



DAKOTA 101

DAKOTA Overview

<http://www.cs.sandia.gov/dakota>

Learning goals: Understand:

- How DAKOTA interfaces with a simulation (computational model)
- DAKOTA input file structure; corresponding DAKOTA abstractions
- How to run DAKOTA and JAGUAR to perform parameter studies
- DAKOTA framework benefits

Training materials can be viewed at:

http://www.sandia.gov/~briadam/sandia_only/training



Training Redesign Goals



Based on class feedback from last three years, have

- **Less lecture, more hands-on and computer interaction time**
- **Less viewing, more creating DAKOTA input files**
- **Less theory, more solving relevant problems**

Learning goals (Overview): Understand:

- **How DAKOTA interfaces with a simulation (computational model)**
- **DAKOTA input file structure; corresponding DAKOTA abstractions**
- **How to run DAKOTA and JAGUAR to perform parameter studies**
- **DAKOTA framework benefits**

Training materials can be viewed at:

<http://dev.sandia.gov/dakota/training>

DAKOTA Team Introductions



Brian Adams
Project Lead
1411



Mike Eldred
Research Mgr.
1411



Bill Bohnhoff
Support Mgr.
1341



Jim Stewart
Business Mgr.
1411



Keith Dalbey
1411



Dave Gay
1411



Patty Hough
8964



Laura Swiler
1411

- John Eddy (6342)
- Karen Haskell (9326)
- Bill Hart (1412)
- John Sirola (1433)

DAKOTA in a Nutshell



Design and Analysis toolKit for Optimization and Terascale Applications includes a wide array of algorithm capabilities to support engineering transformation through advanced modeling and simulation.

Adds value to simulation-based analysis by answering fundamental science and engineering questions:

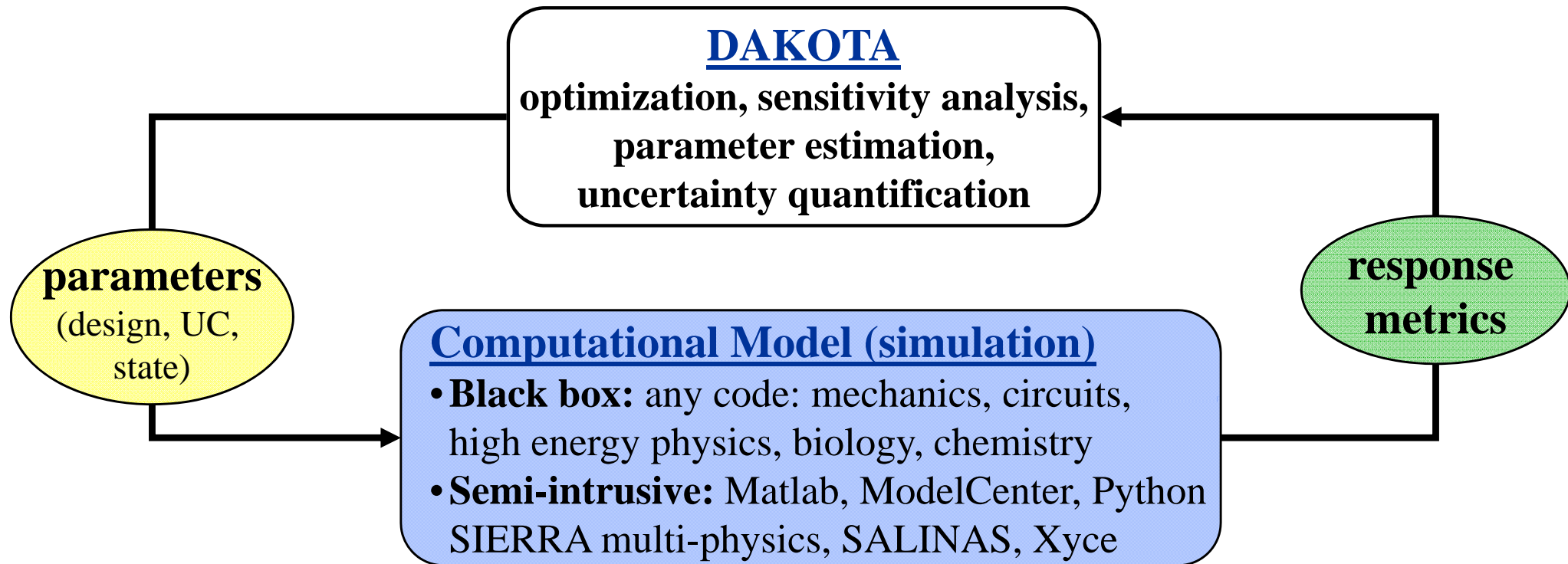
- **What are the crucial factors/parameters and how do they affect key metrics? (*sensitivity*)**
- **How safe, reliable, robust, or variable is my system? (*quantification of margins and uncertainty: QMU, UQ*)**
- **What is the best performing design or control? (*optimization*)**
- **What models and parameters best match experimental data? (*calibration*)**

- ***All rely on iterative analysis with a computational model for the phenomenon of interest***

Automated Iterative Analysis

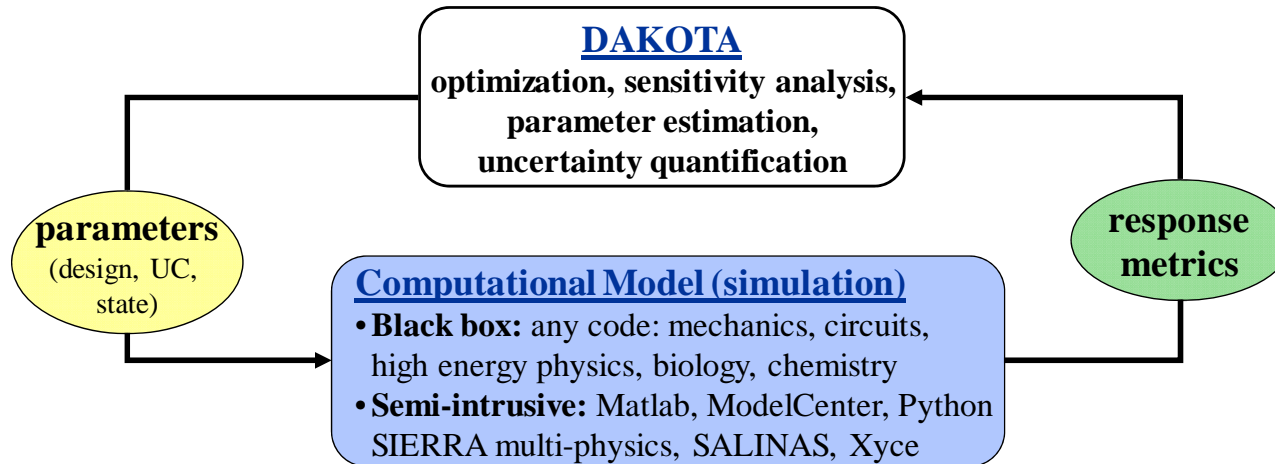


Automate typical “parameter variation” studies with advanced methods and a generic interface to your simulation



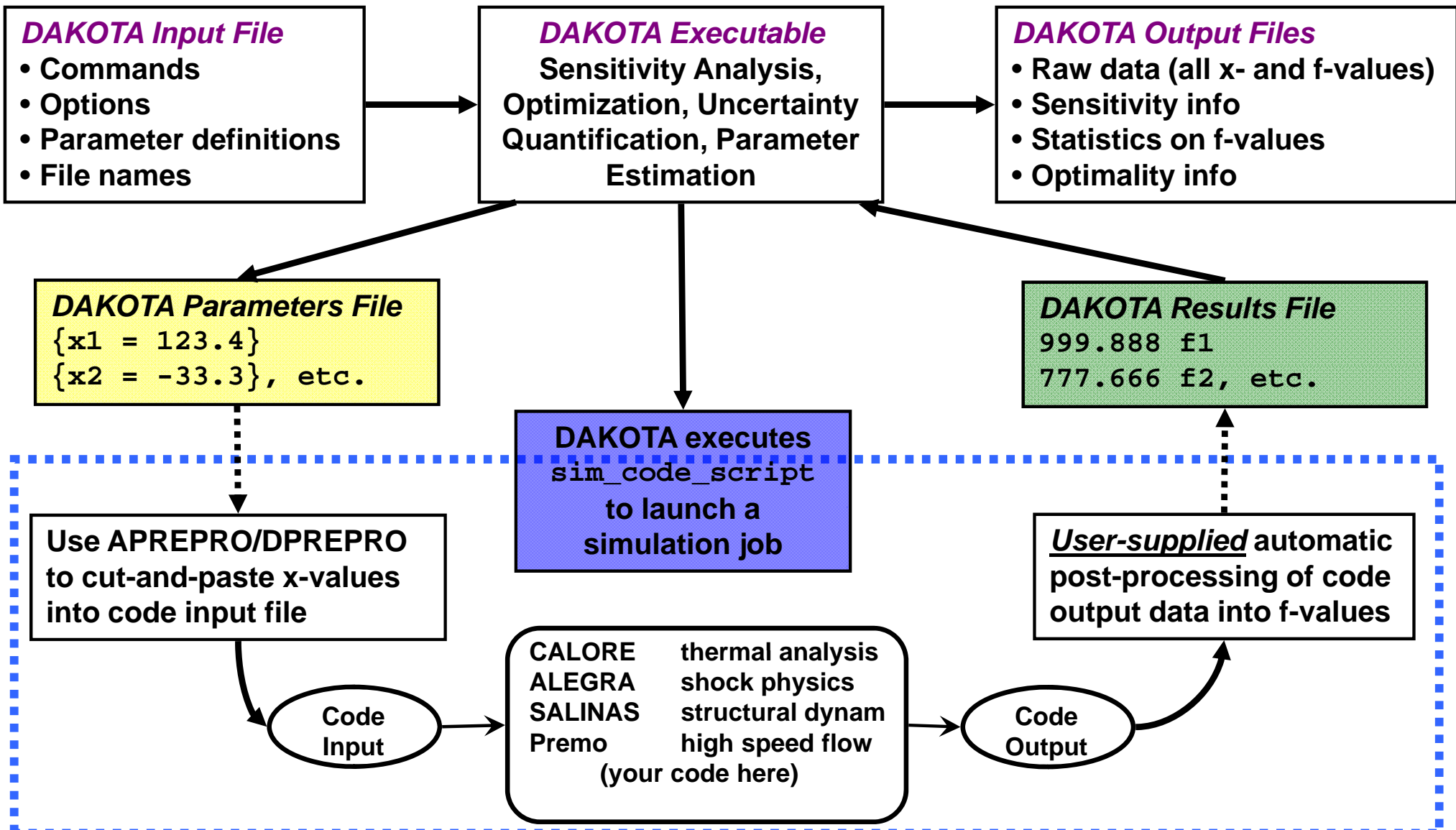
- **Can support experimental testing:** examine many accident conditions with computer models, then physically test a few worst-case conditions.

Overall DAKOTA 101 Goals



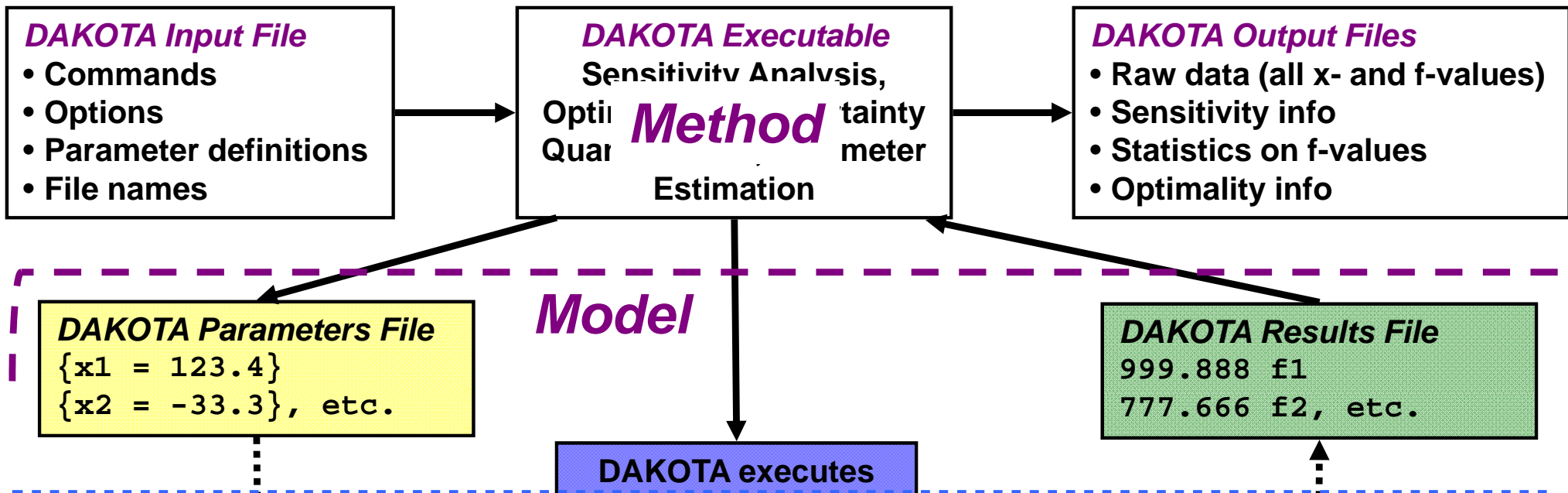
- **DAKOTA 101 will focus on setting up various kinds of studies to drive this iterative flow**
- **We'll use both (1) JAGUAR GUI and (2) text file editing, with command line run**
- **For this class we'll substitute test functions from the DAKOTA repository for the simulation**
- ***DAKOTA application interfacing focuses on connecting to your actual application***

DAKOTA Execution & Info Flow



DAKOTA Application Interfacing Class

DAKOTA Execution & Info Flow



Algebraic test function which “reads”
parameters and writes results, e.g.,
rosenbrock
text_book
osborne

Application Stand-in: Rosenbrock "Banana" Function

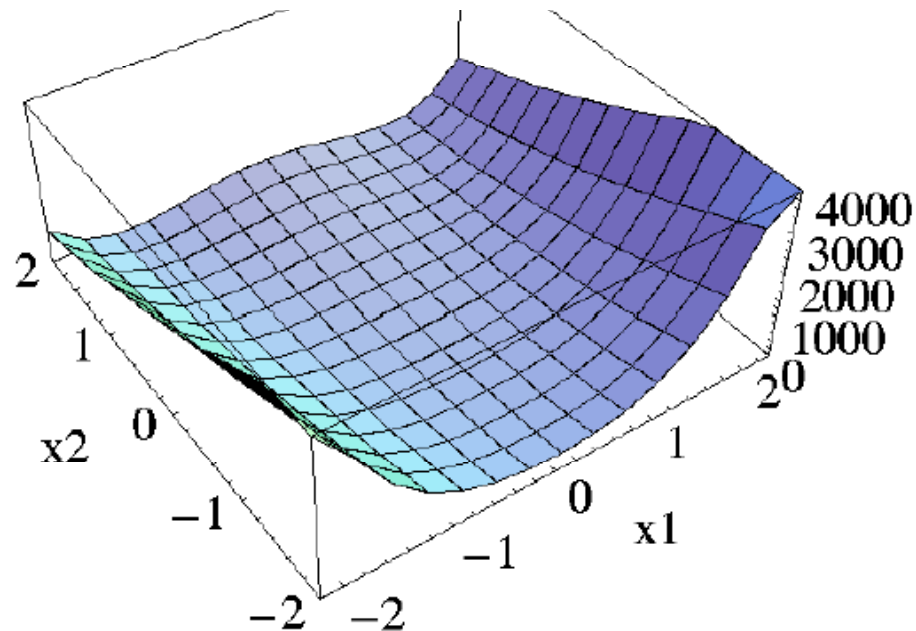
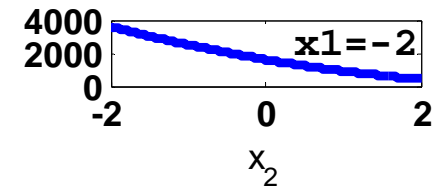
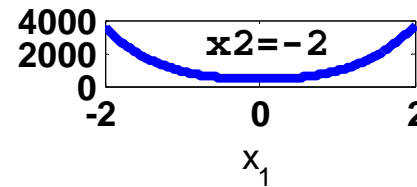
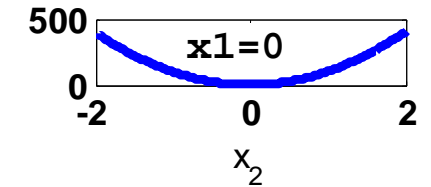
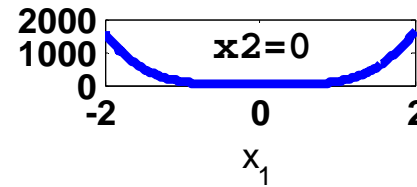
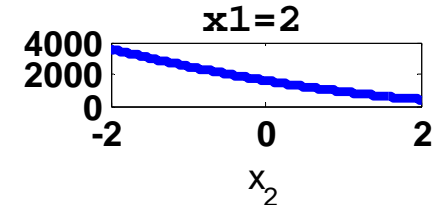
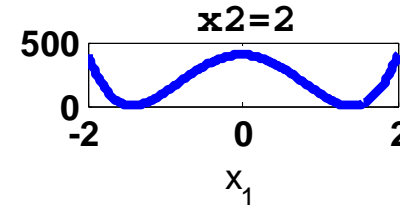
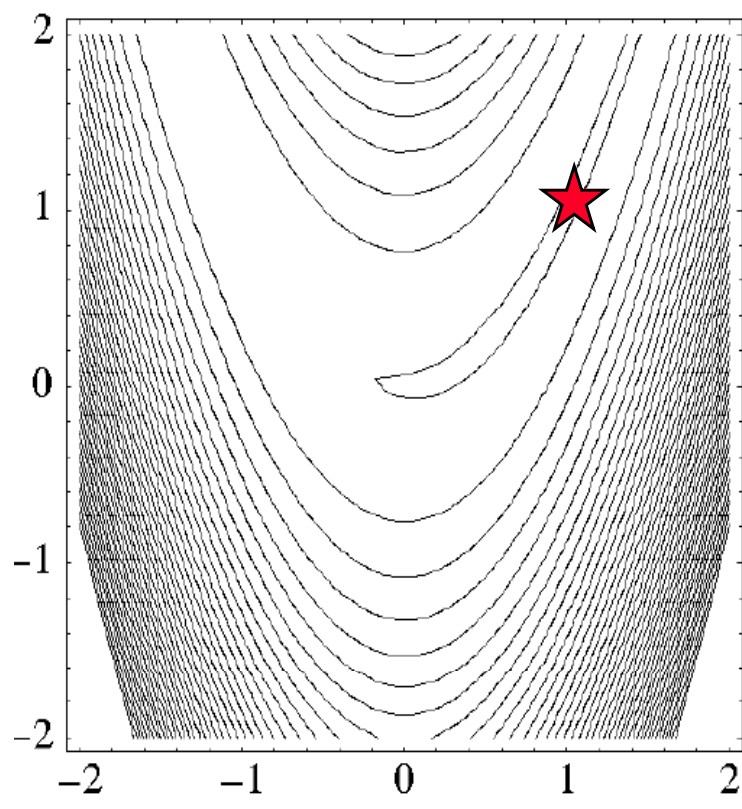


$$f(x_1, x_2) = 100 \cdot (x_2 - x_1 \cdot x_1)^2 + (1 - x_1)^2$$

$$-2 \leq x_1 \leq 2$$

$$-2 \leq x_2 \leq 2$$

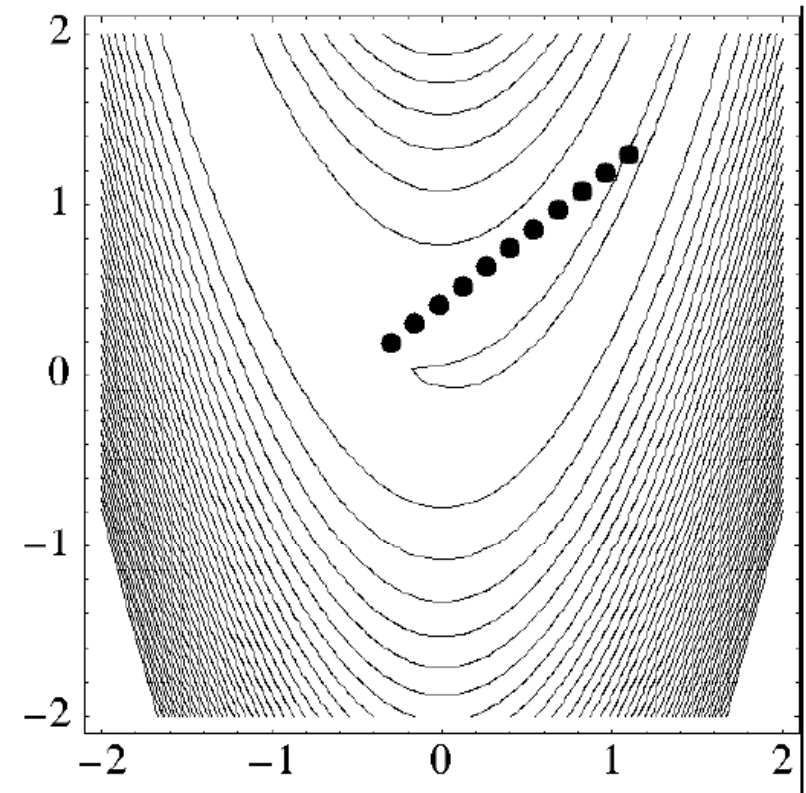
$$\text{Minimum: } f(x_1, x_2) = f(1, 1) = 0.0$$



Exploring Rosenbrock: Vector Parameter Study



- **GOAL:** assess global trend of $f(x_1, x_2)$ along a line in parameter space
- **Example:** 11 equally-spaced samples along a vector in the x_1 - x_2 parameter space (based on start point, end point, number of samples)
- **Not especially useful with $N=2$, but can be when $N>2$**
 - With large steps, provides some global trend info on function values
 - With small steps, provides some local trend info on f -values (quasi-derivatives)



See User's Manual Section 2.4.1.2

Exploring Rosenbrock: Vector PS Exercises / Discussion



- Start JAGUAR ('Jaguar' command) and set the DAKOTA executable location in Window > Preferences > Jaguar
- Create a new file from template: "Parameter Study Vector"
- Discuss the sections of the DAKOTA input file
- Use JAGUAR to run DAKOTA with this input file; examine output
- Use a text editor to open (preferably a copy of)
`Dakota/examples/tutorial/dakota_rosenbrock_vector.in`
- Run the input file from the command line and examine output
- Use either editor to modify the input (vary start or end point, number steps, etc.); discuss observations
- See DAKOTA reference manual: method commands
(<http://www.cs.sandia.gov/dakota/documentation.html>)

Input File: Vector Parameter Study



```
# DAKOTA INPUT FILE - dakota_rosenbrock_vector.in

strategy,
    single_method
    graphics,tabular_graphics_data

method,
    vector_parameter_study
    final_point = 1.1  1.3
    num_steps = 10

model,
    single

variables,
    continuous_design = 2
    initial_point    -0.3      0.2
    descriptors      'x1'      "x2"

interface,
    direct
    analysis_driver = 'rosenbrock'

responses,
    num_objective_functions = 1
    no_gradients
    no_hessians
```

JAGUAR Guides Creation of Dakota Input Deck



6 potential sections (2 optional, 4 required):

Define Problem

- Model (*optional*): single, surrogate (global, local, hierarchical), nested
- Variables (*required*): design, uncertain, and state variables; continuous/discrete
- Interface (*required*): system call, fork, or direct; specify parallel options
- Responses (*required*): number of responses/constraints, gradients, Hessian

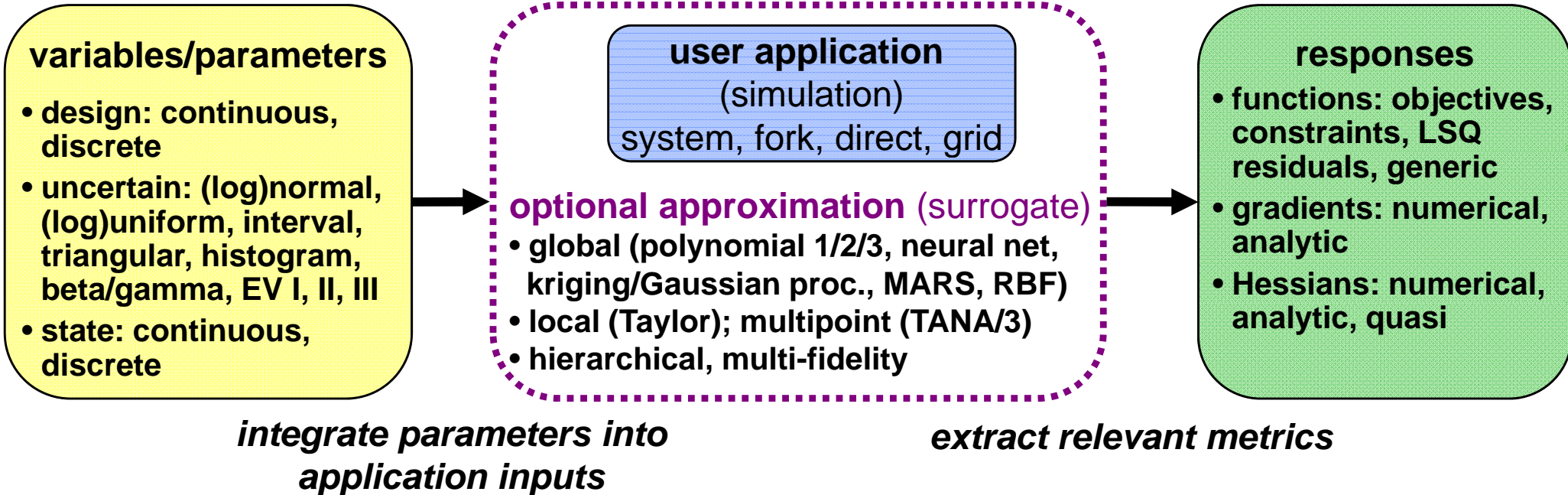
Define Flow

- Strategy (*optional*): *coordination of methods* single_method, hybrid, multi_start, pareto_set
- Method (*required*): parameter studies, nondeterministic methods, optimization methods

Optional Info: Flexibility with Models



DAKOTA models map inputs to response metrics of interest:



For all DAKOTA studies, must specify the:

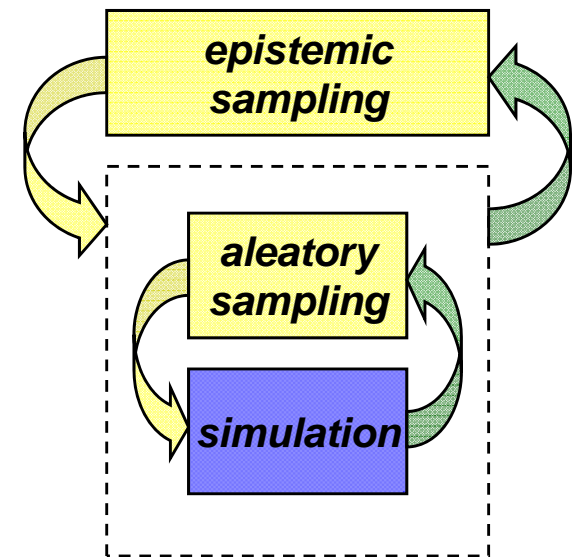
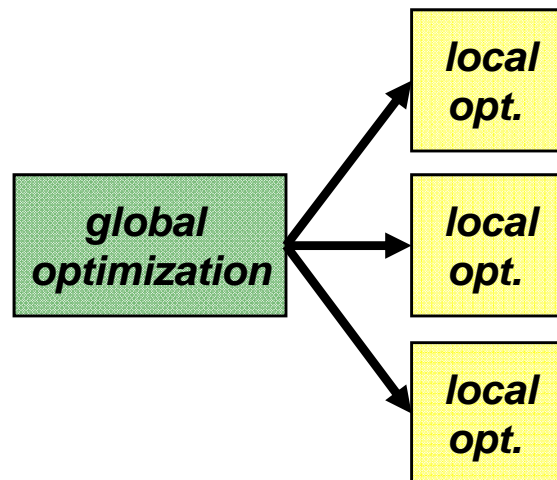
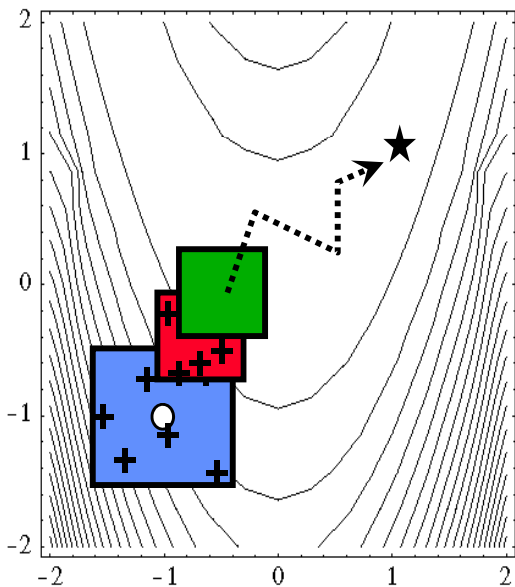
- variables of interest (types vary by study),
- interface that evaluates them
- responses it produces

Optional Info: Strategies (and advanced/multi-component methods)



Strategies (nesting, layering, sequencing and recasting facilities)
combine methods to enable advanced studies:

- Combine optimization/calibration with uncertainty quantification
- **Surrogate-based approaches**
- **Hybrid optimization**
- **Nested UQ**



Exploring Rosenbrock: Vector PS Exercises / Discussion



- Start JAGUAR and set the DAKOTA executable location in Window > Preferences > Jaguar
- Create a new file from template: “Parameter Study Vector”
- Discuss the sections of the DAKOTA input file
- Use JAGUAR to run DAKOTA with this input file; examine output
- **Use a text editor to open**
`Dakota/examples/tutorial/dakota_rosenbrock_vector.in`
- **Run the input file from the command line and examine output**
- **Use either editor to modify the input (vary start or end point, number steps, etc.); discuss observations**
- **See DAKOTA reference manual: method commands**
(<http://www.cs.sandia.gov/dakota/documentation.html>)

Dakota Execution and Output



- DAKOTA is run from a UNIX or Windows command prompt
- Capture output: input variable and response information for each function evaluation; method-specific info

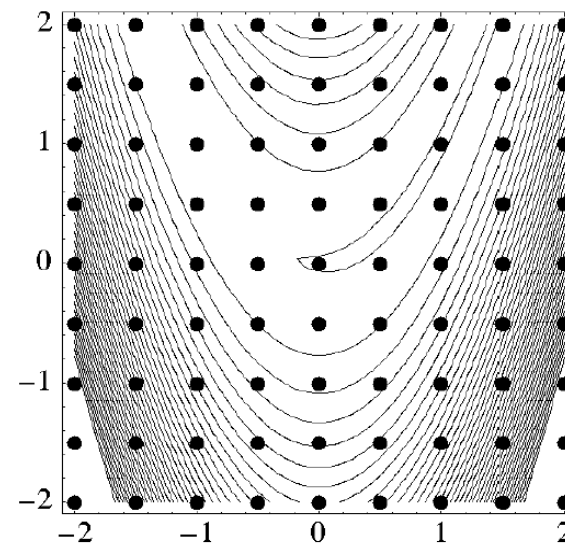
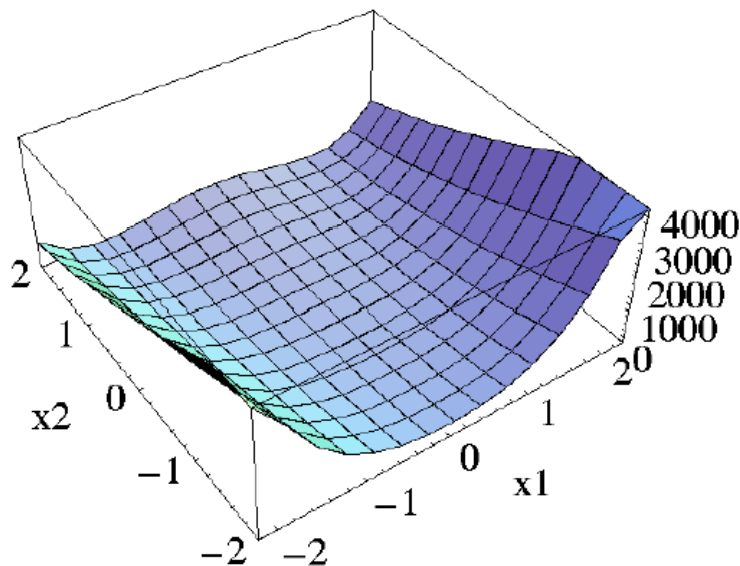
```
>> dakota -i my_run.in -o my_run.out
```
- If strategy includes `tabular_graphics_data`, generates tabular listing of inputs and outputs, called `dakota_tabular.dat`. Useful for Excel, Matlab, or other package import.
- *Other command-line options*

```
>> dakota -help
```

Exercise: Multi-dimensional Parameter Study



- **Goal:** understand how response $f(x_1, x_2)$ varies with respect to the inputs x_1 and x_2
- **Exercise:** create (preferably from scratch) and run an input file to evaluate the Rosenbrock computational model at a grid of points over $[-1.5, 2.5]$
- Try 9 points in one dimension, 6 in the other
- See DAKOTA reference manual: method commands (<http://www.cs.sandia.gov/dakota/documentation.html>)



example of uniform grid

Input File: Multi-dimensional Parameter Study



```
## DAKOTA INPUT FILE - dakota_rosenbrock_2d.in

strategy,
    single_method
    graphics,tabular_graphics_data

method,
    multidim_parameter_study
    partitions = 8 8

model,
    single

variables,
    continuous_design = 2
    lower_bounds -2.0 -2.0
    upper_bounds 2.0 2.0
    descriptors 'x1' "x2"

interface,
    direct
    analysis_driver = 'rosenbrock'

responses,
    num_objective_functions = 1
    no_gradients
    no_hessians
```



DAKOTA 101: Coming Up



Method Tour

- Sensitivity Analysis
- Uncertainty Quantification
- Optimization
- Calibration
- Advanced Topics per class interest



Additional Topics

(revisit later today per class interest)



General features

- Restart
- Evaluation cache
- Utilities in dakota_restart_util
- Tabular graphics data
- Failure capturing: abort, retry, recover, ignore
- Constraint specification: linear, nonlinear; equality, inequality
- Input/output scaling
- Matlab interface

Approximation methods

- Global data fit surrogate methods (polynomials, MARS, Kriging, etc.)
- Local surrogate methods (Taylor series, multipoint)
- Hierarchical: high/low fidelity models
- Corrections

Strategies/Advanced approaches

- Nested models: OUU
- Multi-objective (Pareto) optimization
- Multistart; multi-level hybrid
- Surrogate-based optimization (variety of constraint handling approaches): trust region; EGO/EGRA
- Reliability-based design optimization
- Advanced UQ topics: polynomial chaos, second-order probability, Dempster-Shafer, surrogate-based UQ
- AMPL: for analytic problems / algebraic mappings

Parallel capabilities: message passing, asynchronous local, hybrid

- Asynchronous evaluations
- Dakota parallel, application serial
- Dakota serial, application parallel
- Multi-level parallel: concurrent iteration, concurrent function evaluations, concurrent analyses,
- multiprocessor simulations

Getting Started with DAKOTA



- Access a Sandia installation: `module avail dakota`
AMECH (CA), CEE (ESHPC/SCICO, NM), Computer clusters (both)
or download (see Analyst Home Page or DAKOTA webpage)
- Supported on Linux, Solaris, AIX (purple), Mac OS X, Windows (no MinGW or Cygwin install required), Redstorm
- Key resource: <http://www.cs.sandia.gov/dakota>
 - Extensive documentation (user, reference, developer)
 - Support mailing lists / archives
 - Software downloads: releases and nightly stable & VOTD builds (freely available worldwide via GNU GPL)
- *User's Manual, Chapter 2*: Tutorial with example input files
- Support:
 - `dakota-users@software.sandia.gov`
(DAKOTA team and internal/external user community)
 - `dakota-developers@development.sandia.gov`
(for issues involving proprietary information)



Bonus Slides

DAKOTA Parameters File



- The file is typically named “params.in” and is generated by DAKOTA for each function evaluation
- The file lists the number of variables and the variable values in order, one value per line, followed by additional information:

```
                2 variables
-2.0000000000000000e+00 x1
-2.0000000000000000e+00 x2
                1 functions
                1 ASV_1
                2 derivative_variables
                1 DVV_1
                2 DVV_2
                0 analysis_components
```

- Your script will extract the variables from this file and insert them into your code input deck.
See Chapters 12, 13, and 14

DAKOTA Results File



- The file is typically named “results.out” and is generated by the simulation code (or script) for return to DAKOTA.
- You need to extract the relevant response(s) from your code output and write to the results.out file with the function values in order, one value per line:

f1

f2

...

fM

```
3.6090000000000000e+03 f  
[ -4.8060000000000000e+03 -1.2000000000000000e+03 ]
```

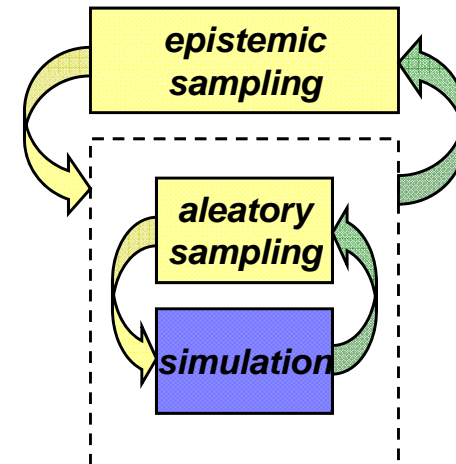
- You can add a text label after the function value (on the same line) to help you keep track of the f-values / constraints.
- If your code generates gradients of the function-values and/or Hessian values (matrix of 2nd derivatives), also report in this file.

Strategies (and advanced/multi-component methods)



Strategies (general nesting, layering, sequencing and recasting facilities) **combine methods to enable advanced studies:**

- opt within opt (multilevel opt & hierarchical MDO)
 - UQ within UQ (second-order probability)
 - UQ within opt (OUU) and NLS (MCUU)
 - opt within UQ (uncertainty of optima)
- with and without surrogate model indirection*

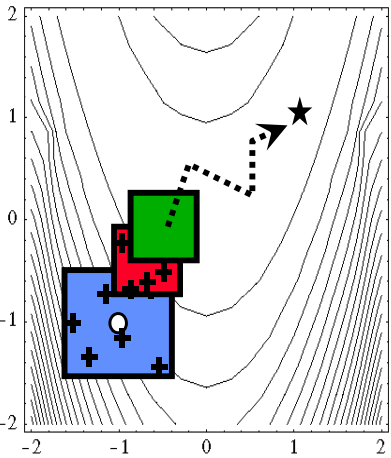


Uncertainty

- Second order probability
- Uncertainty of optima

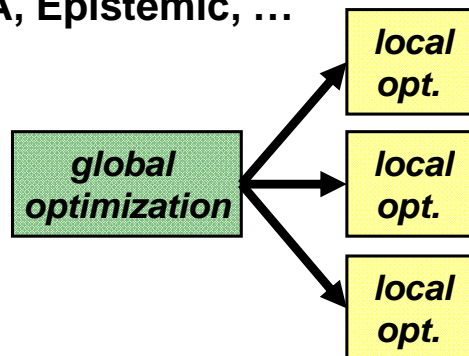
Nonlinear least squares

- Surrogate-based calibration
- Model calibration under uncertainty



Optimization

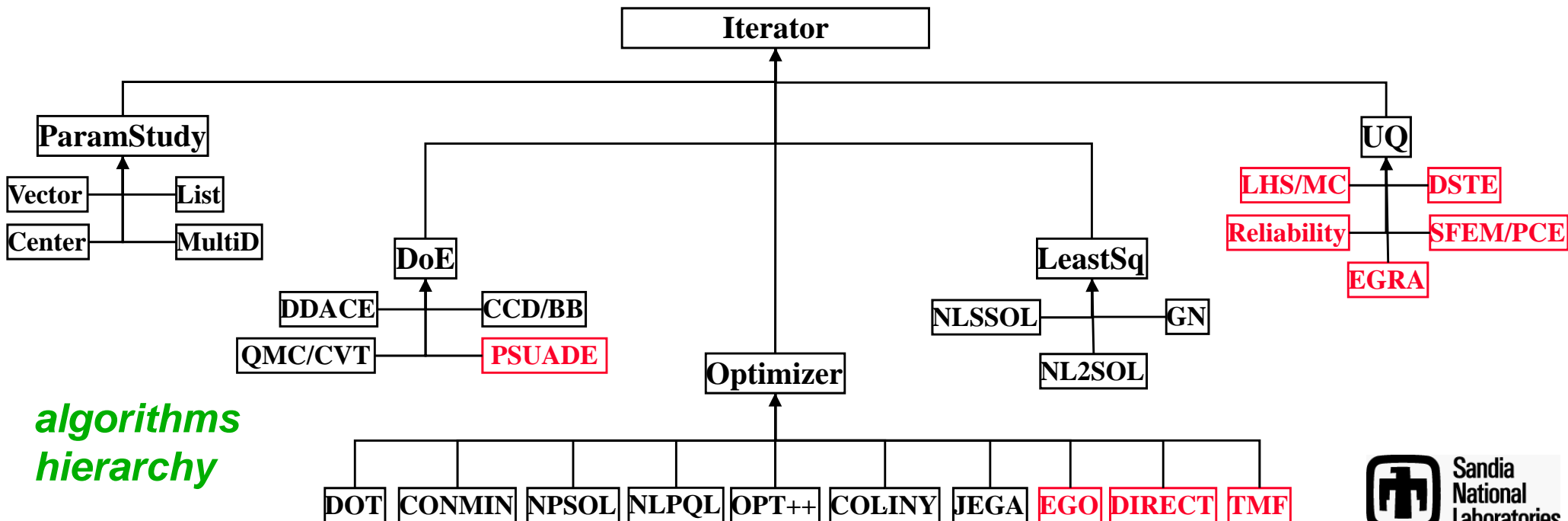
- Surrogate-based: data fit, multifidelity, ROM
- Mixed integer nonlinear programming (MINLP): PEBBL (parallel branch and bound)
- Optimization under uncertainty
 - TR-SBOUU, RBDO (Bi-level, Sequential)
 - MCUU, PC-BDO, EGO/EGRA, Epistemic, ...
- Hybrids (e.g., global/local)
- Pareto set
- Multi-start
- Multilevel methods



DAKOTA C++ Framework Goals



- **Unified software infrastructure:** reuse tools and common interfaces; *integrate commercial, open-source, and research algorithms*
- **Enable algorithm R&D**, e.g., for non-smooth/discontinuous/multimodal responses, probabilistic analysis and design, mixed variables, unreliable gradients/simulations
- **Object-oriented code;** modern software quality practices:
 - Subversion, Bugzilla, Mailman, ViewVC, gcov, cppUnit
 - Nightly platform builds with ~750 serial/parallel regression tests
- **Impact:** tool for DOE labs and external partners; broad application deployment; *free via GNU GPL (>4000 download registrations)*



Method Summary



Time-tested and advanced algorithms for deterministic and probabilistic analysis in a single toolkit to address simulations that are: nonsmooth, discontinuous, multimodal, expensive, mixed variable, failure-prone

Gradient-based Optimization

- **DOT: frcg, bfgs, mmfd, slp, sqp**
- CONMIN: frcg, mfd
- **NPSOL sqp**
- **NLPQLP sqp**
- OPT++: prcg, QN NIP, FDN NIP, FN NIP
- Dynamic plug-in: SNOPT, ...

Derivative-free Optimization

- COLINY: PS, EA, Solis-Wets, COBYLA, DIRECT
- JEGA: MOGA, SOGA
- NCSU: DIRECT, IFFCO
- OPT++: PDS
- APPSPACK, EGO, TMF

Parameter estimation (calibration)

- Nonlinear least squares: NL2SOL, **NLSSOL**, OPT++ Gauss-Newton

Sensitivity/statistical analysis

- Parameter studies: vector, list, centered, grid
- Design of experiments:
 - DDACE: LHS, MC, grid, OA, OA_LHS, CCD, BB
 - FSUDace: CVT, Halton, Hammersley
 - PSUADE: MOAT

Uncertainty quantification

- Sampling: LHS, MC, Incr. LHS, IS/AIS/MMAIS
- Local Reliability: MVFOSM/MVSOSM, x/u AMV/AMV², x/u AMV+/AMV²+, x/u TANA, FORM/SORM
- Global Reliability: EGRA
- Stochastic expansions: Wiener-Askey gen. Polynomial Chaos (Hermite, Legendre, Laguerre, Jacobi, gen. Laguerre); Stochastic collocation (Lagrange)
- Epistemic: Second-order probability, Dempster Shafer Theory of Evidence

Scalable Parallelism

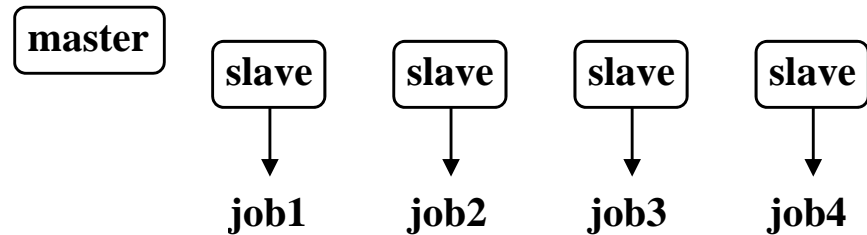


Nested parallel models support large-scale applications and architectures.

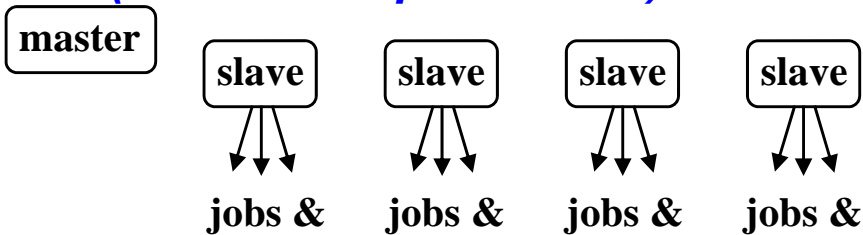
1. SMP/multiprocessor workstations: Asynchronous (external job allocation)



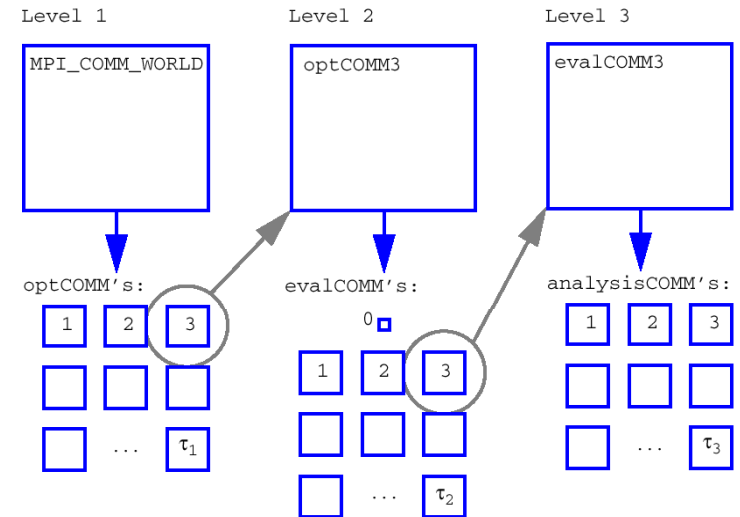
2. Cluster of workstations: Message-passing (internal job allocation)



3. Cluster of SMP's (TLCC, Thunderbird): Hybrid (service/compute model)



4. MPP (Red Storm): Internal MPI partitions (nested parallelism)



Exploiting Parallelism



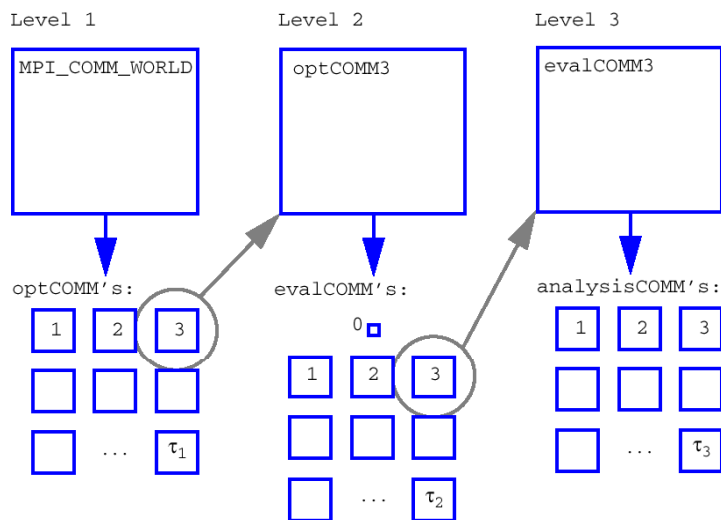
1. **Algorithmic coarse-grained parallelism:** independent fn. evaluations performed concurrently:

- ★ {
 - Gradient-based (e.g., finite difference gradients, speculative opt.)
 - Nongradient-based (e.g., GAs, PS, Monte Carlo)
 - Approximate methods (e.g., DACE)
- ★ • Concurrent-method strategies (e.g., parallel B&B, island-model GAs, OUU)

2. **Algorithmic fine-grained parallelism:** computing the internal linear algebra of an opt. algorithm in parallel (e.g., large-scale opt., SAND)

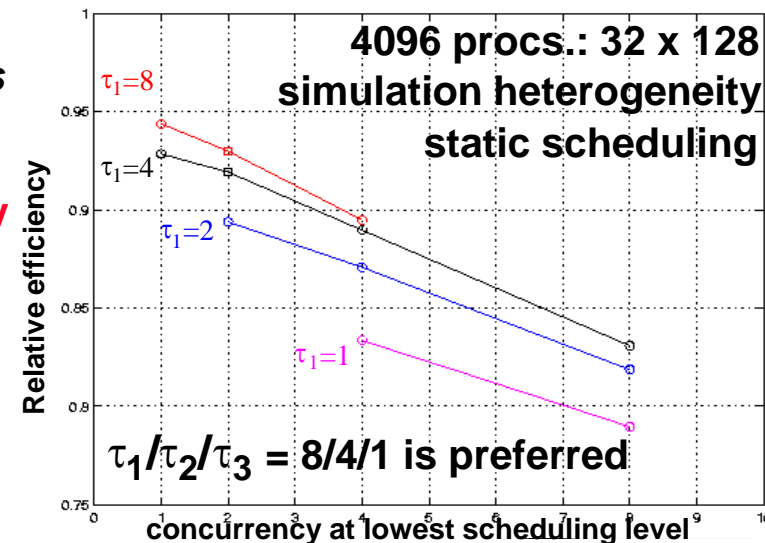
★ 3. **Function evaluation coarse-grained parallelism:** concurrent execution of separable simulations within a fn. eval. (e.g., multiple loading cases)

★ 4. **Function evaluation fine-grained parallelism:** parallelization of the solution steps within a single analysis code (e.g., SALINAS, MPSalsa)



Math analysis & experiments

- identify schemes which **maximize parallel efficiency** and are **robust w.r.t. variability**
- build schemes into automatic configuration utilities



DAKOTA Input/Output (Pre-GUI)



- **Text input/output**

Strategy

Method

Model

Variables

Interface

Responses

```

emacs: dakota_cyl_head.in
File Edit Apps Options Buffers Tools Cmds Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info News Wksho
method,
  npsol_sqp
  convergence_tolerance = 1.e-8
variables,
  continuous_design = 2
  cdv_initial_point 1.8      1.0
  cdv_upper_bounds  2.164   4.0
  cdv_lower_bounds  1.5     0.0
  cdv_descriptor    'intake_dia' 'flatness'
interface,
  application fork
  asynchronous
  analysis_driver= 'cyl_head'
responses,
  num_objective_functions = 1
  num_nonlinear_inequality_constraints = 3
  analytic_gradients
  no_hessians
strategy,
  single_method
  graphics
-----XEmacs: dakota_cyl_head.in (Fundamental PenDel)-----All-----
Wrote /var/scr12/mseldre/dakota/test/dakota_cyl_head.in
  
```

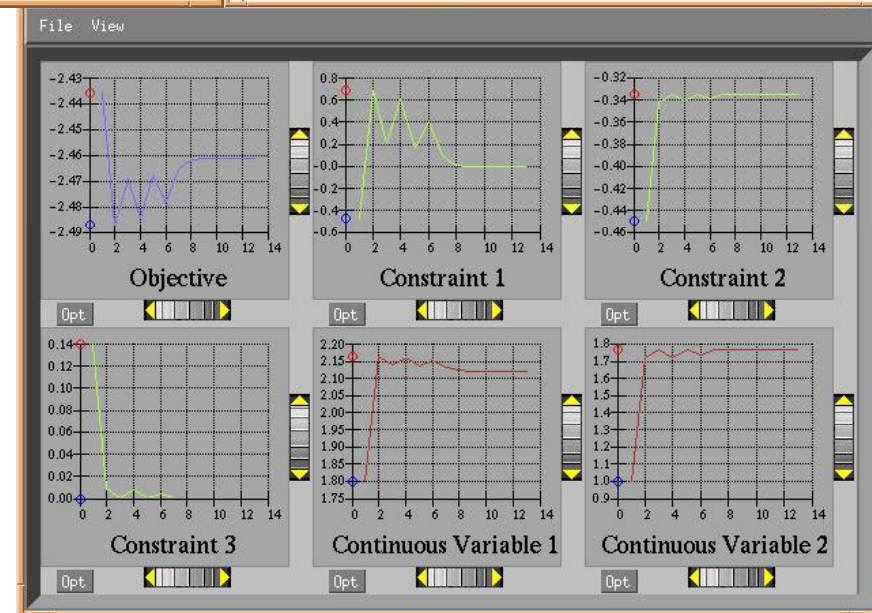
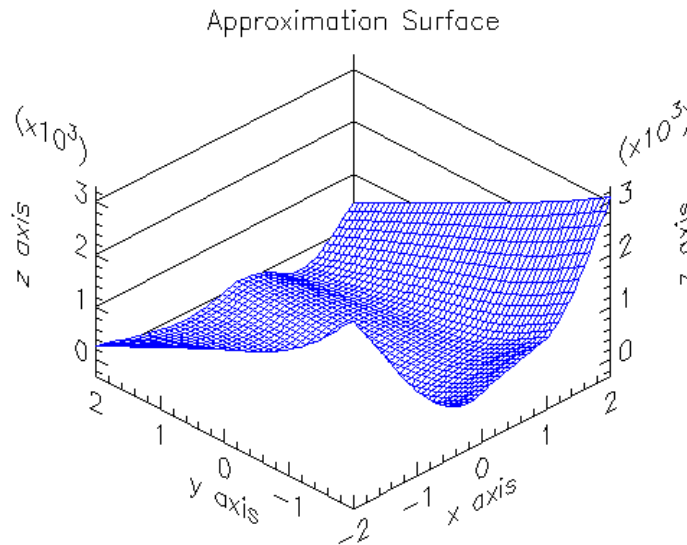
```

sacs2287: /var/scr12/mseldre/Dakota/test
Begin Function Evaluation 1
Parameters for function evaluation 1:
1.8000000000e+00 intake_dia
1.0000000000e+00 flatness
blocking fork: cyl_head /var/tmp/aaabHaa81 /var/tmp/baacHaa81
Active response data for function evaluation 1:
Active set vector = { 3 3 3 3 }
-2.435973813e+00 obj_fn
-4.7428486677e-01 nln_ineq_con1
-4.5000000000e-01 nln_ineq_con2
1.3971143170e-01 nln_ineq_con3
[ -4.3644298963e-01 1.5000000000e-01 ] obj_fn gradient
[ 1.3855136438e-01 0.0000000000e+00 ] nln_ineq_con1 gradient
[ 0.0000000000e+00 1.5000000000e-01 ] nln_ineq_con2 gradient
[ 0.0000000000e+00 -1.9485571585e-01 ] nln_ineq_con3 gradient

Maj Mnr Step Fun Merit function Violtn Norm gZ nZ Penalty Conv
0 2 0.0E+00 1 -1.98878999E+00 3.9E-01 0.0E+00 0 4.6E-01 F TF

Begin Function Evaluation 2
Parameters for function evaluation 2:
2.1640000000e+00 intake_dia
1.7169994018e+00 flatness
blocking fork: cyl head /var/tmp/caadHaa81 /var/tmp/daaeHaa81
Active response data for function evaluation 2:
Active set vector = { 3 3 3 3 }
-2.4869127193e-00 obj_fn
6.9256958800e-01 nln_ineq_con1
-3.4245008973e-01 nln_ineq_con2
8.7142207939e-03 nln_ineq_con3
[ -4.3644298963e-01 1.5000000000e-01 ] obj_fn gradient
[ 2.9814239700e+01 0.0000000000e+00 ] nln_ineq_con1 gradient
[ 0.0000000000e+00 1.5000000000e-01 ] nln_ineq_con2 gradient
[ 0.0000000000e+00 -1.6998301774e-01 ] nln_ineq_con3 gradient
  
```

- **Output graphics**



JAGUAR and DesignVue



The screenshot displays the DesignVue software interface, titled "(c) Lockheed Martin Corporation". The interface includes a menu bar (File, Tabs, View, Help) and a toolbar with various icons. On the left, a project tree shows the current project structure, including "Problem Definition", "Problem Execution", and "Problem Visualization".

The main workspace is divided into four quadrants. The top-left quadrant shows a "Brushing" panel with several sliders and a diamond icon, each associated with a parameter and its current value:

- eval_id_ = (1, 1154)
- cdv_1_ = (-2, 2)
- cdv_2_ = (-2, 2)
- obj_fn_1_ = (0, 1037.099)
- datasetIndex = (1, 1)

The top-right and bottom-right quadrants show 2D scatter plots of data points in a coordinate system with axes labeled "cdv_1_" and "cdv_2_". The bottom-left quadrant shows a 3D scatter plot of the same data points, with axes labeled "cdv_1_1", "cdv_1_2", and "cdv_1_3".

At the bottom of the interface, a status bar displays the following information: "Brushing: remaining pts: 1107 | Data Records: rows: 1154 cols: 5 | cmd: Load Data File".

DAKOTA 5.0 Highlights



- GNU Lesser General Public License (enables library use of DAKOTA)
- All new JAGUAR 2.0 graphical user interface for creating, editing, and running DAKOTA input files (BSD-like license)
- **DAKOTA modules on SNL compute clusters (module avail dakota)**
- Creation and management of evaluation working directories
- Parallelism examples; pre/post run; Mac / Windows binaries
- Additional discrete variable types; supported by parameter studies, nondeterministic sampling (discrete distributions), JEGA, and COLINY
- Stochastic expansion: Anisotropic sparse grids, numerically-generated orthogonal polynomials, and improved expansion tailoring; *many more not detailed here!*
- New epistemic and mixed aleatory-epistemic UQ: local/global interval estimation and local/global evidence.

New Capability in DAKOTA 5.0



Usability / core execution

- JAGUAR 2.0 graphical user interface, based on Eclipse Workbench, with text- and graphical-based editing, templates, sensitivity analysis wizard, and error checking. Available via separate download after registering for DAKOTA download.
- Capability to specify working directories, template directories, and lightweight linking for system and fork interfaces.
- DAKOTA input files can use Matlab-style sequence notation (L:S:U and beyond) to specify variable ranges; improved tolerance for whitespace in input files
- Improved DAKOTA + application parallelism examples; new asynchronous local evaluation static scheduling for improved parallel tiling on clusters.
- Pre-run (with optional variables output) and post-run (with variables/responses input) modes supported by sampling, parameter study, and DACE methods
- Examples of Matlab and Python interfaces
- Automated cleanup of DAKOTA temporary files on unexpected exit

Variables

- Refactor of **Variables** and **Constraints** hierarchies to manage continuous, discrete integer, and discrete real domains among design, aleatory uncertain, epistemic uncertain, and state types.
 - Additional discrete design variable types: discrete design integer range, discrete design integer set, discrete design real set
 - New discrete uncertain distributions: poisson, binomial, negative binomial, geometric, hypergeometric
 - Additional discrete state variable types: discrete state integer range, discrete state integer set, discrete state real set
- Updates to specification of continuous and discrete histograms
- Alternate specification for lognormal using lambdas and zetas

New Capability in DAKOTA 5.0



Methods (general)

- New capability to perform parameter studies and sampling over discrete variables. New discrete variable types also supported in JEGA and COLINY.
- DACE and parameter study classes (DDACE, FSUDACE, and multi-dimensional parameter study) can have correlations calculated and printed in addition to sampling methods
- Improved accuracy and robustness in correlation computations
- PSUADE: use updated (Campolongo 2007) sample generation scheme to improve space-filling properties
- Improved EGO convergence controls (based on nearest neighbor)
- Beta capabilities:
 - wrapper class for LANL's GPMSA code (Bayesian calibration)
 - importance sampling capability
 - nonlinear conjugate gradient optimization solver using Trilinos vector-matrix utilities
- Bug fixes:
 - hang in DDACE orthogonal array LHS
 - OPT++ NIP methods not respecting bound constraints
- New example problems for multifidelity OUU (MVFOSM as low fidelity UQ and stochastic expansion as high fidelity UQ), epistemic UQ, mixed aleatory-epistemic UQ, discrete sampling and parameter studies, etc.

New UQ in DAKOTA 5.0



- Stochastic expansions (polynomial chaos expansion (PCE) and stochastic collocation (SC)):
 - Simplified controls for PCE: expansion formulation now inferred or estimated from `quadrature_order` or `sparse_grid_level` specifications. Automatic expansion tailoring minimizes performance loss due to poor expansion/integration synchronization.
 - Addition of numerically-generated orthogonal polynomials for generating optimal basis for non-Askey distribution types (uses Gauss-Wigert or discretized Stieltjes procedures for polynomial recursion coefficients in combination with Golub-Welsch for Gauss point/weight computation)
 - Addition of analytic variance-based decomposition for global sensitivity analysis using method of Sobol indices
 - Addition of analytic covariance among multiple response functions.
 - Addition of anisotropic Smolyak sparse grids, with user-supplied dimension preference.
 - Revision of Gaussian quadrature rules within sparse grids to use linear growth, providing finer grain control and more uniform integrand coverage.
 - PCE customizations for sparse grids: exclusive usage of linear growth rules (no Clenshaw-Curtis or Gauss-Patterson), total-order expansions for isotropic, custom expansion for anisotropic.
 - Addition of Askey and Wiener basis polynomial over-rides.
 - Modified variable correlation logic to revert to Wiener expansions for de-correlation as needed on a per-variable basis.
 - `all_variables` mode now includes epistemic uncertain variables (previously design and state only) using a Legendre basis.
- Epistemic UQ methods:
 - Refactor of **NonDInterval** hierarchy to generalize and incorporate new methods.
 - Evidence: new global (LHS or EGO) and local (SQP or NIP) methods to calculate belief and plausibility in evidence theory calculations. Removed dependence on legacy Fortran package.
 - Interval estimation: new global (LHS or EGO) and local (SQP or NIP) methods to calculate response output intervals.
 - Extension of active Variables/Constraints views to support aleatory, epistemic, or aggregated aleatory-epistemic uncertain views.
- Mixed aleatory-epistemic UQ methods:
 - New nested approaches (e.g. second-order probability, mixed evidence) where the outer loop (e.g. interval estimation or evidence) can leverage analytic moments and their sensitivities with respect to epistemic parameters.

New Capability in DAKOTA 5.0



Framework Enhancements

- Eliminated dependence on GSL in favor of Boost (to eliminate GNU GPL dependency)
- Fully deployed Teuchos for all numerical data types. Boost and STL are used for all bookkeeping types.
- Better out-of-source (VPATH) builds and documentation, including management of Boost and Teuchos
- Switched from RNUM2 to Boost MT19937 random number generator for longer period, with run-time selection option
- Deployed [Boost multi-index](#) containers to evaluation queue management, supporting multiple indexing options for optimized lookups.
- Lightweight active/inactive data views implemented with Teuchos and [Boost multi-array](#) to eliminate deep data copies.
- Updates to third-party libraries: JEGA, PSUADE, AMPL, Boost, Pecos, Teuchos
- Minimal data mode for variables and responses omits labels, types, ids in some contexts for better scalability
- Unified bounds checking for debugging and reduced overhead in optimized builds

Miscellaneous

- Binary distributions for Mac OS X and Cygwin, including library dependencies
- Better detection of MPICH2 and shmexec communicators
- Automatic synchronization of DAKOTA input specification help docs to JAGUAR
- Surfpack: improved random seed management for repeatability.
- Better handling of numerical comparisons on 32-bit
- Finite differencing honors variable bounds; more efficient finite difference Hessians
- PSUADE generally available under LGPL
- Surfpack and LHS relicensed under LGPL
- DLL API improvements for re-entry, error handling, get/set options
- ModelCenter, MATLAB, and Python interface updates to latest API
- Updates to Matlab memory management
- HTTP-based usage tracking (disabled by default; currently used Sandia internally only)
- Windows: use spawn as alternative to fork