read_tabular, 842
RelaxedVariables, 841
Dakota::Response
functionGradients, 849
Dakota::ResultsDBAny
array_insert, 852
extract_data, 852
insert, 852
Dakota::ResultsEntry
ResultsEntry, 853
Dakota::RichExtrapVerification
converge_order, 860
converge_qoi, 860
estimate_order, 860
print_results, 859
Dakota::SNLLEastSq
constraint1_evaluator_gn, 900
constraint2_evaluator_gn, 900
nlf2_evaluator_gn, 900
post_run, 899
Dakota::SNLLOptimizer
constraint0_evaluator, 906
constraint1_evaluator, 906
constraint2_evaluator, 906
nlf0_evaluator, 905
nlf1_evaluator, 905
nlf2_evaluator, 906
SNLLOptimizer, 904, 905
Dakota::SensAnalysisGlobal
compute_correlations, 863
Dakota::SeqHybridMetaIterator
extract_parameter_sets, 867
print_results, 866
run_sequential, 866
run_sequential_adaptive, 866
Dakota::SharedApproxData
~SharedApproxData, 872
buildDataOrder, 873
get_shared_data, 873
operator=, 872
SharedApproxData, 871, 872
Dakota::SharedResponseData
copy, 878
Dakota::SharedResponseDataRep
copy_rep, 880
Dakota::SharedSurfpackApproxData
SharedSurfpackApproxData, 881
Dakota::SharedVariablesData
copy, 887
Dakota::SharedVariablesDataRep
activeVarsCompsTotals, 891
allContinuousIds, 892
copy_rep, 891
inactiveVarsCompsTotals, 891
variablesCompsTotals, 891
Dakota::SingleModel
eval_tag_prefix, 895
Dakota::SurfpackApproximation
add_anchor_to_surfdata, 913
build, 912
hessian, 913
SurfpackApproximation, 912
surrogates_to_surf_data, 913
Dakota::SurrBasedLocalMinimizer
approx_subprob_constraint_eval, 921
approx_subprob_objective_eval, 921
hard_convergence_check, 919
hom_constraint_eval, 921
hom_objective_eval, 921
minimize_surrogates, 919
tr_ratio_check, 920
update_penalty, 920

Dakota::SurrBasedMinimizer
  augmented_lagrangian_merit, 925
  constraintViolation, 926
  core_run, 924
  lagrangian_merit, 925
  penalty_merit, 926
  print_results, 924
  update_augmented_lagrange_multipliers, 925
  update_filter, 925
  update_lagrange_multipliers, 924

Dakota::SurrogateModel
  approxBuilds, 930
  force_rebuild, 929
  responseMode, 930

Dakota::SysCallApplicInterface
  init_communicators_checks, 932
  set_communicators_checks, 932
  spawn_analysis_to_shell, 933
  spawn_evaluation_to_shell, 933
  spawn_input_filter_to_shell, 933
  spawn_output_filter_to_shell, 933
  synchronous_local_analysis, 932
  test_local_evaluations, 932
  wait_local_evaluations, 932

Dakota::TANA3Approximation
  build, 936
  clear_current, 936

Dakota::TaylorApproximation
  build, 937

Dakota::TestDriverInterface
  derived_map_ac, 940
  herbie, 941
  herbie1D, 940
  mc_api_run, 941
  separable_combine, 941
  shubert1D, 941
  smooth_herbie, 941
  smooth_herbie1D, 941

Dakota::TrackerHTTP
  send_data_using_get, 943
  send_data_using_post, 943

Dakota::UsageTracker
  UsageTracker, 945

Dakota::VPSApproximation
  VPSInstance, 961
  VPSmodel_apply, 961

Dakota::Variables
  build_views, 955
  discrete_string_variables_view, 955

Dakota::Verification
  core_run, 956
  print_results, 956

Dakota::WorkdirHelper
  copy_items, 965
  create_directory, 965
  file_op_items, 966
  init_preferred_env_path, 967
  init_startup_path, 967
  initialize, 964
  link, 965
  link_items, 965
  po_which, 966
  prepend_path_item, 966
  prepend_preferred_env_path, 964
  recursive_copy, 966
  set_preferred_path, 966
  splitWildcard, 965
  tokenize_env_path, 967
  which, 964

dakota_dll_api.cpp, 969
dakota_stop, 970
dakota_dll_api.h, 970
dakota_stop, 971
dakota_response_to_colin_response
  Dakota::COLINApplication, 371
dakota_stop
  dakota_dll_api.cpp, 970
dakota_dll_api.h, 971
dakota_tabular_io.hpp, 971
data_transform_model
  Dakota::Minimizer, 572
  DataEnvironment, 400
  DataEnvironmentRep, 401
  DataFitSurrModel, 403
  DataInterface, 415
  DataMethod, 416
  DataMethodRep, 417
  DataModel, 430
  DataModelRep, 431
  DataResponses, 436
  DataResponsesRep, 437
  DataVariables, 441
INDEX

DataVariablesRep, 442
derivVarsVector
  Dakota::ActiveSet, 329
derivative_concurrency
  Dakota::Model, 607
derived_async_computer_response
  Dakota::DataFitSurrModel, 409
  Dakota::HierarchSurrModel, 519
  Dakota::NestedModel, 626
derived_computer_response
  Dakota::DataFitSurrModel, 408
  Dakota::HierarchSurrModel, 519
  Dakota::NestedModel, 625
  Dakota::RecastModel, 838
derived_init_communicators
  Dakota::DataFitSurrModel, 412
  Dakota::EfficientSubspaceMethod, 477
  Dakota::NestedModel, 627
derived_interface
  Dakota::Model, 604
derived_map_ac
  Dakota::DirectApplicInterface, 463
  Dakota::MatlabInterface, 562
  Dakota::PythonInterface, 829
  Dakota::TestDriverInterface, 940
derived_master_overload
  Dakota::NestedModel, 626
derived_parse_inputs
  Dakota::NIDRProblemDescDB, 634
derived_synchronize
  Dakota::DataFitSurrModel, 409
  Dakota::HierarchSurrModel, 520
  Dakota::NestedModel, 626
derived_synchronize_nowait
  Dakota::DataFitSurrModel, 409
  Dakota::HierarchSurrModel, 520
Description
  Dakota::JEGAOptimizer::Evaluator, 488
DesignAndStateLabelsCheck
  Dakota, 321
DestroyAlgorithm
  Dakota::JEGAOptimizer::Driver, 472
dg_ds_eval
  Dakota::NonDLocalReliability, 721
dimension_preference_to_anisotropic_order
  Dakota::NonDIIntegration, 701
DirectApplicInterface, 459
DiscSetLbl
  Dakota, 320
DiscrepancyCorrection, 463
discrete_string_variables_view
  Dakota::Variables, 955
distribute_list_of_points
  Dakota::ParamStudy, 793
div_index_map
  Dakota::NestedModel, 627
dll_tester.cpp, 973
dotFDSSinfo
  Dakota::DOTOptimizer, 469
dotInfo
  Dakota::DOTOptimizer, 469
dotMethod
  Dakota::DOTOptimizer, 469
dp2_dbeta_factor
  Dakota::NonDLocalReliability, 722
Driver
  Dakota::JEGAOptimizer::Driver, 471
drv_index_map
  Dakota::NestedModel, 628
dsv_index_map
  Dakota::NestedModel, 627
duplication_detect
  Dakota::ApplicationInterface, 346
EffGlobalMinimizer, 472
EfficientSubspaceMethod, 474
EmbedHybridMetaIterator, 478
enforce_input_rules
  Dakota::FSUDesignCompExp, 505
  Dakota::PSUADesignCompExp, 827
enroll
  Dakota::GetLongOpt, 512
Environment, 480
  Dakota::Environment, 482, 483
estimate_derivatives
  Dakota::Model, 607
estimate_message_lengths
  Dakota::SingleModel, 895
estimate_order
  Dakota::RichExtrapVerification, 860
eval_tag_prefix
  Dakota::Interface, 528
  Dakota::Iterator, 540
  Dakota::Model, 605
  Dakota::RecastModel, 838
eval_x
  Dakota::NomadOptimizer::Evaluator, 486
evalInterfaceIds
Dakota::ParamResponsePair, 788
Evaluate
  Dakota::JEGAOptimizer::Evaluator, 489, 490
evaluate_parameter_sets
  Dakota::Analyzer, 334
evaluation_available
  Dakota::COLINApplication, 370
evaluation_cache
  Dakota::Model, 605
evaluation_id
  Dakota::DataFitSurrModel, 413
  Dakota::HierarchSurrModel, 520
  Dakota::NestedModel, 627
Evaluator
  Dakota::JEGAOptimizer::Evaluator, 487, 488
  Dakota::NomadOptimizer::Evaluator, 485
EvaluatorCreator
  Dakota::JEGAOptimizer::EvaluatorCreator, 491
ExecutableEnvironment, 492
exit_mode
  Dakota::Environment, 484
ExperimentData, 493
ExperimentResponse, 498
export_point
  Dakota::DataFitSurrModel, 413
extract_data
  Dakota::ResultsDBAny, 852
extract_parameter_sets
  Dakota::SeqHybridMetaIterator, 867
ExtractAllData
  Dakota::JEGAOptimizer::Driver, 471
FDstep1
  Dakota::Model, 608
FDstep2
  Dakota::Model, 608
FIELD_NAMES
  Dakota, 156
FSUDesignCompExp, 502
  Dakota::FSUDesignCompExp, 504
fdGradStepSize
  Dakota::Model, 610
fdHessByFnStepSize
  Dakota::Model, 610
fdHessByGradStepSize
  Dakota::Model, 610
file_cleanup
  Dakota::ProcessApplicInterface, 812
file_op_items
  Dakota::WorkdirHelper, 966
FileReadException, 499
filtered_interface_list
  Dakota::LibraryEnvironment, 560
filtered_model_list
  Dakota::LibraryEnvironment, 560
finalize
  Dakota::Approximation, 356
  Dakota::PecosApproximation, 799
finalize_run
  Dakota::Iterator, 538
  Dakota::LeastSq, 557
  Dakota::Minimizer, 572
  Dakota::NonD, 656
  Dakota::Optimizer, 764
find_optimum
  Dakota::APPSOptimizer, 367
  Dakota::COLINOptimizer, 374
  Dakota::JEGAOptimizer, 553
flush
  Dakota, 150
  Dakota::CommandShell, 380
force_rebuild
  Dakota::SurrogateModel, 929
ForkApplicInterface, 500
fpinit_ASIL
  library_mode.cpp, 975
  main.cpp, 979
free_iterator
  Dakota::IteratorScheduler, 546
FunctionEvalFailure, 505
functionGradients
  Dakota::Response, 849
functionSurfaces
  Dakota::ApproximationInterface, 362
G1
  Dakota::CONMINOptimizer, 388
G2
  Dakota::CONMINOptimizer, 388
GModel_apply
  Dakota::GaussProcApproximation, 510
GaussProcApproximation, 505
  Dakota::GaussProcApproximation, 509
get_approx
  Dakota::Approximation, 357
get_best_sample
  Dakota::EffGlobalMinimizer, 474
get_confidence_intervals
  Dakota::LeastSq, 557
get_constraints
INDEX

Dakota::Constraints, 400
get_db
   Dakota::ProblemDescDB, 809
get_environment
   Dakota::Environment, 484
get_interface
   Dakota::Interface, 529
get_iterator
   Dakota::Iterator, 540, 541
get_model
   Dakota::Model, 608
get_parameter_sets
   Dakota::NonDSampling, 746, 747
get_pathext
   Dakota, 155
get_shared_data
   Dakota::SharedApproxData, 873
GetBestMOSolutions
   Dakota::JEGAOptimizer, 553
GetBestSOSolutions
   Dakota::JEGAOptimizer, 553
GetBestSolutions
   Dakota::JEGAOptimizer, 552
GetDescription
   Dakota::JEGAOptimizer::Evaluator, 490
GetLongOpt, 510
   Dakota::GetLongOpt, 512
GetName
   Dakota::JEGAOptimizer::Evaluator, 490
GetNumberLinearConstraints
   Dakota::JEGAOptimizer::Evaluator, 489
GetNumberNonLinearConstraints
   Dakota::JEGAOptimizer::Evaluator, 489
gerRmax
   Dakota, 152
getdist
   Dakota, 152
gnewton_set_recast
   Dakota::Minimizer, 573
Graphics, 513
GridApplicInterface, 515

hard_convergence_check
   Dakota::SurrBasedLocalMinimizer, 919
herbie
   Dakota::TestDriverInterface, 941
herbie1D
   Dakota::TestDriverInterface, 940
hessian
   Dakota::SurfpackApproximation, 913
HierarchSurrModel, 516
hom_constraint_eval
   Dakota::SurrBasedLocalMinimizer, 921
hom_objective_eval
   Dakota::SurrBasedLocalMinimizer, 921
IC
   Dakota::CONMINOptimizer, 389
ISC
   Dakota::CONMINOptimizer, 389
id_vars_exact_compare
   Dakota, 154
import_points
   Dakota::DataFitSurrModel, 413
inactiveVarsCompsTotals
   Dakota::SharedVariablesDataRep, 891
increment_grid_from_order
   Dakota::NonDPolynomialChaos, 730
increment_grid_preference
   Dakota::NonDCubature, 672
increment_order_and_grid
   Dakota::NonDPolynomialChaos, 730
increment_order_from_grid
   Dakota::NonDPolynomialChaos, 730
increment_parallel_configuration
   Dakota::ParallelLibrary, 784
increment_reference
   Dakota::NonDCubature, 672
increment_specification_sequence
   Dakota::NonDExpansion, 681
   Dakota::NonDPolynomialChaos, 730
init_communicators
   Dakota::Model, 606
   Dakota::ParallelLibrary, 785
init_communicators_checks
   Dakota::ApplicationInterface, 345
   Dakota::DirectApplicInterface, 463
   Dakota::ProcessHandleApplicInterface, 816
   Dakota::SysCallApplicInterface, 932
init_default_asv
   Dakota::ApplicationInterface, 346
init_iterator
   Dakota::IteratorScheduler, 545, 546
init_mpi_comm
   Dakota::ParallelLibrary, 784
init_preferred_env_path
   Dakota::WorkdirHelper, 967
init_serial
   Dakota::ApplicationInterface, 343
   Dakota::Model, 606
init_startup_path
    Dakota::WorkdirHelper, 967
initial_points
    Dakota::JEGAOptimizer, 554
initial_taylor_series
    Dakota::NonDLocalReliability, 719
initialize
    Dakota::NonDAadaptImpSampling, 660, 661
    Dakota::RecastModel, 838
    Dakota::WorkdirHelper, 964
initialize_class_data
    Dakota::NonDLocalReliability, 719
initialize_export
    Dakota::DataFitSurrModel, 413
initialize_final_statistics
    Dakota::NonD, 656
initialize_graphics
    Dakota::Iterator, 539
initialize_grid
    Dakota::NonDQuadrature, 734
initialize_h
    Dakota::Model, 608
initialize_level_data
    Dakota::NonDLocalReliability, 720
initialize_mpp_search_data
    Dakota::NonDLocalReliability, 720
initialize_random_variable_parameters
    Dakota::NonD, 655
initialize_random_variable_types
    Dakota::NonD, 655
initialize_random_variables
    Dakota::NonD, 654, 655
initialize_run
    Dakota::Analyzer, 333
    Dakota::Iterator, 537
    Dakota::LeastSq, 556
    Dakota::Minimizer, 571
    Dakota::NonD, 655
    Dakota::Optimizer, 764
initialize_scaling
    Dakota::Minimizer, 576
initialize_variables_and_constraints
    Dakota::APPSOptimizer, 367
insert
    Dakota::ResultsDBAny, 852
intCntlParmArray
    Dakota::DOTOptimizer, 470
Interface, 521
    Dakota::Interface, 527
interface_id
    Dakota::Model, 605
isReadyForWork
    Dakota::APPSEvalMgr, 365
Iterator, 529
    Dakota::Iterator, 536, 537
    Dakota::IteratorScheduler, 542
    Dakota::Interface, 527
JEGAOptimizer, 548
    Dakota::JEGAOptimizer, 550
JEGAOptimizer.cpp, 973
JEGAOptimizer.hpp, 974
JEGAOptimizer::Driver, 470
JEGAOptimizer::Evaluator, 486
JEGAOptimizer::EvaluatorCreator, 491
kw_1
    Dakota, 156, 242
    Dakota, 158, 244
kw_10
    Dakota, 158, 244
kw_100
    Dakota, 179, 260
kw_101
    Dakota, 179, 260
kw_102
    Dakota, 179, 261
kw_103
    Dakota, 179, 261
kw_104
    Dakota, 180, 261
kw_105
    Dakota, 180, 261
kw_106
    Dakota, 180, 261
kw_107
    Dakota, 180, 262
kw_108
    Dakota, 181, 262
kw_109
    Dakota, 181, 262
kw_11
    Dakota, 158, 244
kw_110
    Dakota, 181, 262
kw_111
    Dakota, 181, 262
kw_112
    Dakota, 181, 263
kw_113
INDEX

Dakota, 182, 263
kw_114
Dakota, 182, 263
kw_115
Dakota, 182, 263
kw_116
Dakota, 182, 263
kw_117
Dakota, 182, 264
kw_118
Dakota, 183, 264
kw_119
Dakota, 183, 264
kw_12
Dakota, 158, 244
kw_120
Dakota, 183, 264
kw_121
Dakota, 183, 264
kw_122
Dakota, 183, 265
kw_123
Dakota, 183, 265
kw_124
Dakota, 184, 265
kw_125
Dakota, 184, 265
kw_126
Dakota, 184, 265
kw_127
Dakota, 184, 265
kw_128
Dakota, 185, 266
kw_129
Dakota, 185, 266
kw_13
Dakota, 159, 244
kw_130
Dakota, 185, 266
kw_131
Dakota, 185, 266
kw_132
Dakota, 185, 266
kw_133
Dakota, 185, 266
kw_134
Dakota, 186, 267
kw_135
Dakota, 186, 267
kw_136
Dakota, 186, 267
kw_137
Dakota, 186, 267
kw_138
Dakota, 186, 267
kw_139
Dakota, 187, 268
kw_14
Dakota, 159, 244
kw_140
Dakota, 187, 268
kw_141
Dakota, 187, 268
kw_142
Dakota, 187, 268
kw_143
Dakota, 187, 268
kw_144
Dakota, 187, 269
kw_145
Dakota, 187, 269
kw_146
Dakota, 188, 269
kw_147
Dakota, 188, 269
kw_148
Dakota, 188, 269
kw_149
Dakota, 188, 269
kw_15
Dakota, 159, 245
kw_150
Dakota, 189, 270
kw_151
Dakota, 189, 270
kw_152
Dakota, 189, 270
kw_153
Dakota, 189, 270
kw_154
Dakota, 189, 270
kw_155
Dakota, 190, 270
kw_156
Dakota, 190, 271
kw_157
Dakota, 190, 271
kw_158
Dakota, 200, 279
kw_202
Dakota, 200, 279
kw_203
Dakota, 200, 279
kw_204
Dakota, 200, 279
kw_205
Dakota, 201, 280
kw_206
Dakota, 201, 280
kw_207
Dakota, 201, 280
kw_208
Dakota, 201, 280
kw_209
Dakota, 201, 280
kw_21
Dakota, 161, 246
kw_210
Dakota, 201, 280
kw_211
Dakota, 202, 281
kw_212
Dakota, 202, 281
kw_213
Dakota, 202, 281
kw_214
Dakota, 202, 281
kw_215
Dakota, 202, 281
kw_216
Dakota, 202, 282
kw_217
Dakota, 203, 282
kw_218
Dakota, 203, 282
kw_219
Dakota, 203, 282
kw_22
Dakota, 161, 246
kw_220
Dakota, 203, 282
kw_221
Dakota, 204, 282
kw_222
Dakota, 204, 283
kw_223
Dakota, 204, 283
kw_224
Dakota, 204, 283
kw_225
Dakota, 204, 283
kw_226
Dakota, 205, 283
kw_227
Dakota, 205, 283
kw_228
Dakota, 205, 284
kw_229
Dakota, 205, 284
kw_23
Dakota, 161, 246
kw_230
Dakota, 205, 284
kw_231
Dakota, 284
kw_232
Dakota, 206, 284
kw_233
Dakota, 206, 285
kw_234
Dakota, 206, 285
kw_235
Dakota, 206, 285
kw_236
Dakota, 207, 285
kw_237
Dakota, 207, 285
kw_238
Dakota, 207, 286
kw_239
Dakota, 207, 286
kw_24
Dakota, 161, 246
kw_240
Dakota, 207, 286
kw_241
Dakota, 207, 286
kw_242
Dakota, 208, 286
kw_243
Dakota, 208, 286
kw_244
Dakota, 208, 287
kw_245
Dakota, 208, 287
kw_246
Dakota, 208, 287
kw_247
Dakota, 209, 287
kw_248
Dakota, 209, 288
kw_249
Dakota, 210, 288
kw_25
Dakota, 161, 247
kw_250
Dakota, 210, 288
kw_251
Dakota, 211, 288
kw_252
Dakota, 211, 288
kw_253
Dakota, 211, 289
kw_254
Dakota, 211, 289
kw_255
Dakota, 212, 289
kw_256
Dakota, 212, 289
kw_257
Dakota, 212, 289
kw_258
Dakota, 212, 289
kw_259
Dakota, 213, 290
kw_26
Dakota, 162, 247
kw_260
Dakota, 213, 290
kw_261
Dakota, 213, 290
kw_262
Dakota, 213, 290
kw_263
Dakota, 213, 291
kw_264
Dakota, 214, 291
kw_265
Dakota, 214, 291
kw_266
Dakota, 214, 291
kw_267
Dakota, 215, 291
kw_268
Dakota, 215, 291
kw_269
Dakota, 215, 292
kw_27
Dakota, 162, 247
kw_270
Dakota, 216, 292
kw_271
Dakota, 216, 292
kw_272
Dakota, 216, 292
kw_273
Dakota, 216, 292
kw_274
Dakota, 217, 293
kw_275
Dakota, 293
kw_276
Dakota, 217, 293
kw_277
Dakota, 217, 293
kw_278
Dakota, 217, 293
kw_279
Dakota, 217, 294
kw_28
Dakota, 162, 247
kw_280
Dakota, 218, 294
kw_281
Dakota, 218, 294
kw_282
Dakota, 218, 294
kw_283
Dakota, 218, 294
kw_284
Dakota, 218, 295
kw_285
Dakota, 218, 295
kw_286
Dakota, 219, 295
kw_287
Dakota, 219, 295
kw_288
Dakota, 219, 295
kw_289
Dakota, 219, 296
kw_29
Dakota, 162, 247
<table>
<thead>
<tr>
<th>Index Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dakota</td>
<td>219, 296</td>
</tr>
<tr>
<td>kw_291</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>220</td>
</tr>
<tr>
<td>kw_292</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>220, 296</td>
</tr>
<tr>
<td>kw_293</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>220, 296</td>
</tr>
<tr>
<td>kw_294</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>220, 296</td>
</tr>
<tr>
<td>kw_295</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>220, 297</td>
</tr>
<tr>
<td>kw_296</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>221, 297</td>
</tr>
<tr>
<td>kw_297</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>221, 297</td>
</tr>
<tr>
<td>kw_298</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>221, 297</td>
</tr>
<tr>
<td>kw_299</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>221, 297</td>
</tr>
<tr>
<td>kw_3</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>157, 242</td>
</tr>
<tr>
<td>kw_30</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>162, 247</td>
</tr>
<tr>
<td>kw_300</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>221, 298</td>
</tr>
<tr>
<td>kw_301</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>222, 298</td>
</tr>
<tr>
<td>kw_302</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>222, 298</td>
</tr>
<tr>
<td>kw_303</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>222, 298</td>
</tr>
<tr>
<td>kw_304</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>222, 298</td>
</tr>
<tr>
<td>kw_305</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>223, 298</td>
</tr>
<tr>
<td>kw_306</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>223, 299</td>
</tr>
<tr>
<td>kw_307</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>223, 299</td>
</tr>
<tr>
<td>kw_308</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>223, 299</td>
</tr>
<tr>
<td>kw_309</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>224, 299</td>
</tr>
<tr>
<td>kw_31</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>163, 248</td>
</tr>
<tr>
<td>kw_310</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>224, 299</td>
</tr>
<tr>
<td>kw_311</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>224, 299</td>
</tr>
<tr>
<td>kw_312</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>224, 300</td>
</tr>
<tr>
<td>kw_313</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>224, 300</td>
</tr>
<tr>
<td>kw_314</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>225, 300</td>
</tr>
<tr>
<td>kw_315</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>225, 300</td>
</tr>
<tr>
<td>kw_316</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>225, 300</td>
</tr>
<tr>
<td>kw_317</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>225, 300</td>
</tr>
<tr>
<td>kw_318</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>225, 301</td>
</tr>
<tr>
<td>kw_319</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>226, 301</td>
</tr>
<tr>
<td>kw_32</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>163, 248</td>
</tr>
<tr>
<td>kw_320</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>226, 301</td>
</tr>
<tr>
<td>kw_321</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>226, 301</td>
</tr>
<tr>
<td>kw_322</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>227, 302</td>
</tr>
<tr>
<td>kw_323</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>227, 302</td>
</tr>
<tr>
<td>kw_324</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>227, 302</td>
</tr>
<tr>
<td>kw_325</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>227, 302</td>
</tr>
<tr>
<td>kw_326</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>228, 302</td>
</tr>
<tr>
<td>kw_327</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>228, 302</td>
</tr>
<tr>
<td>kw_328</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>228, 303</td>
</tr>
<tr>
<td>kw_329</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>228, 303</td>
</tr>
<tr>
<td>kw_33</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>163, 248</td>
</tr>
<tr>
<td>kw_330</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>229, 303</td>
</tr>
<tr>
<td>kw_331</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>229, 303</td>
</tr>
<tr>
<td>kw_332</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>229, 303</td>
</tr>
<tr>
<td>kw_333</td>
<td></td>
</tr>
<tr>
<td>Dakota</td>
<td>229, 304</td>
</tr>
<tr>
<td>kw_334</td>
<td></td>
</tr>
</tbody>
</table>
Dakota, 230, 304
kw_335
Dakota, 230, 304
kw_336
Dakota, 230, 304
kw_337
Dakota, 230, 305
kw_338
Dakota, 231, 305
kw_339
Dakota, 231, 305
kw_34
Dakota, 163, 248
kw_340
Dakota, 231, 305
kw_341
Dakota, 232, 306
kw_342
Dakota, 232, 306
kw_343
Dakota, 232, 306
kw_344
Dakota, 233, 306
kw_345
Dakota, 233, 306
kw_346
Dakota, 233, 306
kw_347
Dakota, 233, 307
kw_348
Dakota, 233, 307
kw_349
Dakota, 234, 307
kw_35
Dakota, 163, 248
kw_350
Dakota, 234, 307
kw_351
Dakota, 234, 308
kw_352
Dakota, 234, 308
kw_353
Dakota, 235, 308
kw_354
Dakota, 235, 308
kw_355
Dakota, 235, 309
kw_356
Dakota, 235, 309
kw_357
Dakota, 236, 309
kw_358
Dakota, 236, 309
kw_359
Dakota, 236, 310
kw_36
Dakota, 163, 248
kw_360
Dakota, 236, 310
kw_361
Dakota, 237, 310
kw_362
Dakota, 237, 310
kw_363
Dakota, 237, 311
kw_364
Dakota, 237, 311
kw_365
Dakota, 238, 311
kw_366
Dakota, 238, 311
kw_367
Dakota, 238, 311
kw_368
Dakota, 238, 312
kw_369
Dakota, 239, 312
kw_37
Dakota, 164, 249
kw_370
Dakota, 239, 312
kw_371
Dakota, 239, 312
kw_372
Dakota, 239, 313
kw_373
Dakota, 239, 313
kw_374
Dakota, 239, 313
kw_375
Dakota, 240, 313
kw_376
Dakota, 240, 313
kw_377
Dakota, 240, 314
kw_378
Dakota, 241, 314
kw_379
Dakota, 241, 314
kw_38
Dakota, 164, 249
kw_380
Dakota, 241, 314
kw_381
Dakota, 315
kw_382
Dakota, 242, 315
kw_383
Dakota, 315
kw_384
Dakota, 315
kw_385
Dakota, 316
kw_386
Dakota, 316
kw_387
Dakota, 316
kw_388
Dakota, 316
kw_389
Dakota, 316
kw_39
Dakota, 164, 249
kw_390
Dakota, 317
kw_391
Dakota, 317
kw_392
Dakota, 317
kw_393
Dakota, 317
kw_394
Dakota, 318
kw_395
Dakota, 318
kw_396
Dakota, 318
kw_397
Dakota, 318
kw_399
Dakota, 319
kw_4
Dakota, 157, 243
kw_40
Dakota, 164, 249
kw_41
Dakota, 165, 249
kw_42
Dakota, 165, 249
kw_43
Dakota, 165, 250
kw_44
Dakota, 166, 250
kw_45
Dakota, 166, 250
kw_46
Dakota, 166, 250
kw_47
Dakota, 166, 251
kw_48
Dakota, 166, 251
kw_49
Dakota, 167, 251
kw_5
Dakota, 157, 243
kw_50
Dakota, 167, 251
kw_51
Dakota, 167, 251
kw_52
Dakota, 167, 252
kw_53
Dakota, 167, 252
kw_54
Dakota, 168, 252
kw_55
Dakota, 168, 252
kw_56
Dakota, 168, 252
kw_57
Dakota, 168, 252
kw_58
Dakota, 168, 253
kw_59
Dakota, 169, 253
kw_6
Dakota, 157, 243
kw_60
Dakota, 169, 253
kw_61
Dakota, 169, 253
kw_62
Dakota, 169, 253
kw_63
Dakota, 169, 253
kw_64
Dakota, 170, 254
kw.65
dakota, 170, 254
kw.66
dakota, 170, 254
kw.67
dakota, 170, 254
kw.68
dakota, 170, 254
kw.69
dakota, 171, 255
kw.7
dakota, 157, 243
kw.70
dakota, 171, 255
kw.71
dakota, 171, 255
kw.72
dakota, 171, 255
kw.73
dakota, 171, 255
kw.74
dakota, 172, 255
kw.75
dakota, 172, 256
kw.76
dakota, 172, 256
kw.77
dakota, 172, 256
kw.78
dakota, 173, 256
kw.79
dakota, 173, 256
kw.8
dakota, 158, 243
kw.80
dakota, 173, 256
kw.81
dakota, 173, 257
kw.82
dakota, 174, 257
kw.83
dakota, 174, 257
kw.84
dakota, 174, 257
kw.85
dakota, 174, 257
kw.86
dakota, 174, 257
kw.87
dakota, 175, 258
kw.88
dakota, 175, 258
kw.89
dakota, 175, 258
kw.9
dakota, 158, 243
kw.90
dakota, 175, 258
kw.91
dakota, 175, 258
kw.92
dakota, 176, 258
kw.93
dakota, 177, 259
kw.94
dakota, 177, 259
kw.95
dakota, 177, 259
kw.96
dakota, 178, 259
kw.97
dakota, 178, 260
kw.98
dakota, 178, 260
kw.99
dakota, 179, 260

lagrangian_merit
Dakota::SurrBasedMinimizer, 925
LeastSq, 554
library_mode.cpp, 974
callback_function, 976
fpinit_ASIL, 975
main, 977
parallel_input, 977
parallel_interface_plugin, 976
run_dakota.data, 975
run_dakota.mixed, 976
run_dakota.parse, 975
serial_input, 977
serial_interface_plugin, 976
library_split.cpp, 978
LibraryEnvironment, 558
Dakota::LibraryEnvironment, 559
LightWtBaseConstructor, 560
likelihoodScale
Dakota::NonDDREAMBayesCalibration, 676
Dakota::MixedVarConstraints, 578
MixedVariables, 578
Dakota::MixedVariables, 579
mode_counts
Dakota::NonDSampling, 748
Model, 580
Dakota::Model, 602
modify_n2s
Dakota::Minimizer, 576
modify_s2n
Dakota::Minimizer, 573
mpi_debug_hold
Dakota, 151

N1
Dakota::CONMINOptimizer, 387
N2
Dakota::CONMINOptimizer, 387
N3
Dakota::CONMINOptimizer, 387
N4
Dakota::CONMINOptimizer, 388
N5
Dakota::CONMINOptimizer, 388
NCSUOpitimizer, 616
Dakota::NCSUOpitimizer, 618
NIDRProblemDescDB, 630
NL2Res, 635
NL2SOLLeastSq, 635
NLPLQPOptimizer, 638
NLSSOLLeastSq, 643
Dakota::NLSSOLLeastSq, 644
NPSOLOpitizer, 756
Dakota::NPSOLOpitizer, 758
NUMBER_OF_FIELDS
Dakota, 156
Name
Dakota::JEGAOptimizer::Evaluator, 488
need_resp_trans_byvars
Dakota::Minimizer, 573
NestedModel, 619
new_dataset
Dakota::Graphics, 514
nlf0_evaluator
Dakota::SNLLOpimizer, 905
nlf1_evaluator
Dakota::SNLLOpimizer, 905
nlf2_evaluator
Dakota::SNLLOpimizer, 906
nlf2_evaluator_gn
Dakota::SNLLLeastSq, 900
NoDBBaseConstructor, 644
NomadOptimizer, 645
Dakota::NomadOptimizer, 647
NomadOptimizer::Evaluator, 484
NonD, 647
NonDAradImpSampling, 657
Dakota::NonDAradImpSampling, 660
NonDAdaptiveSampling, 661
Dakota::NonDAdaptiveSampling, 665
NonDBayesCalibration, 666
Dakota::NonDBayesCalibration, 668
NonDCalibration, 668
Dakota::NonDCalibration, 670
NonDCubature, 670
Dakota::NonDCubature, 671
NonDDREAMBayesCalibration, 672
Dakota::NonDDREAMBayesCalibration, 675
NonDExpansion, 676
NonDGPImpSampling, 691
Dakota::NonDGPImpSampling, 693
NonDGPMASBayesCalibration, 694
Dakota::NonDGPMASBayesCalibration, 696
NonDGPMASInstance
Dakota::NonDGPMASBayesCalibration, 696
NonDGlobalEvidence, 683
NonDGlobalInterval, 684
NonDGlobalReliability, 687
NonDGlobalSingleInterval, 690
NonDiicremLHSSampling, 696
Dakota::NonDiicremLHSSampling, 698
NonDiIntegration, 698
Dakota::NonDiIntegration, 700
NonDiInterval, 701
NonDLHSEvidence, 703
NonDLHSInterval, 704
NonDLHSSampling, 706
Dakota::NonDLHSSampling, 707
NonDLHSSingleInterval, 708
NonDiLocalEvidence, 709
NonDiLocalInterval, 710
NonDiLocalReliability, 713
NonDiLocalSingleInterval, 722
NonDPOFDarts, 724
NonDPolynomialChaos, 726
Dakota::NonDPolynomialChaos, 729
NonDQUESOBayesCalibration, 735
Dakota::NonDQUESOBayesCalibration, 738
NonDQuadrature, 731
INDEX

Dakota::NonDQuadrature, 733, 734
NonDReliability, 740
NonDSampling, 742
Dakota::NonDSampling, 745, 746
NonDSparseGrid, 749
Dakota::NonDSparseGrid, 751
NonDStochCollocation, 751
Dakota::NonDStochCollocation, 753
NonlinearCGOptimizer, 754
num_samples
Dakota::Analyzer, 333
Dakota::DDACEDesignCompExp, 459
Dakota::FSUDesignCompExp, 504
Dakota::NonDCubature, 672
Dakota::NonDQuadrature, 734
Dakota::NonDSampling, 746
Dakota::NonDSparseGrid, 751
Dakota::PSUADEDesignCompExp, 827

objective
Dakota::Minimizer, 574
objective_eval
Dakota::NCSUOptimizer, 619
objective_gradient
Dakota::Minimizer, 574
objective_hessian
Dakota::Minimizer, 575
objective_reduction
Dakota::Optimizer, 765
operator<<
Dakota::CommandShell, 380
operator=
Dakota::Approximation, 355
Dakota::Constraints, 398
Dakota::Environment, 484
Dakota::Interface, 528
Dakota::Iterator, 537
Dakota::Model, 603
Dakota::ProblemDescDB, 808
Dakota::SharedApproxData, 872
operator==
Dakota, 151, 153
OptDartsOptimizer, 759
OptType
Dakota::GetLongOpt, 512
Optimizer, 761
OptionalValue
Dakota::GetLongOpt, 512
output_helper
Dakota::CommandLineHandler, 378
OutputManager, 766
Dakota::OutputManager, 768
OutputWriter, 769
PMA2_constraint_eval
Dakota::NonDLocalReliability, 719
PMA_constraint_eval
Dakota::NonDLocalReliability, 718
PMA_objective_eval
Dakota::NonDLocalReliability, 718
PRPMultiIndexCache
Dakota, 150
PRPMultiIndexQueue
Dakota, 150
PSUADEDesignCompExp, 825
Dakota::PSUADEDesignCompExp, 826
PStudyDACE, 823
parallel_input
library_mode.cpp, 977
parallel_interface_plugin
library_mode.cpp, 976
ParallelConfiguration, 770
ParallelDirectApplicInterface, 772
ParallelLevel, 773
ParallelLibrary, 776
Dakota::ParallelLibrary, 783
ParamResponsePair, 785
Dakota::ParamResponsePair, 788
ParamStudy, 789
parse
Dakota::Environment, 484
Dakota::GetLongOpt, 512
parse_inputs
Dakota::ProblemDescDB, 808
parse_sigma_types
Dakota::ExperimentData, 497
partial_prp_equality, 794
partial_prp_hash, 794
partition
Dakota::IteratorScheduler, 547
PecosApproximation, 795
peer_dynamic_schedule_evaluations
Dakota::ApplicationInterface, 347
peer_dynamic_schedule_evaluations_nowait
Dakota::ApplicationInterface, 348
peer_static_schedule_evaluations
Dakota::ApplicationInterface, 347
penaltyMerit
Dakota::SurrBasedMinimizer, 926
perform_analysis
Dakota, 153
perform_evaluation_impl
   Dakota::COLINApplication, 371
PerformIterations
   Dakota::JEGAOptimizer::Driver, 471
po_which
   Dakota::WorkdirHelper, 966
pop
   Dakota::Approximation, 356
   Dakota::PecosApproximation, 798
pop_approximation
   Dakota::ApproximationInterface, 362
pop_output_tag
   Dakota::OutputManager, 768
post_process
   Dakota::ProblemDescDB, 809
post_run
   Dakota::Analyzer, 333
   Dakota::COLINOptimizer, 375
   Dakota::DAcEDesignCompExp, 458
   Dakota::FSUDesignCompExp, 504
   Dakota::Iterator, 538
   Dakota::LeastSq, 557
   Dakota::MetaIterator, 564
   Dakota::Minimizer, 572
   Dakota::Optimizer, 764
   Dakota::ParamStudy, 793
   Dakota::PSUADEDesignCompExp, 827
   Dakota::SNLLLeastSq, 899
pre_output
   Dakota::Analyzer, 334
pre_run
   Dakota::ConcurrentMetaIterator, 382
   Dakota::DAcEDesignCompExp, 458
   Dakota::FSUDesignCompExp, 504
   Dakota::Iterator, 538
   Dakota::ParamStudy, 792
   Dakota::PSUADEDesignCompExp, 826
prepare_process_environment
   Dakota::ProcessApplicInterface, 813
prepend_path_item
   Dakota::WorkdirHelper, 966
prepend_preferred_env_path
   Dakota::WorkdirHelper, 964
primary_resp_differencer
   Dakota::Minimizer, 575
primary_resp_reducer
   Dakota::Optimizer, 765
primary_resp_weighter
   Dakota::LeastSq, 558
print_distribution_map
   Dakota::NonD, 657
print_distribution_mappings
   Dakota::NonD, 656
print_restart
   Dakota, 154
print_restart_pdb
   Dakota, 154
print_restart_tabular
   Dakota, 155
print_results
   Dakota::Analyzer, 334
   Dakota::ConcurrentMetaIterator, 383
   Dakota::Iterator, 539
   Dakota::LeastSq, 557
   Dakota::Optimizer, 764
   Dakota::PSUDesignDACE, 824
   Dakota::RichExtrapVerification, 859
   Dakota::SeqHybridMetaIterator, 866
   Dakota::SurfBasedMinimizer, 924
   Dakota::Verification, 956
print_sobol_indices
   Dakota::Analyzer, 335
printControl
   Dakota::CONMINOptimizer, 386
   Dakota::DOTOptimizer, 469
prior_density
   Dakota::NonDDREAMBayesCalibration, 675
prior_proposal_covariance
   Dakota::NonDQUESTBayesCalibration, 739
prior_sample
   Dakota::NonDDREAMBayesCalibration, 675
probDescDB
   Dakota::Analyzer, 335
   Dakota::Iterator, 541
   Dakota::Model, 610
probability
   Dakota::NonDLocalReliability, 722
problem_size
   Dakota::NonDDREAMBayesCalibration, 675
problem_value
   Dakota::NonDDREAMBayesCalibration, 675
ProblemDescDB, 799
   Dakota::ProblemDescDB, 807
ProcessApplicInterface, 809
ProcessHandleApplicInterface, 813
   Dakota::ProcessHandleApplicInterface, 815
ProgramOptions, 818
push_output_tag
INDEX

Dakota::ParallelLibrary, 784
python_convert_int
Dakota::PythonInterface, 829
PythonInterface, 827
quantify_uncertainty
  Dakota::NonDDREAMBayesCalibration, 676
  Dakota::NonDGPImpSampling, 693
  Dakota::NonDGPMSABayesCalibration, 696
  Dakota::NonDIncremLHSSampling, 698
  Dakota::NonDLHSSampling, 708
  Dakota::NonDQUESOBayesCalibration, 739
QuesoJointPdf
  Dakota::QuesoJointPdf, 830
  QuesoJointPdf< V, M >, 829
QuesoVectorRV
  Dakota::QuesoVectorRV, 831
  QuesoVectorRV< V, M >, 830
RIA_constraint_eval
  Dakota::NonDLocalReliability, 718
RIA_objective_eval
  Dakota::NonDLocalReliability, 718
rawResponseMap
  Dakota::Interface, 529
read
  Dakota::ParamResponsePair, 788
  read_challange_points
    Dakota::ApproximationInterface, 362
read_neutral
  Dakota, 155
read_tabular
  Dakota::MixedVariables, 580
  Dakota::RelaxedVariables, 842
read_variables_responses
  Dakota::Analyzer, 335
realCnstParmArray
  Dakota::DOTOptimizer, 469
rebuild
  Dakota::Approximation, 356
  Dakota::PecosApproximation, 798
rebuild_approximation
  Dakota::ApproximationInterface, 362
RecastModel, 831
  Dakota::RecastModel, 837
RecordResponses
  Dakota::JEGAOptimizer::Evaluator, 489
recover_model
  Dakota::ExperimentData, 497
recursive_copy
  Dakota::WorkdirHelper, 966
recv
  Dakota::APPSEvalMgr, 365
reduce_model
  Dakota::Optimizer, 765
reduced_space_uq
  Dakota::EfficientSubspaceMethod, 477
register_signal_handlers
  Dakota, 150
RelaxedVarConstraints, 839
  Dakota::RelaxedVarConstraints, 840
RelaxedVariables, 840
  Dakota::RelaxedVariables, 841
repair_restart
  Dakota, 155
requestVector
  Dakota::ActiveSet, 329
reset_process_environment
  Dakota::ProcessApplicInterface, 813
reshape
  Dakota::Constraints, 399
resize_best_resp_array
  Dakota::Minimizer, 575
resize_best_vars_array
  Dakota::Minimizer, 575
resolve_inputs
  Dakota::ParallelLibrary, 785
resolve_samples_symbols
  Dakota::DDACEDesignCompExp, 459
Response, 842
response_mapping
  Dakota::Interface, 528
  Dakota::NestedModel, 629
response_modify_n2s
  Dakota::Minimizer, 577
response_modify_s2n
  Dakota::Minimizer, 574
responseMode
  Dakota::SurrogateModel, 930
restart_util.cpp, 979
  main, 980
RestartWriter, 849
restore
  Dakota::Approximation, 356
  Dakota::PecosApproximation, 799
restore_approximation
  Dakota::ApproximationInterface, 362
ResultsDBAny, 850
ResultsEntry
Dakota::ResultsEntry, 853
ResultsEntry< StoredType >, 852
ResultsFileError, 853
ResultsID, 854
ResultsManager, 855
ResultsNames, 857
retrieve
  Dakota::GetLongOpt, 512
returns multiple points
  Dakota::COLINOptimizer, 374
  Dakota::JEGAOptimizer, 553
RichExtrapVerification, 858
run
  Dakota::Iterator, 539
run_dakota_data
    library_mode.cpp, 975
run_dakota_mixed
    library_mode.cpp, 976
run_dakota_parse
    library_mode.cpp, 975
run_iterate
  Dakota::IteratorScheduler, 546
run_sequential
  Dakota::SeqHybridMetaIterator, 866
run_sequential_adaptive
  Dakota::SeqHybridMetaIterator, 866
S
  Dakota::CONMINOptimizer, 388
SUBMETHOD_COLLABORATIVE
  Dakota, 150
SCAL
  Dakota::CONMINOptimizer, 389
SCI_FIELD_NAMES
  Dakota, 324
SIM, 325
SIM::ParallelDirectApplicInterface
test_local_evaluations, 773
SIM::SerialDirectApplicInterface
test_local_evaluations, 868
SNLLBase, 895
SNLLLeastSq, 897
SNLOptimizer, 901
  Dakota::SNLOptimizer, 904, 905
SOLBase, 907
sample_likelihood
  Dakota::NonDDREAMBayesCalibration, 675
sample_to_variables
  Dakota::Analyzer, 333
sampling_reset
  Dakota::NonDCubature, 672
  Dakota::NonDQuadature, 734
  Dakota::NonDSampling, 746
  Dakota::NonDSparseGrid, 751
scale_model
  Dakota::Minimizer, 573
schedule_iterators
  Dakota::IteratorScheduler, 547
ScilabInterface, 860
secondary_resp_copier
  Dakota::Minimizer, 573
secondary_resp_scaler
  Dakota::Minimizer, 576
send_data_using_get
  Dakota::TrackerHTTP, 943
send_data_using_post
  Dakota::TrackerHTTP, 943
SensAnalysisGlobal, 861
separable_combine
  Dakota::TestDriverInterface, 941
SeparateVariables
  Dakota::JEGAOptimizer::Evaluator, 489
SeqHybridMetaIterator, 864
serial_input
  library_mode.cpp, 977
serial_interface_plugin
  library_mode.cpp, 976
SerialDirectApplicInterface, 867
serve_analyses_asynch
  Dakota::ProcessHandleApplicInterface, 817
serve_analyses_synch
  Dakota::ApplicationInterface, 346
serve_evaluations
  Dakota::ApplicationInterface, 344
serve_evaluations_asynch
  Dakota::ApplicationInterface, 350
serve_evaluations_asynch_peer
  Dakota::ApplicationInterface, 350
serve_evaluations_synch
  Dakota::ApplicationInterface, 349
serve_evaluations_synch_peer
  Dakota::ApplicationInterface, 349
serve_iterators
  Dakota::IteratorScheduler, 548
set_apps_parameters
  Dakota::APPSOptimizer, 367
set_communicators_checks
  Dakota::ApplicationInterface, 345
  Dakota::DirectApplicInterface, 463
Dakota::ProcessHandleApplicInterface, 816
Dakota::SysCallApplicInterface, 932
set_compare
  Dakota, 153
set_ip_options
  Dakota::NonDQUESTObayesCalibration, 739
set_iterator
  Dakota::IteratorScheduler, 546
set_preferred_path
  Dakota::WorkdirHelper, 966
set_problem
  Dakota::COLINApplication, 370
set_rng
  Dakota::COLINOptimizer, 375
set_solver_parameters
  Dakota::COLINOptimizer, 375
set_u_to_x_mapping
  Dakota::NonD, 657
shape
  Dakota::Constraints, 399
SharedApproxData, 868
  Dakota::SharedApproxData, 871, 872
SharedPecosApproxData, 873
SharedResponseData, 876
SharedResponseDataRep, 878
SharedSurfpackApproxData, 880
  Dakota::SharedSurfpackApproxData, 881
SharedVariablesData, 882
SharedVariablesDataRep, 887
shubert1D
  Dakota::TestDriverInterface, 941
SimulationResponse, 892
SingleModel, 893
smooth_herbie
  Dakota::TestDriverInterface, 941
smooth_herbie1D
  Dakota::TestDriverInterface, 941
solver_setup
  Dakota::COLINOptimizer, 374
spawn_analysis_to_shell
  Dakota::SysCallApplicInterface, 933
spawn_evaluation_impl
  Dakota::COLINApplication, 370
spawn_evaluation_to_shell
  Dakota::SysCallApplicInterface, 933
spawn_input_filter_to_shell
  Dakota::SysCallApplicInterface, 933
spawn_output_filter_to_shell
  Dakota::SysCallApplicInterface, 933
  SpawnApplicInterface, 909
split_filenames
  Dakota::ProgramOptions, 822
split_wildcard
  Dakota::WorkdirHelper, 965
start_dakota_heartbeat
  Dakota, 153
start_grid_computing
  Dakota, 152
stop_evaluation_servers
  Dakota::ApplicationInterface, 345
stop_grid_computing
  Dakota, 153
subModel
  Dakota::NestedModel, 629
submit
  Dakota::APPSEvalMgr, 365
subordinate_iterator
  Dakota::Model, 603
subordinate_model
  Dakota::Model, 603
subordinate_models
  Dakota::Model, 606
SurfpackApproximation, 910
  Dakota::SurfpackApproximation, 912
SurrBasedGlobalMinimizer, 914
SurrBasedLocalMinimizer, 915
SurrBasedMinimizer, 922
surrogate_model
  Dakota::Model, 603
SurrogateModel, 926
surrogates_to_surf_data
  Dakota::SurfpackApproximation, 913
symmetric_eigenvalue_decomposition
  Dakota, 152
synch
  Dakota::ApplicationInterface, 344
synch_nowait
  Dakota::ApplicationInterface, 344
synchronize_derivatives
  Dakota::Model, 609
synchronous_local_analyses
  Dakota::ProcessApplicInterface, 813
synchronous_local_analysis
  Dakota::DirectApplicInterface, 463
Dakota::GridApplicInterface, 516
Dakota::ProcessHandleApplicInterface, 816
Dakota::SysCallApplicInterface, 932
synchronous_local_evaluations
Dakota::ApplicationInterface, 348
SysCallApplicInterface, 930
TANA3Approximation, 934
TabularDataTruncated, 934
TaylorApproximation, 936
terminate_modelcenter
Dakota::ParallelLibrary, 784
test_local_evaluations
Dakota::SysCallApplicInterface, 932
SIM::ParallelDirectApplicInterface, 773
SIM::SerialDirectApplicInterface, 868
TestDriverInterface, 937
ToDoubleMatrix
Dakota::JEGAOptimizer, 553
tokenize_env_path
Dakota::WorkdirHelper, 967
tr_ratio_check
Dakota::SurrBasedLocalMinimizer, 920
TrackerHTTP, 942
trendOrder
Dakota::GaussProcApproximation, 510
truth_model
Dakota::Model, 604
uncertain_vars_to_subspace
Dakota::EfficientSubspaceMethod, 478
update_actual_model
Dakota::DataFitSurrModel, 414
update_approximation
Dakota::ApproximationInterface, 360, 361
Dakota::DataFitSurrModel, 410, 411
update_augmented_lagrange_multipliers
Dakota::SurrBasedMinimizer, 925
update_filter
Dakota::SurrBasedMinimizer, 925
update_from_actual_model
Dakota::DataFitSurrModel, 414
update_from_sub_model
Dakota::RecastModel, 838
update_from_subordinate_model
Dakota::Model, 604
update_lagrange_multipliers
Dakota::SurrBasedMinimizer, 924
update_level_data
Dakota::NonDLocalReliability, 721
update_mpp_search_data
Dakota::NonDLocalReliability, 720
update_penalty
Dakota::SurrBasedLocalMinimizer, 920
update_quasi_hessians
Dakota::Model, 609
update_response
Dakota::Model, 609
usage
Dakota::GetLongOpt, 513
UsageTracker, 944
Dakota::UsageTracker, 945
useDerivs
Dakota::NonDExpansion, 682
user_proposal_covariance
Dakota::NonDQUESTOBayesCalibration, 739
uses_method
Dakota::Iterator, 539
VLUncertainInt
Dakota, 321
VLUncertainReal
Dakota, 321
VLUncertainStr
Dakota, 322
VLint, 957
VLreal, 957
VLstr, 958
VPSApproximation, 958
VPSInstance
Dakota::VPSApproximation, 961
VPSmodel_apply
Dakota::VPSApproximation, 961
Valueless
Dakota::GetLongOpt, 512
Var_icheck, 945
var_mp_ebound
Dakota, 324
var_mp_check_cau
Dakota, 322
var_mp_check_ceu
Dakota, 323
var_mp_check_cv
Dakota, 322
var_mp_check_daui
Dakota, 323
var_mp_check_daur
Dakota, 323
var_mp_check_daus
Dakota, 323
var_mp_check_deui
Dakota, 323
var_mp_check_deur
Dakota, 324
var_mp_check_deus
   Dakota, 323
var_mp_check_dset
   Dakota, 322
var_mp_drange
   Dakota, 324
Var_rcheck, 945
Variables, 946
variables_scaler
   Dakota::Minimizer, 576
variables_to_sample
   Dakota::Analyzer, 335
   Dakota::NonDSampling, 747
variablesCompsTotals
   Dakota::SharedVariablesDataRep, 891
variance_based_decomp
   Dakota::Analyzer, 335
vars_u_to_x_mapping
   Dakota::NonD, 656
vars_x_to_u_mapping
   Dakota::NonD, 657
Verification, 955
view_aleatory_uncertain_counts
   Dakota::NonDSampling, 747
view_design_counts
   Dakota::NonDSampling, 747
view_epistemic_uncertain_counts
   Dakota::NonDSampling, 747
view_uncertain_counts
   Dakota::NonDSampling, 748
volumetric_quality
   Dakota::PStudyDACE, 824

wait_local_evaluations
   Dakota::SysCallApplicInterface, 932
weight_model
   Dakota::LeastSq, 558
which
   Dakota::WorkdirHelper, 964
   WorkdirHelper, 962
write
   Dakota::ParamResponsePair, 788
write_ordered
   Dakota, 151, 152