

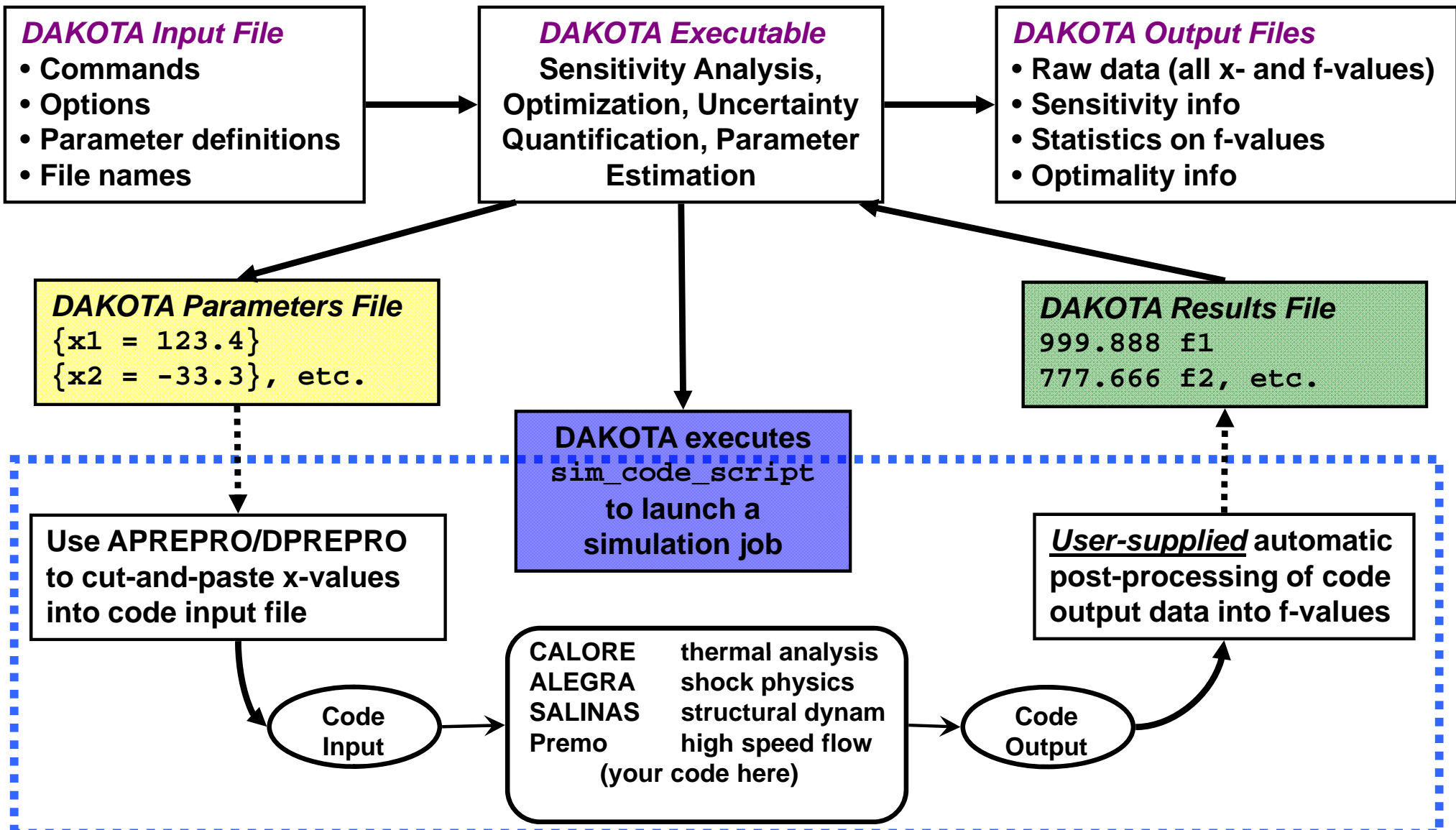


DAKOTA Application Interfacing

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

- **How DAKOTA interfaces with a simulation (computational model)**
- **Steps in building a black-box interface script:**
 - Integrate parameters from DAKOTA (aprepro/dprepro)
 - Run analysis
 - Post-process to capture results of interest
- **Build interface to your code and then share your experience**

DAKOTA Execution & Info Flow



DAKOTA Application Interfacing Class

Application Stand-in: Rosenbrock "Banana" Function

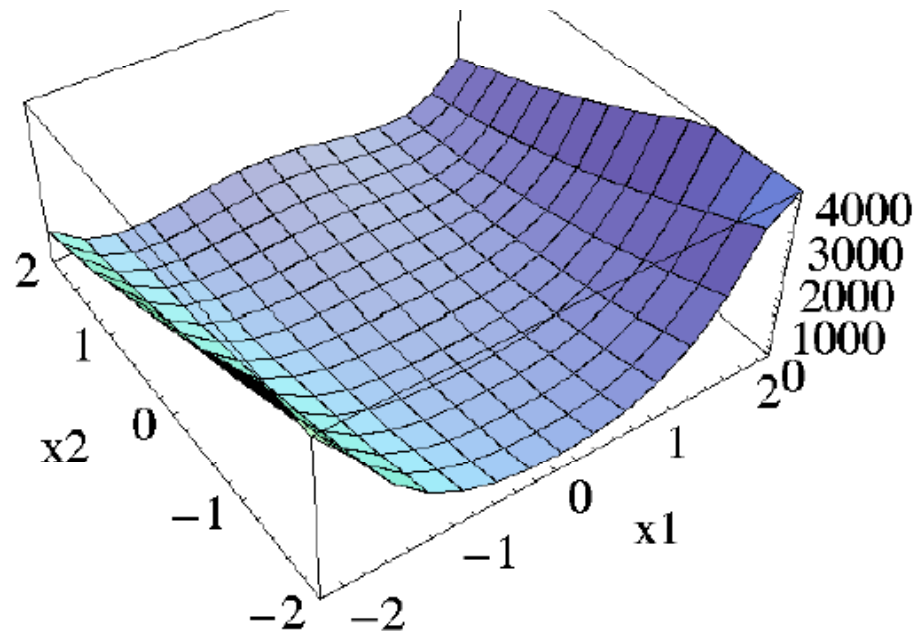
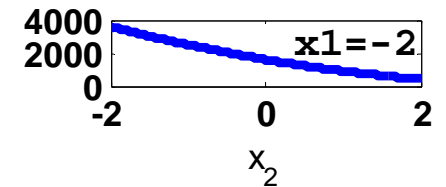
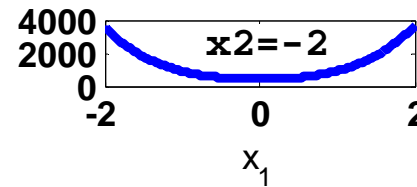
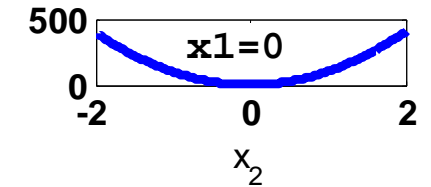
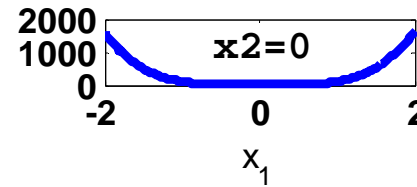
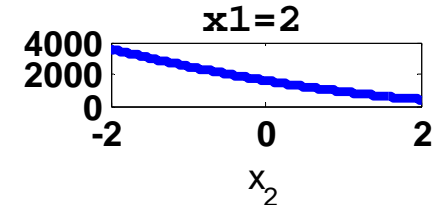
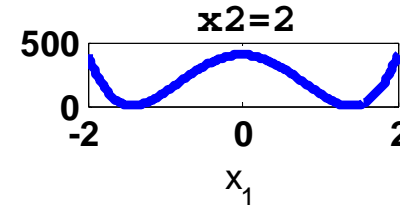
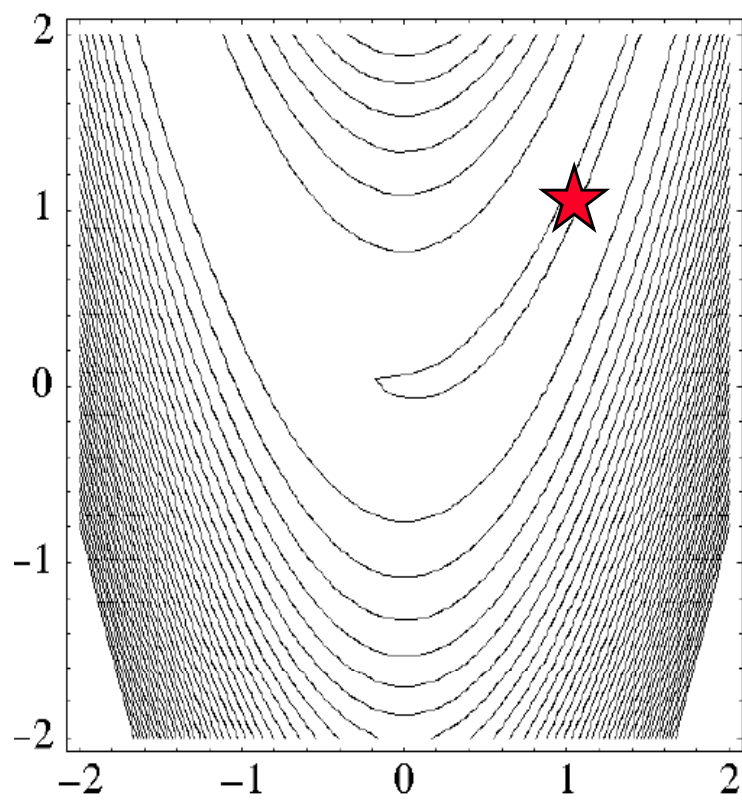


$$f(x_1, x_2) = 100 \cdot (x_2 - x_1 \cdot x_1)^2 + (1 - x_1)^2$$

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

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

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





Demo: Rosenbrock as a “black box”



- **Locate example in**
`Dakota/examples/script_interfaces/generic`
- **Described in DAKOTA 5.0 User's Manual 17.1**
- **Explore top-down (DAKOTA down to application and back)**
- **Since you're familiar with your application, may want to build from application up**

Interfacing to Your Simulation (Assuming Text-based I/O)



1. Annotate your input file to create template

```
{ stress }           { alpha1 }
```
2. Create a representative DAKOTA `params.in` file in `aprepro` format (see User's 12.6.2) and test:

```
dprepro params.in analysis.in.template analysis.in
```
3. Verify commands to run application with `analysis.in`
4. Determine how to automatically extract results of interest (direct application to export, shell commands, python, perl, visual basic, etc.) to create `results.out` (see User's 14.2)
5. Assemble into a script, e.g., `run_analysis.sh`; test script with sample `params.in`:

```
./run_analysis.sh params.in results.out
```
6. Test with a simple DAKOTA input deck, e.g., parameter study

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



Advanced Topics (dictated by 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 SNL-specific or issues involving proprietary information)