LA-UR-17-29214
Approved for public release; distribution is unlimited.

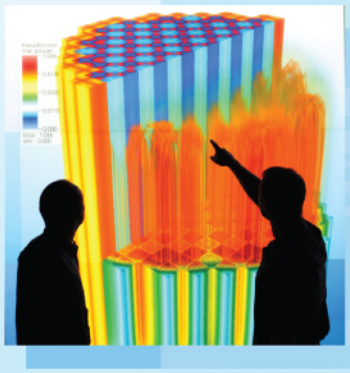
Title: CASL Dakota Capabilities Summary

Author(s): Adams, Brian M.
Simmons, Chris
Williams, Brian J.

Intended for: Report

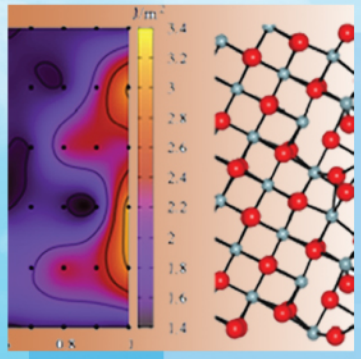
Issued: 2017-10-10

Power uprates
and plant life extension



Engineering design
and analysis

Science-enabling
high performance
computing



Fundamental science

Plant operational data

L2:VVI.P15.03

Consortium for Advanced Simulation of LWRs

CASL Dakota Capabilities Summary

CASL-U-2017-1446-000

Brian M. Adams[1]
Chris Simmons[2]
Brian J. Williams[3]

[1]Sandia National Laboratories
[2]University of Texas Austin
[3]Los Alamos National Laboratory

September 22, 2017

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

CASL Dakota Capabilities Summary

Brian M. Adams, Chris Simmons, Brian J. Williams, September 22, 2017

Dakota Background

The Dakota software project serves the mission of Sandia National Laboratories and supports a worldwide user community by delivering state-of-the-art research and robust, usable software for optimization and uncertainty quantification. These capabilities enable advanced exploration and risk-informed prediction with a wide range of computational science and engineering models [1]. *Dakota is the verification and validation (V&V) / uncertainty quantification (UQ) software delivery vehicle for CASL, allowing analysts across focus areas to apply these capabilities to myriad nuclear engineering analyses.*

In its simplest mode, Dakota automates iterative analysis using a general- purpose interface to a computational model, as shown in Figure 1. Its fundamental strength is a broad suite of algorithmic techniques facilitating parameter exploration, global sensitivity analysis, design optimization, model calibration, uncertainty quantification, and statistical inference. These core algorithms provide a foundation for more advanced, multicomponent solution approaches, including hybrid optimization, surrogate-based optimization, multi-fidelity uncertainty quantification and optimization, mixed aleatory-epistemic uncertainty analyses, and optimization under uncertainty. By integrating these capabilities within a single software tool, users can easily transition between different types of studies when exploring a computational model – from identifying to calibrating influential parameters and from characterizing the effect of uncertainties to performing design optimization in their presence.

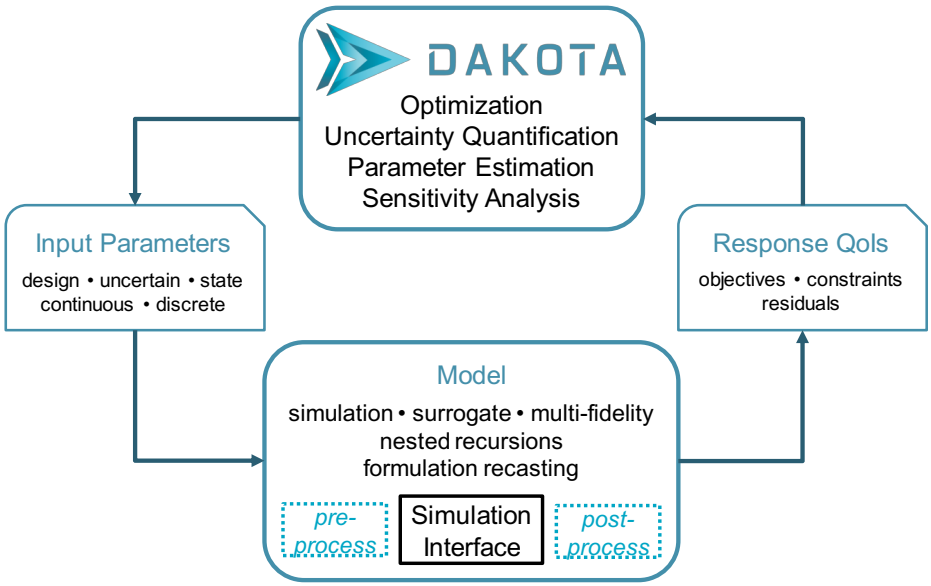


Figure 1. Interaction between Dakota and a parameterized simulation

Dakota's development activities span a spectrum from algorithm research and prototyping to production application deployment, with the goal of delivering exploration and prediction capabilities for all kinds of computational models. Efficient computing is also a central goal, with support ranging from desktops to the latest supercomputers. *The overall goal of CASL-funded Dakota work is to develop advanced methods for calibration and UQ, and deploy them through Dakota and VERA, to ultimately have impact on CASL challenge and other problems.*

Since CASL's inception in 2010, there have been **eleven major Dakota releases** with new capabilities, bug fixes, and robustness/usability improvements. Since 2014, Dakota has been released twice yearly in May and November. CASL/Dakota capabilities have been stewarded and developed by these **principal contributors:**

- **Sandia National Laboratories**: Brian Adams, Russell Hooper, Chris Jones, Kathryn Maupin, Vince Mousseau, Jim Stewart, and Laura Swiler
- **Los Alamos National Laboratory:** Brian Williams
- **The University of Texas at Austin:** Damon McDougall, Chris Simmons, and Roy Stogner
- **North Carolina State University:** Research group of Ralph Smith (including students)

Dakota Capabilities Emphasized by CASL

Since 2010, CASL has largely focused its Dakota investment on capabilities for forward uncertainty quantification, inverse uncertainty quantification (inference), surrogate (response surface) models, and parameter subspace identification. In addition, it has supported software deployment through VERA, application to various CASL challenge problems and fundamental physics studies, and outreach/education. The discussion here focuses primarily on Dakota developments in the past 3—4 years. It is by no means exhaustive; we aim to be representative.

Forward UQ Methods

Dakota algorithms for forward propagation of parametric uncertainties include sampling (Monte Carlo, Latin hypercube, and other variants), stochastic expansions (polynomial chaos and variants), reliability methods, and interval/evidence-based approaches. CASL largely relies on Dakota's historical investments in these methods, as they are production-ready for application use. One UQ feature specifically chartered and funded by CASL is a new implementation of **Wilks' sampling**, an NRC-typical Monte Carlo analysis. A user can specify the index of an order statistic and confidence level. Dakota will determine the requisite number of UQ samples, run the computational model and compute statistics on model outputs, including Wilks' tolerance intervals/bounds.

Leveraged Dakota forward UQ capabilities developed in the past few years include the following.

- **Sampling-based UQ:** New sampling options for D-optimal design selection and batch sampling and greater support for discrete variables in incremental LHS, all of which permit iterative design of computational experiments. More accurate and numerically stable correlation calculations and higher quality design generation with discrete variables.
- **Multi-fidelity/multilevel UQ methods**: With both sampling-based and polynomial chaos variants, these rigorous UQ methods, such as multi-level and control variate approaches, make UQ practical for costly models by leveraging lower fidelity calculations when available. They orchestrate the simultaneous execution of models of differing fidelity and/or numerical accuracy (Figure 2), to perform a UQ calculation.

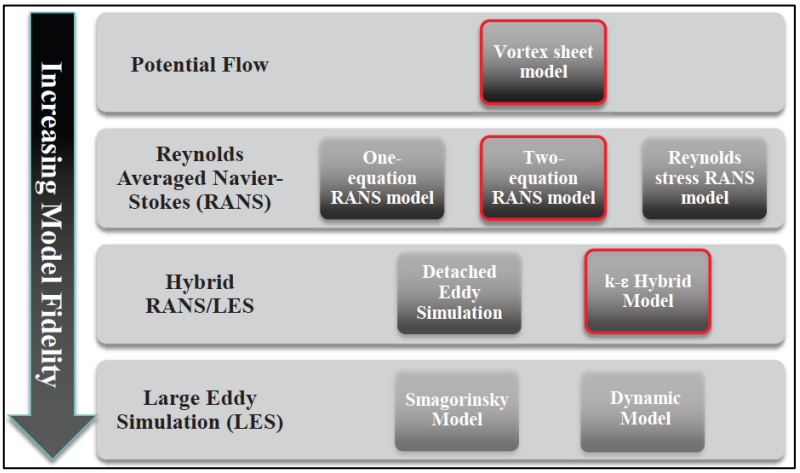


Figure 2. Multi-model UQ methods can use models of different forms and numerical accuracy.

- **PCE/SC:** Numerous improvements to polynomial chaos expansion and stochastic collocation methods for accuracy, robustness, and usability. New algorithm for adaptive basis selection for compressed sensing using expanding front.
- **Reliability methods:** New reliability method POFDarts works with any of Dakota's surrogate model types; adaptive importance sampling (including Gaussian process-based) improvements.

Model Calibration

Model calibration, also known as parameter estimation or inference, seeks to maximize agreement between a simulation model and a data source (whether experimental data or output from another

computational model). CASL has emphasized Dakota's **Bayesian calibration** methods in order to make uncertainty-aware calibration/prediction practical for CASL applications. For example, these methods allow an analyst to estimate joint distributions of parameters consistent with data, as shown in Figure 3.

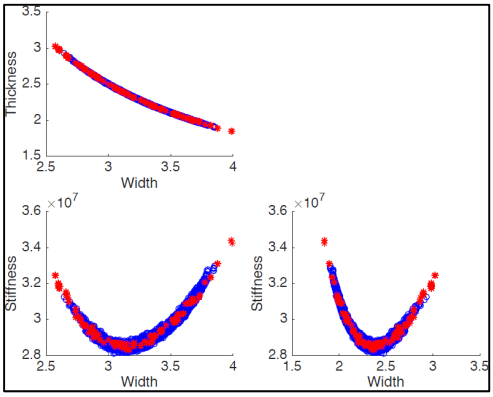


Figure 3. Bayesian inference methods allow estimation of joint parameter distributions consistent with experimental (or high-fidelity model) data.

QUESO for Bayesian Calibration

The Parallel C++ Statistical Library **Quantification of Uncertainty for Estimation, Simulation and Optimization (QUESO)** software library provides Dakota's principal Bayesian inference methods. Designed and primarily developed at The University of Texas at Austin, QUESO is a collection of statistical algorithms and programming constructs supporting research into the uncertainty quantification (UQ) of models and their predictions. It has been designed with three objectives in mind: (a) to be sufficiently abstract in order to handle a large spectrum of large-scale computationally intensive models; (b) to be extensible, allowing easy creation of custom-defined objects; and (c) leverage parallel computing through use of high-performance vector and matrix libraries. Such objectives demand a combination of an object-oriented design with robust software engineering practices. QUESO is written in C++, uses MPI, and utilizes libraries already available to the scientific community.

CASL funding has supported new capability development in QUESO and helped make it a robust, usable engine for Dakota integration. The process for integrating QUESO into Dakota has been streamlined, and Dakota's own interfaces improved to make Bayesian inference more practical for users. Dakota/QUESO improvements include:

- Support for DRAM, DR, AM, MH, and multi-level algorithms, with various options.
- Support for general prior distributions using any of Dakota's twelve continuous uncertain variables types, with no special syntax required.
- Scaling options: Users can scale parameters in Bayesian calibration problems using Dakota's standard probability transformations that map to well-scaled standard distributions. (This is

enabled by default when using certain kinds of surrogate models such as PCE or SC.) Likelihood scaling, which could be misleading, is now off by default.

- Basic user options through Dakota input can now be augmented by power user options for advanced users.

QUESO/GPMSA

Over the course of the last several years, with the help of CASL funding, QUESO has been extended to include the **Gaussian Process Models for Simulation Analysis (GPMSA)** framework developed by Brian Williams, Jim Gattiker, Dave Higdon, et al., at LANL. While originally written in Matlab, the GPMSA functionality has been re-implemented in C++ and is now part of QUESO. Owing to previous QUESO integration with Dakota, this new GPMSA functionality is now also available to Dakota users. With the help of CASL funding, the GPMSA functionality in QUESO continues to be extended and now provides Dakota users a high-performance alternative for their GPMSA needs. The remainder of this section discusses the extensions to the GPMSA implementation in QUESO with CASL funding.

Scalar and Multivariate Use Case: Overall, the quality and reliability of the GPMSA code in QUESO has improved greatly. For the scalar case, a suite of regression tests has been added to assert consistency with the Matlab implementation. These led to correcting several indexing bugs as well as improving the efficiency of some of the initial calculations. It's also allowed us to add defensive techniques such as asserting that user-provided covariance matrices are symmetric and positive definite and utilizing newer and safer memory management techniques for large matrices. Most of these implementation improvements carry over automatically, and have aided in the ongoing verification efforts, for the multivariate case.

Input options have been extended to allow the user more fine-grained control over problem setup for both the scalar and multivariate cases, including situations where the user may want to calibrate (or exclude) certain parameters from the problem. Users are now no longer responsible for normalizing the simulation/experiment output and scaling the calibration parameters. Functions for scaling and normalizing this data have been added to QUESO and we use these to sanity-check the users' input. This improves the usability and barrier to entry for users to leverage QUESO/GPMSA.

Functional Use Case: The QUESO GPMSA code has been extended, in a development branch, to handle mixtures of multivariate data (where each simulation and experiment output is an independent scalar) and functional data (where simulation output includes discretized field data and experiment outputs include values from those fields at particular space/time points.

A SimulationOutputMesh abstract base class allows us to extend to more complex discretizations in the future, and a TensorProductMesh allows us to support first-order interpolation for functions of up to 4 coordinates, on meshes which are rectilinear (but not necessarily equi-spaced) in the desired coordinate space. Multiple functional output variables, on independent grids, can be used as data to inform the same inverse problem. Experiment data can now vary from experiment to experiment, with each experiment specifying data variable index, and data point location in the case of functional data, for an independent number of experiment outputs.

Auto-scaling is now handled correctly for groups of discretized coefficients corresponding to functional variables. Auto-generation of Gaussian discrepancy bases is available, with a few input options to control kernel width and spacing, and with auto-normalization for kernel magnitude.

Multi-fidelity Model Calibration ("Hi2Lo")

Kathryn Maupin led development of an information theory-guided Hi2Lo calibration approach in Dakota based on [2]. As depicted in Figure 4, Dakota will generate "data" by running a high fidelity model across scenarios of interest and perform Bayesian calibration of a low fidelity model to best match the data. This process is iterated, using a mutual information criterion to add additional high fidelity runs until the estimated parameter distributions converge. The process can be executed in an online mode, where Dakota executes both models during the course of calibration, or an offline mode, where the user runs the high fidelity model manually at each proposed experiment design point.

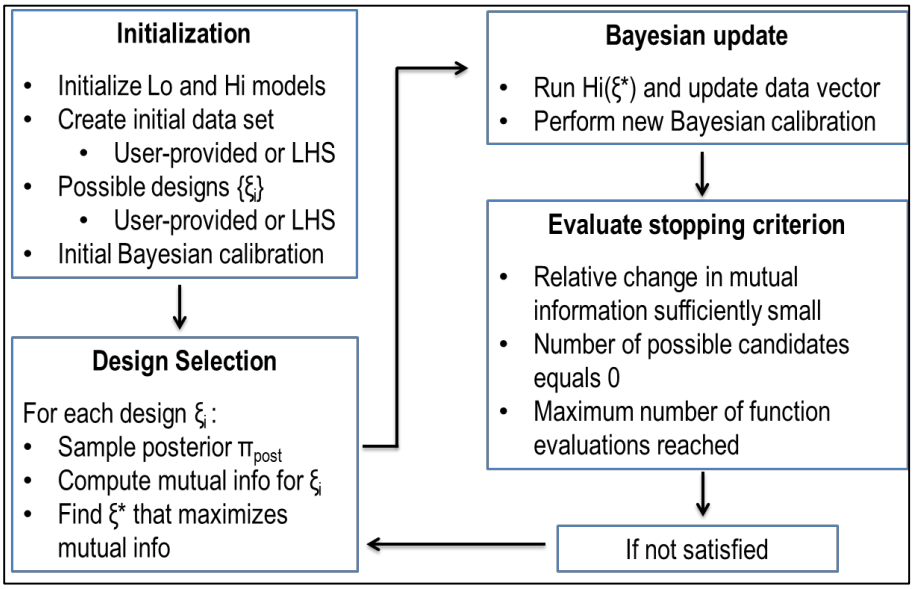


Figure 4. Mutual information-based design of experiments to calibrate a low-fidelity model to high-fidelity model output.

The Hi2Lo capability required adding **information theoretic metrics**, specifically approximate *k*-nearest neighbor-based algorithms for estimating mutual information and Kullback-Leibler discrepancy. These measure information gain between a Bayesian calibration prior and posterior and are used to select points at which to run additional high fidelity calculations in the Hi2Lo workflow.

General Calibration Method Enhancements

- **Bayesian inference:** Robustness and usability improvements made to Bayesian methods, including adaptive surrogate-based approaches. Features include ability to conduct forward UQ from a posterior chain, full support for Gaussian Process, PCE, and SC emulators, and reporting of final results (acceptance chain) in the original user-specified problem space. Users can opt to return simulation responses separately from experimental data or return computed residuals (model minus data) to Dakota.

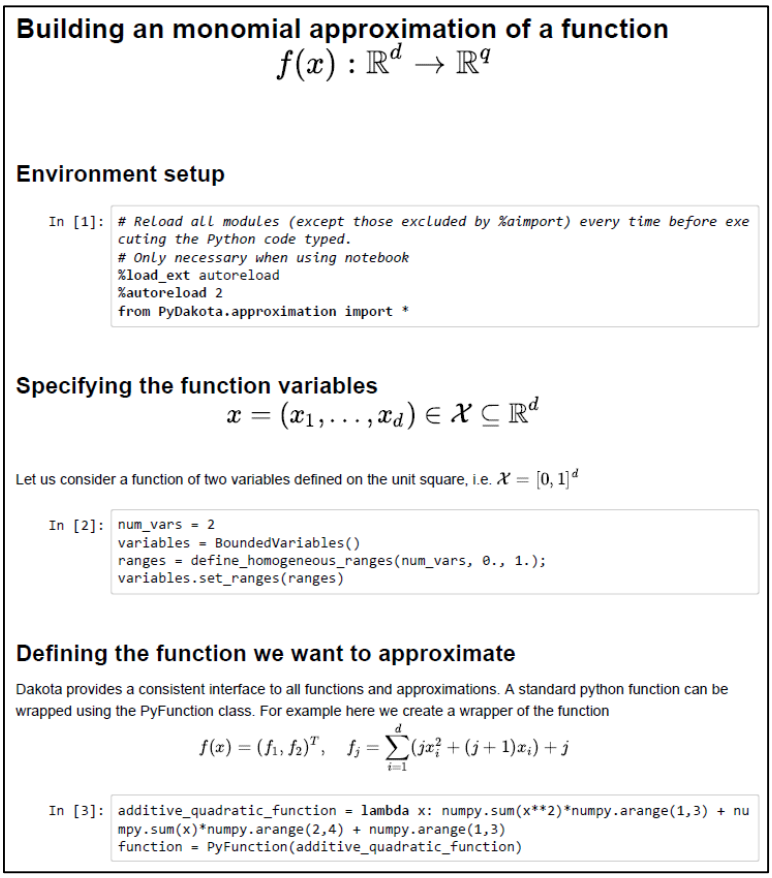
- **Bayesian results output:** Ability to discard chain burn-in and subsample the posterior chain; compute additional statistics including information gain from prior to posterior; more consistent file input and output, including for coupling Bayesian inference to forward prediction.
- **Credible and prediction intervals:** To aid in forward uncertainty quantification, Bayesian methods produce credible and prediction bounds/intervals for quantities of interest.
- **Bayesian proposal covariance:** Dakota now has three options for specifying the covariance of the proposal (jumping) distribution used in QUESO-based Bayesian inference: (1) default based on variance of the prior distribution, (2) user-provided either in Dakota input file or from separate files, and (3) adaptive based on gradient and/or Hessian information.
- **Experiment configuration variables:** Model calibration (including Bayesian inference) methods can now directly incorporate data from multiple experimental configurations/scenarios, reducing burden on users. Both deterministic and Bayesian calibration methods will evaluate the model at each of the scenarios in the course of calibrating its parameters. Scenarios can be specified through the calibration data file using continuous or discrete values.
- **Measurement (observational error) covariance:** For each response in a Dakota study, a user can now specify no variance, a single scalar variance for all observations of the response, or a diagonal or full covariance matrix compatible with the size of the response. This information will be incorporated into calibration formulations and its magnitude can be calibrated as a hyper-parameter.

Surrogate Models

Parameterized response surface (surrogate) models can serve as stand-in replacements for computationally costly models, when performing any kind of Dakota analysis. Some specialized Dakota methods can also use them in concert with actual simulation models to accelerate convergence to statistics or optimized designs.

CASL funding is being leveraged with other Dakota funding to refactor the surrogate modelling capability to be able to more easily add new features, improve user interfaces / usability, and improve testing and software quality. For example, a user will be able to (1) use an orthogonal polynomial (PCE) surrogate in other contexts as easily as a monomial or Gaussian process, or (2) be able to export a Dakota surrogate and then query the cheap replacement model through a Python interface or efficiently reuse it in other Dakota studies. As a practical CASL example, this capability would drastically reduce the per-iteration computation time in Hi2Lo calibration studies such as those currently being conducted with COBRA-TF and Star-CCM+.

Progress: As of this writing, considerable progress has been made to unify Dakota's surrogate models (from Dakota, Surfpack, and Pecos libraries) and a working prototype is available. Foundational utilities have been modularized and new surrogate model APIs designed, including Python interfaces to C++ model components for accessibility by power users. A Jupyter-based interactive Python tutorial (depicted in Figure 5) demonstrates more usable coordination of model components with Python, and that a tutorial can link to working C++ code for training purposes.



Building an monomial approximation of a function

$$f(x) : \mathbb{R}^d \rightarrow \mathbb{R}^q$$

Environment setup

```
In [1]: # Reload all modules (except those excluded by %aimport) every time before exe
        cuting the Python code typed.
        # Only necessary when using notebook
        %load_ext autoreload
        %autoreload 2
        from PyDakota.approximation import *
```

Specifying the function variables

$$x = (x_1, \ldots, x_d) \in \mathcal{X} \subseteq \mathbb{R}^d$$

Let us consider a function of two variables defined on the unit square, i.e. $\mathcal{X} = [0, 1]^d$

```
In [2]: num_vars = 2
        variables = BoundedVariables()
        ranges = define_homogeneous_ranges(num_vars, 0., 1.);
        variables.set_ranges(ranges)
```

Defining the function we want to approximate

Dakota provides a consistent interface to all functions and approximations. A standard python function can be wrapped using the PyFunction class. For example here we create a wrapper of the function

$$f(x) = (f_1, f_2)^T, \quad f_j = \sum_{i=1}^{d} (jx_i^2 + (j+1)x_i) + j$$

```
In [3]: additive_quadratic_function = lambda x: numpy.sum(x**2)*numpy.arange(1,3) + nu
        mpy.sum(x)*numpy.arange(2,4) + numpy.arange(1,3)
        function = PyFunction(additive_quadratic_function)
```

Figure 5. Jupyter-based interactive Python tutorial demonstrates new surrogate model capabilities (graphic truncated, for illustration purposes only).

The Dakota surrogate refactor will make it easier (in FY18) to include CASL-requested features such as user-selectable polynomial terms, new stepwise regression schemes, and information theoretic quality diagnostics (AIC, BIC). In addition, the following surrogate model features have been developed over the past few years, in part with CASL funding:

- **Iteratively refined surrogates:** New capability to incrementally refine a surrogate model by gathering more training data from a simulation (truth) model.
- **Composite surrogates:** Improvements to Voronoi Piecewise Surrogates allow users to automatically build response surface models for expensive simulations, including for

discontinuous simulation quantities of interest (Figure 6).

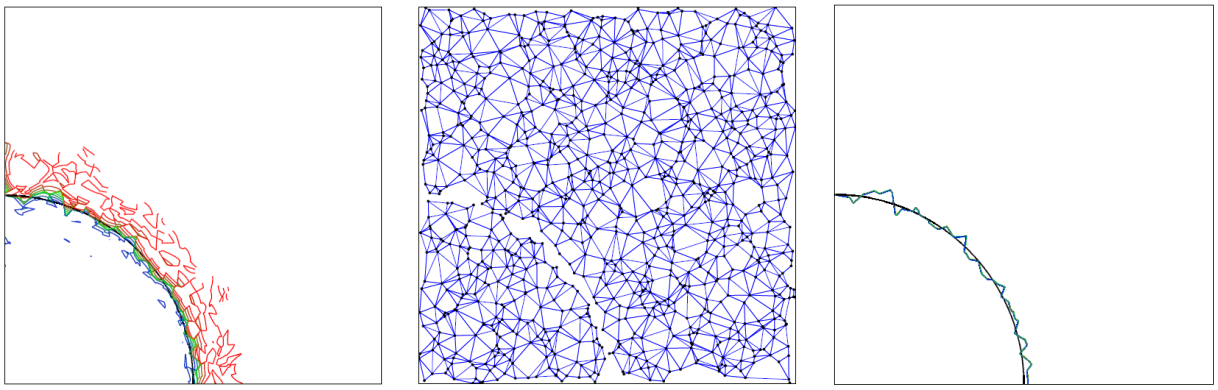


Figure 6. Voronoi Piecewise Surrogates can model local (in parameter space) response discontinuities.

- **Miscellaneous:** Extended derivative support in PCE and Dakota GP emulators; enabled automatic basis selection in neural network models; surrogates can be output to binary or algebraic files, so users can use in another context.

Active Subspace Methods

For models with many input parameters, active subspace methods automatically identify reduced parameter spaces to make UQ and optimization more tractable. CASL has primarily conducted research in gradient-free active subspace methods and has leveraged other Dakota investments in derivative-based approaches.

- **Gradient-based methods:** Substantial algorithm and usability improvements were made to gradient-based active subspace methods, including options for a number of subspace identification metrics and automated bootstrap validation. Subspaces can now be used in a number of UQ methods.
- **Gradient-free methods** will be improved in FY18 using lessons learned from the research programs of Smith, Constantine, and Williams/Khuwaileh.

Verification and Validation

For simulations that expose numerical solution parameters, Dakota can assist in conducting **verification studies**. For example, Dakota could be used to conduct parametric studies to assess the effect of

- Linear/nonlinear solver tolerances;
- Time step or time step control parameters;
- Discretization, given a knob controlling uniform or adaptive refinement or a discrete parameter to select from pre-generated grids; and
- Solution algorithm/solver choices.

Basic Dakota parameter studies assess the effect of varying any of these controls, while its sensitivity analysis methods can identify which solution control parameters most affect quantities of interest and therefore should be considered more carefully in formal verification studies.

Dakota's simple **Richardson Extrapolation method** allows a user to specify initial values of solution controls, e.g., mesh quality, along with a refinement rate, e.g., 1.5. Dakota will then evaluate the computational model with a sequence of meshes and either (1) estimate the order of convergence from three grids, (2) refine until the convergence order estimate stabilizes, or (3) refine until the response QoI converges. FY18 work will make more advanced verification metrics such as Rider's Robust Method of Regression to CASL studies.

Dakota's primary role in validation is in enabling **uncertainty-aware validation**, or statistical comparisons to experimental data. Both forward and inverse UQ methods can be used to generate predictive distributions accounting for parametric uncertainties as well as observation error. Dakota parametric studies can further support validation by assessing the effect of model closure laws, model form choices and associated parameters, or even discrete selection of alternative models. All together, these can help an analyst assess the validity of a simulation for a particular application, accounting for various source of uncertainty.

Deployment

Throughout CASL the Dakota team has worked to deploy Dakota and ensure it has impact. Some notable accomplishments include:

- **VERA Integration:** Dakota is deployed with VERA through the "DakotaExt" adapter to the full open source Dakota distribution. Examples are included in the VERA repository for using Dakota to study parameters of VERA simulations.
- **Milestones:** A number of sensitivity analysis, calibration, and UQ milestones in which Dakota was involved are included as tests and examples in the VERA repository. These demonstrate the use of Dakota for various challenge problem-relevant analyses on various HPC resources.
- **Best Practice Manual:** An extensive (180 page) CASL/Dakota manual "User Guidelines and Best Practices for CASL VUQ Analysis Using Dakota" summarizes best practices for Dakota use in CASL applications [3]. Also significant improvements have been made to the Dakota users' and reference manuals.
- **Platform support:** Using both CASL and leveraged funding, we have extended Dakota support to newer compiler, C++11, and MPI versions including Jenkins-automated build/test on wider array of platforms, including Windows. Deploying to CASL and external partner sites revealed additional issues that were resolved.

Education and Outreach

Partially developed in CASL, Dakota now has extensive training presentations, videos, and exercises as well as an interactive training virtual environment, available on the Dakota website: https://dakota.sandia.gov/training/dakota-training-materials. The materials were used as the basis of a number of Dakota training sessions:

- CASL Summer Student Workshops / Institutes (2015, 2016, 2017), including development of hands-on exercises

- March 2015: AREVA Training (1 day)
- Aug 2015, ORNL: Joint NEAMS/CASL training (3 days)
- Dec 2015: Westinghouse training (3 days)

Other Dakota Capabilities Benefitting CASL

CASL's investment in Dakota is leveraged with funding from DOE/NNSA/ASC, DOD/DARPA, DOE/SC, NEAMS, SNL LDRD, and CRADA funding. While not specifically chartered/funded by CASL, the following additional major capabilities and enhancements also benefit CASL Dakota analysts.

Algorithms and Architecture

- **Multi-fidelity methods:** New multi-level, multi-fidelity approaches make UQ and optimization analyses more practical for expensive high-fidelity computational models by leveraging lower fidelity calculations when available. These include multi-level and control variate Monte Carlo approaches for UQ and generalizing Dakota's existing multi-fidelity optimization to multiple levels/forms.
- **Mixed-integer optimization:** Enabled a new branch and bound method, useful for optimization/calibration of mixed continuous/discrete variables. Extended support for discrete categorical and string-values variables to additional solvers. Added NOMAD mesh adaptive search optimization solver.
- **Random fields:** New support for field-valued (functional) simulation responses and experimental data, together with field identification and propagation methods that enable field-valued UQ, e.g., propagation of time- or space-varying model inputs.
- **Parallel concurrency:** Support for increased algorithm parallel concurrency.

Usability and Deployment

- **GUI:** New Dakota GUI, including integration in Sandia Analysis Workbench (SAW), helps users create, run, and analyze Dakota studies. It includes features such as input file creation/editing, building interfaces to simulations, and visualizing results.
- **Capability maturity assessment:** A new taxonomy of Dakota concept together with updated capability maturity matrices and query tools allow analysts to determine Dakota algorithm and architecture maturity for a given problem. An example and test file browser helps links taxonomy terms (domain concepts) to Dakota unit, verification, and regression tests. These Dakota resources can be used by an analyst in support of a Predictive Capability Maturity Model (PCMM) assessment of their overall V&V/UQ application.
- **Concurrent research and production:** Implemented software development processes and workflows to better support concurrent research and production deployment to ensure Dakota remains leading edge while better meeting user needs.
- **Simulation interfacing:** Deployed new Python-based interfacing and parallel job management helpers and made Dakota-to-simulation interfaces including system/fork and working directories more robust.
- **User web portal:** All new http://dakota.sandia.gov with Dakota resources.
- **Testing:** Improved unit and system-level regression tests, and documented a sanctioned process for user acceptance testing.

- **File I/O:** Improved error handling and messages, consistency of tabular I/O formats, whitespace control, console output redirection, and working directories for concurrent methods.
- **Misc.:** Numerous other small features added and bugs fixed.

FY18 Plans

FY18 work will continue to refine these Dakota (and related) capabilities and make them increasingly ready for production work. Specific proposed activities include:

- Improve Hi2Lo Bayesian calibration algorithms, integrate QUESO/GPMSA, and generally make Bayesian inference more accessible and user-friendly.
- Extend active subspace methods to include additional gradient-free variants and be interoperable with more calibration and UQ methods.
- Complete initial surrogate model re-architecting and add new features.
- Deploy tools for verification, such as Rider's RMR method, and validation metrics.
- Make basic Dakota study types accessible to VERA users through simplified VERA input syntax and improved distribution of Dakota with VERA, including CASL-relevant examples.

References

[1] Adams, B.M., Bohnhoff, W.J., et al., Dakota, a Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.5 User's Manual, Technical Report SAND2014-4633, updated November 2016, Sandia National Laboratories, Albuquerque, NM. Retrieved from http://dakota.sandia.gov/documentation.html.

[2] Lewis A., Smith, R., Williams, B., and Figueroa, V., An Information Theoretic Approach to Use High-Fidelity Codes to Calibrate Low-Fidelity Codes, Journal of Computational Physics, 324, 24-43.

[3] Adams, B.M., Coleman, K., et al., User Guidelines and Best Practices for CASL VUQ Analysis Using Dakota, Technical Report SAND2016-11614, September 2016, Sandia National Laboratories, Albuquerque, NM.

Acknowledgment

This research is supported by and performed in conjunction with the Consortium for Advanced Simulation of Light Water Reactors (http://www.casl.gov), an Energy Innovation Hub (http://www.energy.gov/hubs) for Modeling and Simulation of Nuclear Reactors under U.S. Department of Energy Contract No. DE-AC05-00OR22725.