LA-UR-16-27626
Approved for public release; distribution is unlimited.

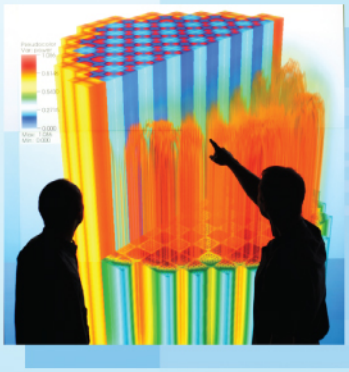
Title: User Guidelines and Best Practices for CASL VUQ Analysis Using Dakota

Author(s): Adams, Brian M.
Coleman, Kayla
Hooper, Russell W.
Khuwaileh, Bassam
Lewis, Allison
Smith, Ralph C.
Swiler, Laura P.
Turinsky, Paul J.
Williams, Brian J.

Intended for: Report

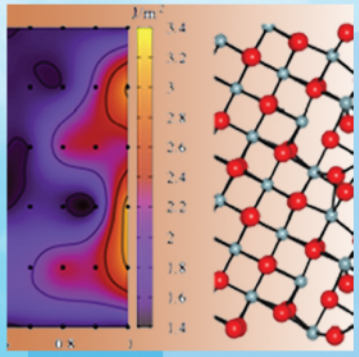
Issued: 2016-10-04

Power uprates
and plant life extension



Engineering design
and analysis

Science-enabling
high performance
computing



Fundamental science

Plant operational data

L2:VMA.P12.02

Consortium for Advanced Simulation of LWRs

User Guidelines and Best Practices for CASL VUQ Analysis Using Dakota

CASL-U-2016-1233-000

Brian M. Adams[1]
Kayla Coleman[2]
Russell W. Hooper[1]
Bassam A. Khuwaileh[3]
Allison Lewis[2]
Ralph C. Smith[2]
Laura P. Swiler[1]
Paul J. Turinsky[2]
Brian J. Williams[3]

[1]Sandia National Laboratories
[2]North Carolina State University
[3]Los Alamos National Laboratory

October 3, 2016

U.S. DEPARTMENT OF ENERGY | Nuclear Energy

User Guidelines and Best Practices for CASL VUQ Analysis Using Dakota

Brian M. Adams [1]
Kayla Coleman [2]
Russell W. Hooper [3]
Bassam A. Khuwaileh [4]
Allison Lewis [5]
Ralph C. Smith [6]
Laura P. Swiler [7]
Paul J. Turinsky [8]
Brian J. Williams [9]

October 3, 2016

[1]briadam@sandia.gov
[2]kdcolem2@ncsu.edu
[3]rhoope@sandia.gov
[4]bkhuwaileh@lanl.gov
[5]allewis2@ncsu.edu
[6]rsmith@ncsu.edu
[7]lpswile@sandia.gov
[8]turinsky@ncsu.edu
[9]brianw@lanl.gov

Contents

Chapter 1

Overview

Sandia's Dakota software (available at `http://dakota.sandia.gov`) supports science and engineering transformation through advanced exploration of simulations. Specifically it manages and analyzes ensembles of simulations to provide broader and deeper perspective for analysts and decision makers. This enables them to enhance understanding of risk, improve products, and assess simulation credibility.

In its simplest mode, Dakota can automate typical parameter variation studies through a generic interface to a physics-based computational model. This can lend efficiency and rigor to manual parameter perturbation studies already being conducted by analysts. However, Dakota also delivers advanced parametric analysis techniques enabling design exploration, optimization, model calibration, risk analysis, and quantification of margins and uncertainty with such models. It directly supports verification and validation activities. Dakota algorithms enrich complex science and engineering models, enabling an analyst to answer crucial questions of

- **Sensitivity:** Which are the most important input factors or parameters entering the simulation, and how do they influence key outputs?
- **Uncertainty:** What is the uncertainty or variability in simulation output, given uncertainties in input parameters? How safe, reliable, robust, or variable is my system? (Quantification of margins and uncertainty, QMU)
- **Optimization:** What parameter values yield the best performing design or operating condition, given constraints?
- **Calibration:** What models and/or parameters best match experimental data?

In general, Dakota is the Consortium for Advanced Simulation of Light Water Reactors (CASL) delivery vehicle for verification, validation, and uncertainty quantification (VUQ) algorithms. It permits ready application of the VUQ methods described above to simulation codes by CASL researchers, code developers, and application engineers.

More specifically, the CASL VUQ Strategy [33] prescribes the use of Predictive Capability Maturity Model (PCMM) assessments [37]. PCMM is an expert elicitation tool designed to characterize and communicate completeness of the approaches used for computational model definition, verification, validation, and uncertainty quantification associated with an intended application. Exercising a computational model with the methods in Dakota will yield, in part, evidence for a predictive capability maturity model (PCMM) assessment. Table 1.1 summarizes some key predictive maturity related activities (see details in [33]), with examples of how Dakota fits in.

Table 1.1: Summary of Dakota relevance for PCMM-related activities.

VUQ/PCMM Activity	**Dakota relevance**
select quantities of interest (QOIs)	(limited)
software quality assurance (SQA)	(limited)
code verification	conduct parameter studies as a function of mesh quality; calculate convergence rate
solution verification	conduct parameter studies over solver parameters or mesh quality; calculate convergence rate
validation	run ensemble of simulations and make (potentially uncertainty-aware) comparisons with experimental data, assess model form uncertainty
sensitivity	conduct global sensitivity analysis to rank or screen parameters; supports quantified parameter ranking table (QPRT)
uncertainty quantification	compute uncertainty in QOIs for risk-informed decision making
calibration	tune or refine models for use in particular scenarios

This manual offers CASL partners a guide to conducting Dakota-based VUQ studies for CASL problems. It motivates various classes of Dakota methods and includes examples of their use on representative application problems. On reading, a CASL analyst should understand why and how to apply Dakota to a simulation problem.

1.1 Manual Contents

This user's guide emphasizes best practice and highlights a few key approaches to solving the VUQ analysis problems of greatest interest to CASL today. The remainder of this chapter summarizes high level steps to getting started using Dakota. Chapter 2, Application Example Problems describes simple, but physically meaningful, application problems that will be used to demonstrate each of the Dakota algorithmic approaches described in subsequent chapters. Among the examples are CASL-relevant COBRA-TF thermal-hydraulic simulator and coupled COBRA-MPACT-MAMBA1D Crud Induced Power Shift (CIPS) phenomenon problems.

Chapters 3 through 6 tour four of Dakota's major algorithmic capabilities. In each chapter you will find high-level analysis goals and terminology, with references to more detailed descriptions and theory; guidance on how to choose from the available Dakota approaches and assess whether they are working; and application examples, including Dakota input, Dakota output, and post-processing/interpretation. The major VUQ activities addressed by this manual are:

- **Parameter Studies and Sensitivity Analysis:** Dakota parameter studies automate typical parameter variation studies such as running the model at a tensor grid of parameter values or varying them 1%, 5%, 10% from a nominal value. Sensitivity analysis determines model parameters most influential on quantities of interest (responses). This can be used to rank the influence of parameters, as in a Quantified Parameter Ranking Table (QPRT) [33], or

screen/down-select to a tractable number of free parameters for follow-on analyses. See Chapter 3, Sensitivity Analysis for an overview of parameter studies, global sensitivity analysis methods and metrics, and a demonstration of using Dakota to perform parameter ranking.

- **Surrogate Models:** Any of the Dakota studies described can be conducted directly on a computational model or with surrogate model indirection. Surrogate models are inexpensive approximate models that are intended to capture the salient features of an expensive high-fidelity model. In this manual and the Dakota context, surrogate models are not based on simplifying physical assumptions. Rather they are response surface models constructed automatically by Dakota based on empirical samples of the true simulation's input/output behavior. For example, in CASL one might run costly CFD simulations at a set of design points in a parameter space and then have Dakota build an algebraic Kriging model on the QOI data for use in optimization. Chapter 4, Surrogate Models has an overview of the most commonly used surrogate models, which can smooth noisy model responses, or reduce computational cost. On reading it, you will be able to create Dakota studies that automatically run a computational model, generate a response surface model, and evaluate it in the context of another Dakota study.

- **Calibration:** Dakota provides capabilities for automatically tuning model parameters to best match experimental (or high-fidelity model) data. This process is also known as parameter estimation, calibration, data assimilation, or model inversion to update knowledge of parameter values based on additional data. A CASL example would be tuning crud chemistry reaction rates to match experimental data. Approaches yielding single point estimates of parameters are described in Chapter 5, Optimization and Deterministic Calibration, while Bayesian methods resulting in a probability distribution for the unknown parameters are covered in Section 6.2, Bayesian Model Calibration.

- **Design Optimization:** Adjusting model parameters to meet desired performance criteria while satisfying other constraints. For example, determine optimal shape to minimize vibration, or design mixing vanes to minimize crud formation. Chapter 5, Optimization and Deterministic Calibration will help you choose from among Dakota optimization methods based on problem characteristics and your specific optimization goals.

- **Uncertainty Quantification (UQ):** Model predictions with quantified uncertainty support validation and follow-on decision making. UQ methods accept characterizations of input parameter uncertainty and run the computational model to compute the resulting uncertainties on response quantities of interest. UQ methods, which yield statistics on QOIs (mean, standard deviation, distribution, range), are described in Chapter 6, Uncertainty Quantification.

This guide selectively focuses on two to three ways to execute each type of VUQ analysis depending on goals and problem characteristics. The approaches included have worked well in practice on a broad range of problems in computational science and engineering. When challenges are encountered, many alternative and advanced methods that may perform better are available in Dakota.

The manual concludes with two CASL-relevant examples. The first is a realistic thermal-hydraulics example from a CASL "Progression Problem" in Chapter 7, COBRA-TF VUQ Studies. The example demonstrates a VUQ activity flow from initial parameter studies, through sensitivity analysis for parameter screening, to deterministic and Bayesian calibration on a reduced parameter set. Construction of the surrogate utilized in model calibration as a substitute for more costly direct COBRA-TF calculations is also illustrated. The second is a coupled thermal-hydraulics, neutronics, and crud chemistry example developed to model the CIPS phenomenon at the single assembly level

in Chapter 8, CIPS — Crud Induced Power Shift. Sensitivity analysis is conducted to identify important parameters, succeeded by uncertainty quantification to establish Wilks upper bounds on maximum assembly crud thickness and total boron.

Additional Resources

This user's guide is not an exhaustive guide to Dakota's capabilities. It is a high-level supplement to other Dakota and VUQ resources. Users reaching its extent should consult:

- The Dakota User's Manual [1]: a more complete summary of Dakota capabilities from getting started through advanced methods (`https://dakota.sandia.gov/sites/default/files/docs/6.4/Users-6.4.0.pdf`);
- The Dakota Reference Manual [2]: extensive guidance on valid keywords to use in a Dakota input file to specify a Dakota study (`https://dakota.sandia.gov//sites/default/files/docs/6.4/html-ref/index.html`);
- The directories `dakota/examples` and `dakota/test` included with Dakota distributions which contain examples of input files referenced in the documentation and many more (also available at `https://software.sandia.gov/trac/dakota/browser/branches/6.4/examples` and `https://software.sandia.gov/trac/dakota/browser/branches/6.4/test`);
- The Dakota Theory Manual [3] (`https://dakota.sandia.gov/sites/default/files/docs/6.4/Theory-6.4.0.pdf`) and research publications available at `http://dakota.sandia.gov/publications.html`: detailed background on algorithmic approaches developed directly in Dakota to tackle challenging science and engineering analyses; and
- Publications referenced throughout all the above.

This document refers to Dakota 6.4 and its documentation; newer versions may be available on the Dakota website. This guide for Dakota usage in CASL aims to be generic and thus does not supplant any domain-specific best practices or guidance for performing VUQ-related studies.

1.2 Getting Started with Dakota

The remainder of this manual largely focuses on selecting and applying Dakota methods and understanding the results. This section surveys at a higher level some prerequisites for using Dakota, with references to additional resources for help.

Know Why to Use Dakota

Understanding your simulation's characteristics, your VUQ analysis goals, and Dakota's relevance in achieving them are critical first steps. These likely seem obvious, but are crucial in order to select from the many available methods in Dakota. This guide aims to address this background by offering a high-level introduction to some key analysis methods, their application, and benefits. Other resources to understanding Dakota's applicability include training materials, publicity materials, and publications on the Dakota website `http://dakota.sandia.gov`, as well as the Dakota User's Manual [1].

Access the Software and Other Resources

Dakota: Dakota is available to CASL partners through the Virtual Environment for Reactor Applications (VERA). Binary distributions include the `dakota` executable and related utilities. In source distributions, Dakota appears under `DakotaExt/Dakota`. Examples of Dakota applied to CASL problems are evolving. These are archived in milestone reports and protected in the VERA software in the VUQDemos suite, available from the CASL Git repository. Dakota also has a public download site `http://dakota.sandia.gov/download.html`, which may be useful for CASL partners without ready access to the VERA development environment at Oak Ridge National Laboratory (ORNL).

This Manual and Examples: Examples from this manual, including input, output, auxiliary data files, and scripts are available from the VUQDemos section of the CASL Git repository. The LaTeXsource and images are also included.

Help Resources: For help beyond this manual and the documents referenced herein, see the Dakota website: `http://dakota.sandia.gov`. It includes software downloads, documentation, publications, and training materials. It also has guidance on seeking general help with Dakota via the `dakota-users` mailing list (`dakota-users@software.sandia.gov`). CASL-specific issues should be directed to `casl-vvi@casl.gov`.

Dakota Analysis Workflow

An overall Dakota analysis process is depicted in Figure 1.1. The specification of the VUQ analysis problem to Dakota is given in the text-based Dakota input file, including the method (algorithm) which dictates how Dakota generates parameter sets at which to run the user's simulation. As Dakota runs, it will iteratively evaluate the simulation at these parameter sets by running a user-provided analysis driver and collecting corresponding quantities of interest output by the simulation workflow. When complete, the Dakota executable will produce console text output and tabular data with VUQ results for subsequent analysis.

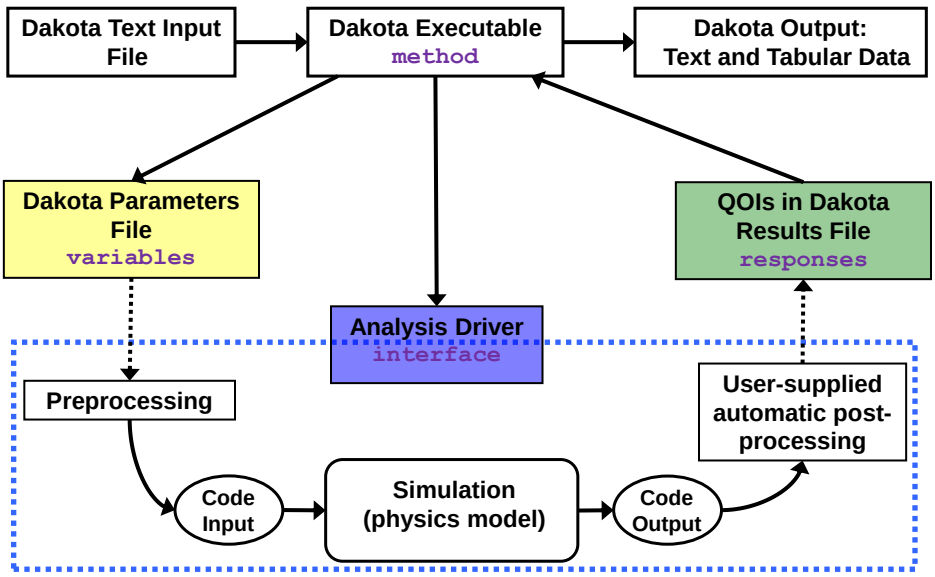


Figure 1.1: Components of a Dakota study, including Dakota input and output, and interface to a computational model (simulation).

Interface Dakota to the Simulation

Dakota requires an analysis driver to communicate with the computational model. The contract for this Dakota/simulation interface is straightforward: it must be an automated workflow that accepts Dakota parameters as input from a text file, runs the simulation in a batch/non-interactive mode, and produces responses (quantities of interest derived from simulation output) in a text file for consumption by Dakota.

As Dakota runs, it will determine values of parameters for which response data is needed. When ready to evaluate the simulation at such a parameter set, Dakota will write a "parameters file" with the values of the variables. Dakota will then invoke the specified analysis driver, represented by the dashed blue box in Figure 1.1. This analysis driver must implement the automated process that takes a Dakota parameters file as input and produces a Dakota results file as output. Typically this driver is a script which includes preprocessing, running the code, and postprocessing to extract QOIs from simulation output. When the driver completes, Dakota requires the "results file" to contain the quantities of interest resulting from running the simulation at the specified parameter values.

Additional resources for creating and running a Dakota/simulation workflow include:

- High level guidance in "User Supplied Simulation Code Examples" in the Dakota Tutorial in the Dakota User's Manual [1], with more details in the "Advanced Simulation Code Interfaces" chapter, together with example code in `dakota/examples/script_interfaces`.
- Considerations for having Dakota manage concurrent simulation runs in parallel, as this is a common need. Managing parallel concurrency locally, within queue, and out of queue, including batch submission and later retrieval are addressed by the Dakota User's Manual, "Application Parallelism Use Cases", together with examples in `dakota/examples/parallelism`.
- CASL-specific Dakota workflows with COBRA-TF and Insilico, demonstrated in the above referenced VUQDemos.

Understand Dakota Input Files

Once an interface is constructed between Dakota and the simulation, one may readily apply any Dakota method by simply changing the input file. Dakota input files are simple plain text files with six categories of information that can appear (some are optional and some may appear multiple times in advanced studies). Four of these are indicated notionally in Figure 1.1:

- `environment` (not depicted): overall control of Dakota methods and tabular output data.
- `method`: specifies the iterative analysis method being run on the model, for example Latin hypercube sampling or gradient-based parameter estimation.
- `variables`: characterization of the model parameters Dakota is varying in the study, such as lognormal uncertain or continuous design, together with supplementary fixed (state) parameters.
- `responses`: quantities of interest returned to Dakota for analysis.
- `interface`: the simulation workflow mapping variables to responses in an automated way; typically a script that orchestrates this workflow.

- `model` (not depicted): a container encapsulating a set of variables, interface, and responses for presentation to a method; useful for specifying that a surrogate model should be automatically constructed to serve as a proxy for an expensive computational model.

This guide shows and explains a number of examples of input files which configure Dakota to conduct various kinds of iterative analyses. They can be taken verbatim from the text to conduct Dakota studies and will also be available from CASL records systems when this manual is published. Additional examples are available in the various Dakota manuals and with the Dakota software itself. Specific guidance on individual Dakota keywords is available in the Dakota Reference Manual [2], which can help when trying to determine how to configure Dakota for a new kind of study.

Understand Dakota Results

This document offers an introduction to Dakota output, including log file output and tabular data to understand the results of studies. Often this data must be interpreted or post-processed with external tools to be useful in a decision making context. Examples are included in this manual to demonstrate this.

1.3 Acknowledgments

This CASL/Dakota manual borrows heavily from the Dakota User's Manual [1]. We heartily thank the authors of the most recent version of that document: Brian M. Adams, Lara E. Bauman, William J. Bohnhoff, Keith R. Dalbey, John P. Eddy, Mohamed S. Ebeida, Michael S. Eldred, Russell W. Hooper, Patricia D. Hough, Kenneth T. Hu, John D. Jakeman, Kathryn A. Maupin, Jason A. Monschke, Ahmad Rushdi,Laura P. Swiler, J. Adam Stephens, Dena M. Vigil, and Timothy M. Wildey.

Dakota's use of the Quantification of Uncertainty for Estimation, Simulation, and Optimization (QUESO) library is facilitated through close interaction with its developers at the University of Texas at Austin. Ernesto E. Prudencio, Nicholas Malaya, and Damon McDougall have kindly provided examples and test problems implemented in QUESO.

We also appreciate the efficient implementation of code changes to COBRA-TF made by Noel K. Belcourt to expose code parameters to Dakota needed to drive the various COBRA-TF parameter studies described in this manual.

David Salazar and Andrew Godfrey provided invaluable expertise which facilitated execution of the Crud Induced Power Shift (CIPS) example of Chapter 8.

We express our deep thanks and gratitude towards North Carolina State High Performance Computing (NCSU-HPC) for providing the required computer allocation to support the calculations conducted in Appendix C, ROMUSE2.0 User Manual.

We appreciate the valuable feedback provided by several reviewers of version one of this manual, including Patty Hough, Vince Mousseau, Rod Schmidt, and Dena Vigil.

This research was supported by 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. Sandia National Laboratories (SNL), North Carolina State University (NCSU), and Los Alamos National Laboratory (LANL) are core CASL partners.

Chapter 2

Application Example Problems

This chapter describes representative, though simplified, application problems that will be used to demonstrate various Dakota approaches. Each of the four application examples has strengths to help bridge abstract Dakota concepts and guidance throughout the remainder of this manual to concrete practice:

- **Cantilever beam:** A simple static mechanics analysis where prescribed geometry, material properties, and loads on a cantilevered beam map to quantities of interest such as weight, displacement, and stress. The physical meaning is intuitive, the physics equations have a simple algebraic form, and a simulator for it is included with Dakota, along with several example input files. Each Dakota analysis technique in Chapters 3 through 6 includes a worked example using the cantilever beam problem.

- **General linear model:** A model with closed algebraic form specifically designed to support algorithm and code verification, i.e., to verify that VUQ algorithms are working as expected. This example consists of a linear mapping from model parameters to responses, with an additive noise term. It is used to verify the performance of Bayesian calibration methods in Section 6.2.6.

- **COBRA-TF thermal-hydraulics:** A coupled physics, single assembly reactor model problem simulated with CASL's COBRA-TF thermal-hydraulics code. A physically realistic example, this problem demonstrates VUQ infrastructure for varying model form and other parameters, running a computational model, and distilling quantities of interest from code output. This example is the focus of Chapter 7, where it is used to demonstrate parameter screening, calibration, and surrogate construction. It also serves as the precursor for the next more complicated CASL problem.

- **Coupled COBRA-MPACT-MAMBA1D Crud Induced Power Shift:** A coupled multi-physics, single assembly reactor model problem simulated with CASL's COBRA-TF, MPACT (neutronics) and MAMBA1D (crud chemistry) codes coupled together. The most complex physically realistic example, this problem employs the approach for the standalone COBRA-TF UQ study now extended to one of the more challenging phenomena addressed in CASL, Crud Induced Power Shift (CIPS). UQ results for this problem are described in of Chapter 8.

2.1 Cantilever beam

The cantilever beam example problem is adapted from the reliability-based design optimization literature [45], [55]. The uniform cantilever beam is shown in Figure 2.1, with a left anchor and a fixed length $L = 100$`in`. The beam width and thickness are parameterized by w and t, respectively. The free end of the beam is subject to horizontal load X and vertical load Y.

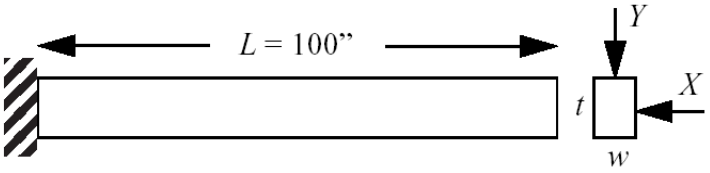


Figure 2.1: Cantilever beam test problem.

Given Young's elastic modulus E, the simplified algebraic physics equations used to model the beam area A (a stand-in for weight W), stress S, and displacement D are:

$$\begin{aligned} W &\propto A = wt \\ S &= \frac{600}{wt^2}Y + \frac{600}{w^2t}X \\ D &= \frac{4L^3}{Ewt}\sqrt{\left(\frac{Y}{t^2}\right)^2 + \left(\frac{X}{w^2}\right)^2}. \end{aligned} \tag{2.1}$$

Given a specified maximum displacement D_0 and yield stress R, the quantities of interest for the Dakota study (Dakota "responses") are defined to be:

$$\begin{aligned} \texttt{area} &= wt \\ \texttt{stress} &= S - R = \frac{600}{wt^2}Y + \frac{600}{w^2t}X - R \\ \texttt{displacement} &= D - D_0 = \frac{4L^3}{Ewt}\sqrt{\left(\frac{Y}{t^2}\right)^2 + \left(\frac{X}{w^2}\right)^2} - D_0, \end{aligned} \tag{2.2}$$

where `area` is used as a surrogate for weight, and `stress` and `displacement` are defined as differences with respect to the prescribed values of stress and displacement. The parameters in the problem are summarized with their nominal values in Table 2.1. Parameters w and t have simple bounds, while R, E, X, and Y may be considered as random variables and modeled with normal distributions $\mathcal{N}(\mu, \sigma^2)$ when conducting uncertainty studies.

The built-in Dakota analysis driver `mod_cantilever` computes the outputs `area`, `stress`, and `displacement` given specified inputs w, t, R, E, X, and Y. These cantilever input/output mappings will be utilized throughout this manual to illustrate the application of core CASL VUQ technologies with Dakota. In Section 3.2, the sensitivity of the quantities of interest with respect to the input parameters is assessed to rank their importance and exercise the model.

Then, in Sections 4.1.1 and 4.2.1, response surface models (surrogates) for the parameter to response mapping are generated based on a small number of sampled runs of cantilever. This emulates the practical process one must use when models are costly. In Section 5.2, a deterministic design optimization problem is solved to design the beam geometry. The goal is to minimize the weight (or equivalently, the cross-sectional area) of the beam subject to a displacement constraint

Table 2.1: Parameters for the cantilever beam problem.

parameter	description	nominal value	range/distribution
w	width	2.5	[1.0, 4.0]
t	thickness	2.5	[1.0, 4.0]
R	yield stress	40000	$\mathcal{N}(4.0E5, 4.0E6)$
E	Young's modulus	2.9E7	$\mathcal{N}(2.9E7, 2.1025E12)$
X	horizontal load	500	$\mathcal{N}(500, 1.0E4)$
Y	vertical load	1000	$\mathcal{N}(1.0E3, 1.0E4)$
L	beam length	100	-
D_0	maximum displacement	2.2535	-

and a stress constraint. The parameters R, E, X, and Y are fixed at their nominal values and the deterministic design problem is given by

$$\begin{aligned} \texttt{minimize} \quad & \texttt{area} = wt \\ \texttt{subject to} \quad & \texttt{stress} = S - R \leq 0 \\ & \texttt{displacement} = D - D_0 \leq 0 \\ & 1.0 \leq w \leq 4.0 \\ & 1.0 \leq t \leq 4.0 \end{aligned} \tag{2.3}$$

In Section 5.2.1, the cantilever beam is calibrated to synthetic experimental data for area, stress, and displacement, to find the values of w, t, and E yielding best agreement with the data. In Chapter 6, the cantilever beam is utilized for both uncertainty quantification and Bayesian calibration. In the uncertainty quantification study, the design variables are fixed at their nominal or optimal values, and the Dakota study is conducted over the normally-distributed uncertain parameters R, E, X, and Y. This yields estimates of the mean, standard deviation, and overall distribution of the quantities of interest. In the Bayesian calibration study, synthetic data are generated and the objective is to determine probability distributions for uncertain parameters given these data and an initial assessment of uncertainty in these parameters. The performance of two algorithms for sampling the final parameter distributions is compared for two cases.

2.2 General Linear Model Verification Test Suite

Whereas most CASL codes exhibit a nonlinear input-output relation, linearly parameterized problems serve an important role for algorithm and code verification. These uses include the following:

- They provide a hierarchy of models, which can be used to test the convergence of Bayesian model calibration algorithms through comparison with analytic solutions.
- They provide a regime to test the accuracy of uncertainty propagation algorithms since one can employ analytic relations between input and output densities.
- They facilitate the testing of algorithms for heavy-tailed distributions.
- They provide a framework for analytically testing algorithms to construct Sobol global sensitivity indices (described in Section 3.1.2).

The family of linear models described in this section is used to verify the performance of Bayesian calibration methods in Section 6.2.6. A high-level overview of the problem appears here, with technical details relegated to Appendix A. A simulator implementing an even more comprehensive verification test suite is available on request from the authors.

We employ the linear regression model

$$Y = G\beta + \varepsilon(\lambda, \phi) \tag{2.4}$$

as a test problem for verifying the Bayesian calibration capabilities in Dakota, which currently include Quantification of Uncertainty for Estimation, Simulation, and Optimization (QUESO) and Differential Evolution with Self-Adaptive Randomized Subspace Sampling (via the DREAM software package). In this model Y is the N-dimensional vector of noisy observations of the quantity of interest. The $N \times N_\beta$ matrix G is known and the N_β components of the vector β are unknown regression parameters to be estimated. The N-dimensional random variable ε represents the random noise in the observations. This noise is normally distributed with mean zero and $N \times N$ covariance matrix $R(\phi)/\lambda$. The precision (inverse variance) λ of the ε process is a positive scalar and the permissible values for the correlation structural parameter ϕ depend on the type of correlation being considered as discussed below.

The data for the calibration problem is generated using (2.4) after selecting values for β, λ, and ϕ which are considered the true parameter values (designated β_0, λ_0, and ϕ_0). Appendix A.1 provides additional details on the data generation process. Table 2.2 summarizes two separate cases distinguished by the choice of parameters to be calibrated (or equivalently, by the set of true values considered to be known).

Case	**Calibrated**	**Known**
1	β	λ_0, ϕ_0
2	β, λ	ϕ_0

Table 2.2: Cases considered for the general linear model verification test suite.

In Case 1, we estimate the regression parameters β assuming the statistics of the observational errors are perfectly characterized. Case 2 removes explicit knowledge of the error ε precision λ. A third case, described in Appendix A.1 but not considered further in this manual, assumes that the correlation structure $R(\phi)$ of the observational errors is only qualitatively known and estimates its parameters ϕ.

The likelihood function used in calibration is proportional to

$$\frac{\lambda^{N/2}}{\det\left(R(\phi)\right)^{1/2}} \exp\left[-\frac{\lambda}{2}\left(y - G\beta\right)^T R^{-1}(\phi)\left(y - G\beta\right)\right].$$

For cases where the true value of λ is known, $\lambda = \lambda_0$ is used for computing the likelihood. Similarly, $\phi = \phi_0$ in likelihood calculations when ϕ is known.

Each of the three cases has two subcases defined by whether an informative or noninformative prior is specified for β. The informative prior weights the regression parameter space to indicate a belief that the true parameters are more likely to lie within certain subsets of the parameter domain, while the noninformative prior is agnostic with respect to the location of the regression parameters.

For Case 1, the informative prior for β is specified via a Gaussian random variable having fixed N_β-dimensional mean vector μ_0 and diagonal $N_\beta \times N_\beta$ covariance matrix

$$\frac{1}{\lambda_0}\begin{bmatrix} \frac{1}{r_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{r_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{r_{N_\beta}} \end{bmatrix},$$

where $r_1, \ldots, r_{N_\beta}$ are fixed positive hyperparameters.

In Case 2, the informative prior for β is the same as in Case 1, and λ_0 replaced by a random variable λ having Gamma prior density with parameters $a > 0$ and $b > 0$,

$$\pi(\lambda) = \frac{b^a}{\Gamma(a)}\lambda^{a-1}\exp(-b\lambda)\,.$$

This distribution has mean a/b and variance a/b^2. A noninformative prior for λ is found by taking $a = b = 0$ in the Gamma density function, resulting in $\pi(\lambda) \propto 1/\lambda$. This prior specification is invariant to bijective transformation, i.e. it does not depend on how scale is represented in the error process ε.

A noninformative prior for β specifies a uniform density $\pi(\beta) \propto 1$. The Jeffreys noninformative prior for (β, λ) in Case 2 is specified by $\pi(\beta, \lambda) \propto 1/\lambda$, which is invariant to bijective transformations of both location and scale parameters.

Under the assumption of independent and identically distributed errors ε_i, the sample means of the parameters being calibrated converge to their corresponding true values as the number of measurements, N, is increased. Furthermore, for any N and errors ε_i sampled from a mean-zero process having arbitrary covariance structure, the distribution of parameters sampled from QUESO converges to a probability distribution known analytically for (β, λ) when ϕ is fixed as the number of QUESO samples is increased. Appendix A.1 provides a more general and rigorous specification of prior distributions for the three cases as well as analytical results for calibrated parameter distributions.

Two types of correlation structure are considered. The types and corresponding domain of ϕ follow:

1. No correlation, no ϕ dependence.

2. Order 1 autoregressive correlation, $-1 < \phi < 1$.

For each correlation type, a correlation function and the resulting correlation matrix $R(\phi)$ are provided in Appendix A.2. The no correlation case indicates no correlation between output measurements and no dependence on ϕ. For order 1 autoregressive correlation, any two output measurements y_i, y_j become less correlated the further apart they are in index (i.e., as $|i - j|$ increases).

Qualitative comparisons of marginal posterior distributions obtained from Dakota MCMC samples to analytical solutions are made by visually inspecting plots of both distributions for each uncertain parameter. A more rigorous assessment of Dakota MCMC tools is provided by conducting a statistical test of the hypothesis that joint posterior samples from Dakota MCMC and joint posterior samples from the corresponding analytical solutions arise from the same multivariate distribution. Appendix A.3 describes the procedure based on energy statistics used in Section 6.2.6 for testing the equality of these two distributions.

2.3 COBRA-TF Thermal-Hydraulics Simulation Problem

This section provides an overview of COBRA-TF and a particular thermal-hydraulics simulation problem, CASL VERA Progression Problem 6. An end-to-end demonstration of Dakota methods using this COBRA-TF model is presented in Chapter 7, COBRA-TF VUQ Studies. The full Dakota/COBRA-TF example is available in the CASL software repositories. Details on accessing it are provided in Appendix B.

2.3.1 COBRA-TF Simulator Overview

COBRA-TF is a thermal-hydraulic (T/H) simulation code designed for light water reactor (LWR) analysis) [5]. COBRA-TF has a long lineage back to the original COBRA computer code developed in 1980 by Pacific Northwest Laboratory, under sponsorship of the Nuclear Regulatory Commission (NRC). The original COBRA began as a thermal-hydraulic rod-bundle analysis code, but subsequent versions have updated and expanded over the past several decades to cover almost all steady-state and transient analyses of both pressurized water reactors (PWRs) and boiling water reactors (BWRs). COBRA-TF is currently developed and maintained by the Reactor Dynamics and Fuel Modeling Group (RDFMG) at the North Carolina State University (NCSU). Additional information can be found at the RDFMG website, `https://www.ne.ncsu.edu/rdfmg/cobra-tf`.

COBRA-TF includes a wide range of thermal-hydraulic models important to LWR safety analysis including flow regime dependent two-phase wall heat transfer, inter-phase heat transfer and drag, droplet breakup, and quench-front tracking. COBRA-TF also includes several internal models to help facilitate the simulation of realistic fuel assemblies. These models include spacer grid models, a fuel rod conduction model, and built-in material properties for both the structural materials and the coolant (i.e., steam tables).

COBRA-TF uses a two-fluid, three-field representation of the two-phase flow. The equations and fields solved are:

- Continuous vapor (mass, momentum and energy)
- Continuous liquid (mass, momentum and energy)
- Entrained liquid drops (mass and momentum)
- Non-condensable gas mixture (mass)

Reasons for selecting COBRA-TF as the primary T/H solver in the VERA core simulator include: reasonable run-times compared to CFD (although CFD will be available as an option), the fact that it is being actively developed and supported by NCSU, ability to support future applications of VERA such as transient safety analysis and BWR and SMR applications.

2.3.2 COBRA-TF test problem description

The thermal-hydraulics application example problem used in this manual is a coupled single assembly problem known in CASL as Progression Problem 6 [38]. It simulates a single PWR assembly based on the dimensions and state conditions of Watts Bar Unit 1 Cycle 1. The dimensions for the assembly are identical to AMA Progression Benchmarks "Problem 3" and "Problem 6". Problems 3 and 6 are identical, except that Problem 3 is at Hot Zero Power (HZP) and has no T/H feedback, and Problem 6 is at Hot Full Power (HFP) and includes T/H feedback. The test case was run at a boron concentration of 1300 ppm and a 100% power level.

The assembly is a standard 17x17 Westinghouse fuel design with uniform fuel enrichment. There are no axial blankets or enrichment zones. The assembly has 264 fuel rods, 24 guide tubes, and a single instrument tube in the center. There are no control rods or removable burnable absorber assemblies in this problem. The primary geometry specifications of the fuel rod and guide tube materials are given in Figure 2.2 and Table 2.3. The geometry specification for the assembly is given in Figure 2.3 and Table 2.4. The thermal-hydraulic specifications for this problem are shown in Table 2.5.

The COBRA-TF results in this CASL/Dakota manual use a simplified and more efficient adaptation of Progression Problem 6. This study involves only the thermal hydraulic component of Progression Problem 6, holding constant the power supplied by the neutronics component in the full problem. In practice, the neutronics component in the full problem has proved to be at least an order of magnitude more computationally expensive than the thermal hydraulics component. This adapted problem allows relatively rapid and representative sensitivity studies to be performed while expediting testing and refinement of the CASL VUQ software toolset that drives these studies.

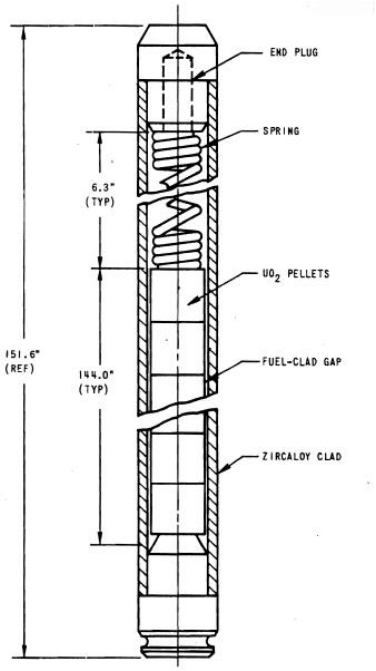


Figure 2.2: COBRA-TF Problem 6 fuel rod diagram.

2.3.3 VUQ Parameters in COBRA-TF Problem 6

At present, CASL VUQ workflows support COBRA-TF simulation parameter variation via two mechanisms. The first allows Dakota to seamlessly integrate with the VERA Common Input tool suite to perturb any parameters exposed to a user. The second path targets specific code parameters in the thermal hydraulics code that represent physical phenomena modeled with closure laws in COBRA-TF. These "VUQ parameters" are not exposed to a normal user but are instead exposed to Dakota using an auxiliary input file. The principle is that most analyst users should only perturb input data appearing in the VERA text input file, while advanced VUQ users may need to perturb more advanced parameters such as closure laws. The COBRA-TF studies presented in Chapter 7 use the second mode of parameter variation, i.e. perturbing code parameters via the auxiliary file.

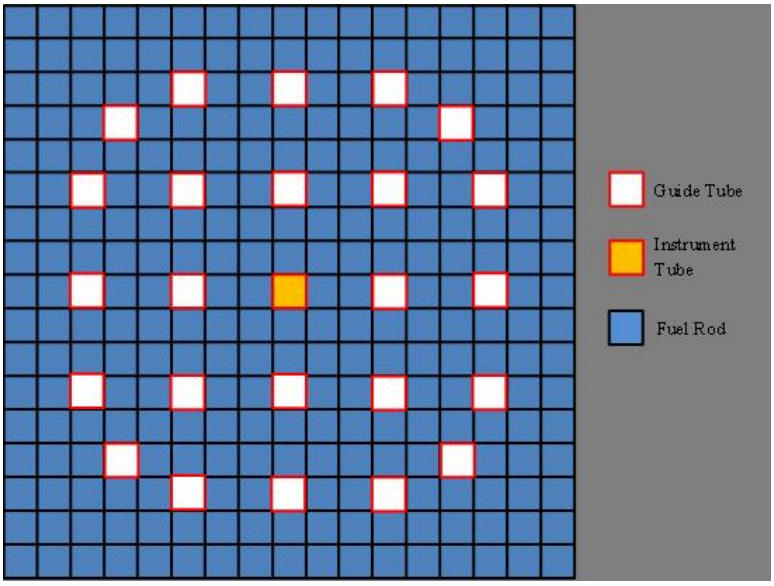


Figure 2.3: COBRA-TF Problem 6 assembly layout showing guide tubes and instrument tube placement.

Table 2.3: COBRA-TF Problem 6 fuel rod and guide tube descriptions.

Parameter	Value	Units
Fuel Pellet Radius	0.4096	cm
Fuel Rod Clad Inner Radius	0.418	cm
Fuel Rod Clad Outer Radius	0.475	cm
Guide Tube Inner Radius	0.561	cm
Guide Tube Outer Radius	0.602	cm
Instrument Tube Inner Radius	0.559	cm
Instrument Tube Outer Radius	0.605	cm
Outside Rod Height	385.10	cm
Fuel Stack Height (active fuel)	365.76	cm
Plenum Height	16.00	cm
End Plug Heights (x2)	1.67	cm
Pellet Material	UO2	
Clad / Caps / Guide Tube Material	Zircaloy-4	

Table 2.4: COBRA-TF Problem 6 assembly specification.

Parameter	Value	Units
Rod Pitch	1.26	cm
Assembly Pitch	21.5	cm
Inter-Assembly Half Gaps	0.04	cm
Geometry	17x17	
Number of Fuel Rods	264	
Number of Guide Tubes	24	
Number of Instrument Tubes	1	

For each parameter, Dakota is able to apply perturbations representing combined shift and scaling, e.g. for an arbitrary parameter p, Dakota can specify values for k_p and ka_p which are used

Table 2.5: COBRA-TF Problem 6 nominal thermal-hydraulic conditions.

Parameter	Value	Units
Inlet Temperature	559	degrees F
System Pressure	2250	psia
Rated Flow (100% flow)	0.6824	Mlb/hr
Rated Power (100% power)	17.67	MWt

as follows:

$$\bar{p} = k_p * p + ka_p \tag{2.5}$$

The relevant parameters identified by Noel Belcourt and the COBRA-TF code team along with brief descriptions taken from [39] are summarized in Table 2.6. The entries containing "(??)" in their description were not documented in [39] but instead were inferred from the COBRA-TF source code.

2.4 CIPS — Crud Induced Power Shift

This section summarizes the CASL CIPS problem to which the UQ approach described in detail in Chapter 7 is extended in Chapter 8. CIPS is a coupled multi-physics phenomenon in which impurities (crud) present in the coolant deposit on the fuel pin cladding and absorb boron. The presence of boron locally reduces moderation which suppresses power and shifts the power profile accordingly. This phenomenon is modelled in CASL by treating the coupled effects of thermal hydraulics, neutronics and crud chemistry using the VERA codes COBRA-TF, MPACT and MAMBA1D, respectively. The interaction among the codes including data exchanged is shown in Figure 2.4.

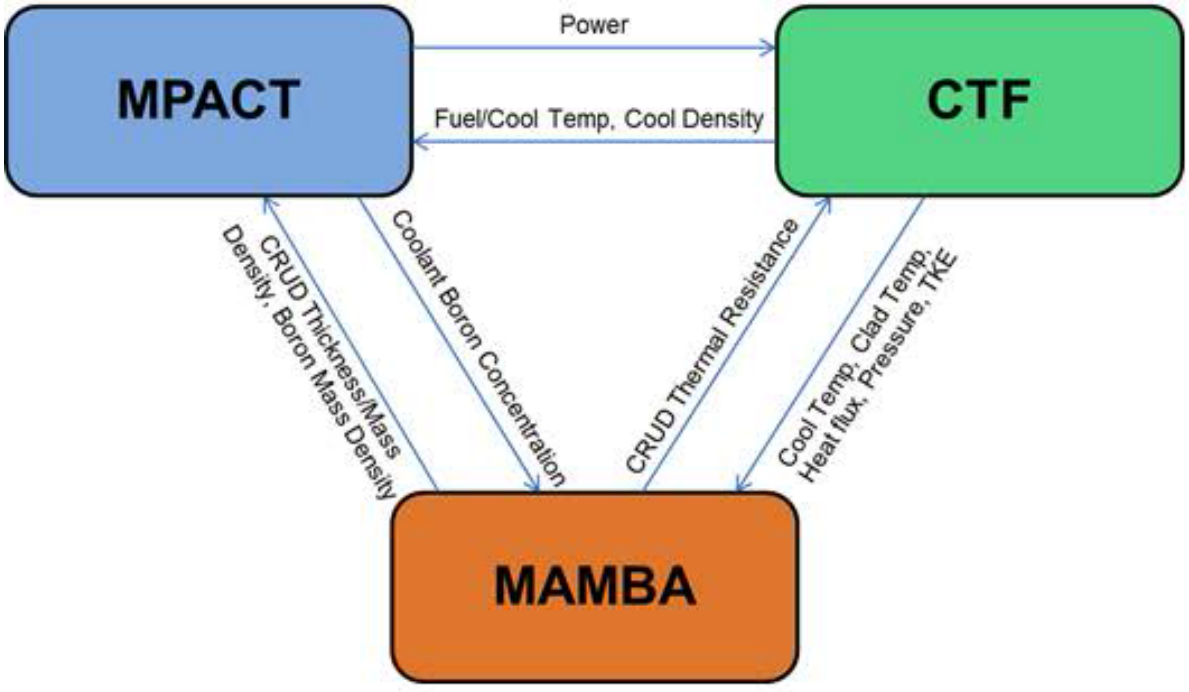


Figure 2.4: Coupled codes to enable CASL CIPS simulations.

Table 2.6: Relevant COBRA-TF thermal-hydraulic code parameters identified by PIRT study.

cd	Pressure loss coefficient of spacer in sub-channel
cdfb	Pressure loss coefficient for sub-channel flow blockage (??)
cond	Thermal conductivity of radial heat transfer
eta	Fraction of vapor generation rate coming from the entrained liquid field
gama	New time vapor generation rate in sub-channel
ql*	Heat transfer rate to liquid in sub-channel
qliht	Heat transfer due to drop impact (??)
qradd	Radiative heat transfer rate from wall to entrained liquid
qradv	Radiative heat transfer rate from wall to vapor
qv*	Heat transfer rate to liquid in sub-channel
qvapl	Incremental heat transferred from grid to vapor (??)
rodqq	Externally supplied heat rate of current rod at current time step (axially averaged)
sdent	Deposition mass flow rate in sub-channel
sent	Entrainment mass flow rate in sub-channel
spht s	Specific heat of radial heat transfer
tmasg	Loss of mass of non-condensable gas in local axial fluid continuity cell due to mixing and void drift to radially adjacent fluid cells
tmasl	Loss of mass of continuous liquid in local axial fluid continuity cell due to mixing and void drift to radially adjacent fluid cells
tmasv	Loss of mass of vapor in local axial fluid continuity cell due to mixing and void drift to radially adjacent fluid cells
tmome	Loss of momentum of droplets in sub-channel due to mixing and void drift to radially adjacent fluid cells
tmoml	Loss of momentum of continuous liquid in sub-channel due to mixing and void drift to radially adjacent fluid cells
tmomv	Loss of momentum of vapor in sub-channel due to mixing and void drift to radially adjacent fluid cells
tnrgl	Loss of enthalpy of liquid in local axial fluid continuity due to mixing and void drift to radially adjacent fluid cells
tnrgv	Loss of enthalpy of vapor in local axial fluid continuity due to mixing and void drift to radially adjacent fluid cells
wkr	Lateral gap pressure loss coefficient
xk	Vertical interfacial drag coefficient between the continuous liquid and vapor phases
xkes	Sink interfacial drag coefficient between the liquid and vapor phases
xkge	Vertical interfacial drag coefficient between the entrained liquid and vapor phases
xkl	Transverse interfacial drag coefficient between the continuous liquid and vapor phases
xkle	Transverse interfacial drag coefficient between the entrained liquid and vapor phases
xkvls	Sink interfacial drag coefficient between the continuous liquid and vapor phases
xkwew	Transverse entrained liquid form loss coefficient
xkwlw	Transverse liquid wall drag coefficient
xkwlx	Vertical liquid wall drag coefficient
xkwvw	Transverse vapor wall drag coefficient
xkwvx	Vertical vapor wall drag coefficient

A PIRT study for CIPS was conducted to identify all potentially influential parameters affecting the two quantities of interest. For parameters available through normal input, e. g. via the VERA Common Input, the following list summarizes expert opinion for the CIPS problem:

- Temperature uncertainty ± 5 F, normal distribution
- Pressure uncertainty ± 50 psi, normal distribution (there is also a 20 psi bias not included so the total uncertainty is ± 70 psi instead)
- Coolant chemistry (to be provided later for MAMBA1D, consistent with the current input to BOA calc)
- Core Average Power uncertainty is ± 0.6%, normal distribution
- Single Assembly Power Uncertainty is ± 4%, normal distribution
- Boron uncertainty; this is a measured value so ± 5 ppm should be acceptable, assumed uniform distribution
- Core Average Flow uncertainty is 2%, normal distribution
- Fuel average density uncertainty is typically within ± 0.5% of target (for example target 95% TD, final 95.5%), assumed normal distribution
- Fuel region average enrichment uncertainty is typically within ± 0.05 wt% of target (example target 3% U235, final 3.05%), assumed normal distribution
- Local Heat Flux uncertainty is ± 3% uncertainty to account for manufacturing uncertainties, normal distribution

This information is incorporated into the VUQ workflow by exposing the corresponding input parameters shown Table 2.7 to Dakota. The table includes the unit conversions needed in moving from the native VERA Input file (`.inp`) to the intermediate XML format (`*.xml`) where Dakota parameter manipulations are performed.

The notation in the fourth column represents the following:

- The '+' symbol prefix for entries under `f8.xml` indicates an additive perturbation.
- The '*' symbol prefix indicates a scaling perturbation.
- The use of an index, [#], in the enrichments perturbations indicates perturbations applied only to the first value of the enrichment array.

It should be noted that not all of the parameters identified as potentially relevant in the PIRT have been included. These are Coolant Chemistry, Boron and Local Heat Flux. Perturbations to Boron require augmenting the current tools used in VUQ. There is currently no parameter for Local Heat Flux in VERA Input though its effect is related to the resulting power distribution. The Coolant Chemistry sensitivity is addressed by the MAMBA1D parameters described next.

The MAMBA1D Coolant Chemistry parameters are treated in the same manner as the COBRA-TF closure parameters, i.e. an auxiliary file is used to impose perturbations via (2.5). The current set of parameters is summarized in Table 2.8. Note that baseline values are not reported in order to prevent disclosure of potentially proprietary information. At the time of this study, the MAMBA1D Theory Manual had not been made available, but it has since been disseminated and can be consulted for more detailed information on the parameters [24].

Table 2.7: Important VERA Common Input CIPS parameters with nominal values.

Parameter	`f8.inp`	Value	`f8.xml`	Value
Temperature	tinlet	556.4 ± 5 F	+STATES/State_1/ tinlet	291.33 ± 2.78 C
Pressure	pressure	2250 ± 70 psia	+STATES/State_1/ pressure	15.513 ± 0.483 MPa
Power	rated (first value)	17.922 ± 4% MW	*CORE/rated_power	17.922 ± 4% MW
Flow	rated (second value)	0.7047 ± 2% Mlbs/hr	*CORE/rated_flow	88.79 ± 2% kg/s
Fuel avg. den.	Fuel U43 (second value)	(95.585 ± 0.5)%	*ASSEMBLIES/ Assembly_B9B-128I/ Fuels/Fuel_U43/thden	(95.585 ± 0.5)%
Fuel avg. den.	Fuel BLK (second value)	(95.459 ± 0.5)%	*ASSEMBLIES/ Assembly_B9B-128I/ Fuels/Fuel_BLK/thden	(95.459 ± 0.5)%
Fuel avg. enrich.	fuel U43	3.1 u-234=0.0395 u-236=0.0044	+ASSEMBLIES/ Assembly_B9B-128I/ Fuels/Fuel_BLK/ enrichments[1]	(3.1 ± 0.05) %
Fuel avg. enrich.	fuel BLK	2.7 u-234=0.029 u-236=0.0036	+ASSEMBLIES/ Assembly_B9B-128I/ Fuels/Fuel_BLK/ enrichments[1]	(2.7 ± 0.05) %

Table 2.8: Relevant MAMBA1D parameters (without nominal values).

Bfract	Threshold for fracton of CRUD that is precipitate
Bthresh	Boron mass precipitation threshold
Cpor	Crud porosity (unitless)
crud_solid	Nominal solid crud density (g/cm^3)
Dc	Effective B Diffusion coefficient (cm^2/s)
delta_r	Nominal minimum length scale (cm)
fac	Scaling factor (parameterized value) unitless
Hc	Chimney heat transfer coefficient W/(cm^2 K)
Hfg	Heat of water vaporization (J/g)
kp2	Subcooled nucleate boiling enhancement to deposition rate
MB	atomic mass of boron (g/mole)
MB10	
MB11	
MFe	
mit0	First of 5 constants in a MIT correlation for coolant density in CRUD
mitMax	maximum temperature for which the correlation is valid
MLi	
MNi	
Nc	Chminey density #/cm^2
rc	Chimney radius in cm
RtcB	Boiling value (calibrated value)
RtcB2	Intermediate Boiling values (calibrated value)
RtcB3	Intermediate Boiling values (calibrated value)
RtcB4	Intermediate Boiling values (calibrated value)
RtcB5	Intermediate Boiling values (calibrated value)
RtcB6	Intermediate Boiling values (calibrated value)
RtcNB	Non-boiling Crud thermal resistance (K*m^2/W)
Tsat	Saturation temperature in deg C

Chapter 3

Sensitivity Analysis

Broadly, the primary goal of sensitivity analysis is to determine which input parameters most influence computational model responses, or deterministic quantities of interest. A ranked list of parameter influences can focus resources for data gathering or model/code development, or can make calibration, optimization, or uncertainty quantification more tractable over a reduced set of parameters. In a post-optimization role, sensitivity information is useful for determining whether or not the response functions are robust with respect to small changes in the optimum design point. The Dakota sensitivity analysis studies, recommended in this chapter, have important secondary benefits as well: (1) they can help identify key model characteristics such as smoothness, nonlinear trends, and robustness to enable selection of suitable Dakota methods for follow-on studies; and (2) some yield sampling designs that can be used to construct the surrogate models described in Chapter 4 for subsequent analyses.

In the CASL context, a phenomena identification and ranking table (PIRT) might help identify the superset of parameters to consider in a sensitivity analysis study. Then the relative parameter rankings resulting from a Dakota-driven sensitivity study form the basis of a quantitative PIRT, or QPRT. These results could also help prioritize model development or data gathering, or identify insensitive parameters to omit from calibration or UQ studies.

3.1 Terminology

This section introduces key sensitivity analysis terminology and defines the metrics typically used to assign relative ranks to parameter influences on a response.

3.1.1 Local Versus Global Sensitivity

Dakota primarily focuses on sensitivity analysis in a global sense; i.e., over the whole valid parameter domain. We contrast that here with more traditional local or partial derivative-based sensitivity analysis.

Local Sensitivity: In some instances, the term sensitivity analysis is used in a local sense to denote the computation of response derivatives with respect to parameters at a nominal value. These local derivatives can then be used to make design decisions or rank parameter influences. Dakota supports this type of study through numerical finite-differences or retrieval of analytic gradients computed within the analysis code. The desired gradient data is specified in the responses section of the Dakota input file and the collection of this data at a single point is accomplished through a parameter study method with no steps.

This approach to sensitivity analysis should be distinguished from the activity of augmenting analysis codes to internally compute derivatives using techniques such as direct or adjoint differentiation, automatic differentiation (e.g., ADIFOR), or complex step modifications. These sensitivity augmentation activities are completely separate from Dakota and are outside the scope of this manual. However, once completed, Dakota can utilize these analytic gradients to perform optimization, uncertainty quantification, and related studies more reliably and efficiently. In CASL, some simulation codes, such as TSUNAMI, have adjoint capabilities and can return not only function values, but also derivative data to Dakota, thus enhancing analyses.

Global Sensitivity: In other instances, the term sensitivity analysis is used in a more global sense to denote the investigation of variability in the response functions over the whole valid range of the input parameters. Dakota supports this type of study through computation of response data sets at a series of sample design points in the parameter space. The series of points is typically defined using a parameter study or a design and analysis of computer experiments (DACE) design, such as orthogonal arrays or space filling Monte Carlo sampling. These more global approaches to sensitivity analysis can be used to obtain trend data even in situations when gradients are unavailable, unreliable, or not indicative of global trends.

This chapter offers guidance solely on Dakota's global sensitivity analysis procedures. Using them typically consists of:

1. Specifying ranges for each parameter and a sensitivity analysis method in the Dakota input
2. Running Dakota which will:
   (a) construct a sampling design in the parameter hypercube;
   (b) run the computational model at these points, collecting returned response data; and
   (c) calculate and output sensitivity metrics to rank inputs.
3. Post-processing the Dakota-generated parameter/response table with external statistics and visualization tools to further assess trends and which input factors most strongly influence the responses

3.1.2 Sensitivity Metrics

Sensitivity metrics output by Dakota are used to assess the relative influence of or rank parameters. The metrics output by Dakota can vary in the manner discussed in Section 3.2, but may include the following.

- **Correlation coefficients:** Dakota prints correlation tables with the simple (Pearson), partial, and rank (Spearman) correlations between inputs and outputs. These are all bounded between -1 and 1 and measure the strength of the linear relationship between the considered variables. These can be useful to get a quick sense of how correlated the inputs are to each other, and how correlated various outputs are to inputs, but can be misleading for detecting nonlinear relationships. For example a model with a perfectly quadratic input/output relationship centered at zero would have zero correlation thus obscuring the actual strong nonlinear relationship.

  The simple correlations are Pearson's correlation coefficient, which is defined for two factors w and x (where each of these could represent an input or an output) as

$$\texttt{Corr}(w,x) = \frac{\sum_i (w_i - \bar{w})(x_i - \bar{x})}{\sqrt{\sum_i (w_i - \bar{w})^2 \sum_i (x_i - \bar{x})^2}}.$$

Partial correlation coefficients are similar, but measure correlation while adjusting for the effects of other variables. For example, in a problem with two inputs and one output, where the two inputs are highly correlated, the correlation of the second input and the output may be very low after accounting for the effect of the first input. The rank correlations in Dakota are obtained using Spearman's rank correlation. Spearman's rank is the same as the Pearson correlation coefficient except that it is calculated on the rank data. Rank correlation can be more informative when responses vary over orders of magnitude. The correlation analyses are explained further in the Uncertainty Quantification chapter of the Dakota User's Manual. [1]

- **Morris metrics** [32] are computed from "elementary effects" based on a sample design of large steps around the parameter space. Here each dimension of a $M-$dimensional input space is uniformly partitioned into p levels, creating a grid of p^M points $x \in \Re^M$ at which evaluations of the model $y(x)$ might take place. An elementary effect corresponding to input i is computed by a forward difference

$$d_i(x) = \frac{y(x + \Delta e_i) - y(x)}{\Delta}, \tag{3.1}$$

where e_i is the i^{th} coordinate vector, and the step Δ is typically taken to be large (this is not intended to be a local derivative approximation); e.g., for an input variable scaled to $[0, 1]$, $\Delta = \frac{p}{2(p-1)}$, so the step is slightly larger than half the input range. We note that users may obtain improved results for some problems with smaller stepsizes.

The distribution of elementary effects d_i over the input space characterizes the effect of input i on the output of interest. After generating N samples from this distribution, their mean,

$$\mu_i = \frac{1}{N} \sum_{j=1}^{N} d_i^{(j)}, \tag{3.2}$$

modified mean

$$\mu_i^* = \frac{1}{N} \sum_{j=1}^{N} |d_i^{(j)}|, \tag{3.3}$$

(using absolute value) and standard deviation

$$\sigma_i = \sqrt{\frac{1}{N-1} \sum_{j=1}^{N} \left(d_i^{(j)} - \mu_i \right)^2} \tag{3.4}$$

are computed for each input i. The mean and modified mean give an indication of the overall effect of an input on the output. Standard deviation indicates nonlinear effects or interactions, since it is an indicator of elementary effects varying throughout the input space.

- **Sobol indices:** Dakota can compute sensitivity indices via Variance-based Decompositions (VBD). A variance-based decomposition is a global sensitivity method that quantifies how the uncertainty in model output can be apportioned to uncertainty in individual input variables. VBD uses two primary measures, the main effect sensitivity index S_i and the total effect sensitivity index T_i. The main effect sensitivity index corresponds to the fraction of the total uncertainty in the output, Y, that can be attributed to input x_i alone. The total effect sensitivity index corresponds to the fraction of the total uncertainty in the output, Y, that can be attributed to input x_i and its interactions with other variables. The main effect sensitivity

index compares the variance of the conditional expectation $Var_{X_i}[E(Y|X_i)]$ against the total variance $Var(Y)$.

Formulas for the indices are:

$$S_i = \frac{Var_{X_i}[E(Y|X_i)]}{Var(Y)} \tag{3.5}$$

and

$$T_i = \frac{E_{X_{-i}}[Var(Y|X_{-i})]}{Var(Y)} = \frac{Var(Y) - Var_{X_{-i}}[E(Y|X_{-i})]}{Var(Y)} \tag{3.6}$$

where $Y = f(\mathbf{x})$ and $x_{-i} = (x_1, ..., x_{i-1}, x_{i+1}, ..., x_M)$. The calculation of S_i and T_i requires the approximation of M-dimensional integrals which are typically approximated by Monte-Carlo sampling.

When using VBD, a rough guide is that variables with main effect indices greater than $100/M\%$ are significant as they can be considered to have greater than average effect on output variability, barring higher-order interactions. More details on the calculations and interpretation of the sensitivity indices can be found in [40].

- **Main effects** quantify the effects of a single variable, averaging across the effect of other input variables. For a full factorial design, with each of M inputs taking on p levels, the main effect of input variable x_i is calculated at each level $k = 1, ..., p$ it takes on as

$$m_i^k = \frac{1}{p^{M-1}} \sum_x y(x|x_i = x_i^k).$$

  To calculate main effects with Dakota, one can either use (1) the orthogonal array method from DDACE with the supplementary command `main_effects`, or (2) a grid parameter study, which has to be post-processed to compute main effects in an external statistics tool.

Supplementary Approaches: Running any of the parameter study, design of experiments, or sampling methods allows the user to save the results in a tabular data file, which then can be read into a spreadsheet or statistical package for further analysis. One example of this is the well-known technique of scatter plots, in which the set of samples is projected down and plotted against one parameter dimension, for each parameter in turn. Scatter plots with a uniformly distributed cloud of points indicate parameters with little influence on the results, whereas scatter plots with a defined shape to the cloud indicate parameters which are more significant. Related techniques include analysis of variance (ANOVA) [34] and main effects analysis, in which parameters having the greatest influence on the output are identified from sampling results. Scatter plots and ANOVA may be accessed through import of Dakota tabular results into external statistical analysis programs such as R (`http://www.r-project.org`) and Minitab (`http://www.minitab.com`).

3.2 Recommended Methods

This section summarizes a few recommended Dakota sensitivity analysis methods at a high level and provides input file examples, resulting output, and post-processing/visualization approaches that can help. The choice of method will depend on the analysis goal and available computational budget. We begin with high-level best practices before delving into examples of specific methods.

We almost always recommend starting with simple **centered parameter studies** that yield univariate effects only. Do this first with small perturbations, then large variations that span the parameter space. These simple studies test the model interface, assess the relative smoothness of

the response, and assess model robustness over single parameter variations. These studies readily determine the effect of a single parameter in a practical way, as they are automated versions of typical "perturb ±5%, ±10%" studies that analysts manually conduct.

The type of follow-on sensitivity study to conduct depends on simulation budget and goal. The cost for various methods is shown in Table 3.1, together with the key metrics that they yield. Here M is the number of input parameters studied, p a user-specified number of increments or partitions in each variable (often taken to be $p = 3$), N a total number of samples in a single Latin hypercube sampling (LHS) replicate, and k a number of replicates (often $k = 4$) which may be needed to evaluate the formulas for Sobol indices from Section 3.1.2.

Table 3.1: Key sensitivity analysis methods with the metrics they produce, roughly ordered by increasing computational cost. M: number of parameters, p: increments per variables, N: total samples in a single replicate, k: number of replicates.

method	**design points**	**metrics**
centered parameter study (Sec. 3.2.1)	$p \times M + 1$	univariate effects
global LHS sampling (Sec. 3.2.3)	$N = 2 \times M$ to $10 \times M$	Pearson, partial, Spearman correlations
PSUADE/Morris (Sec. 3.2.4)	$k \times (M + 1)$, with p odd	elementary effects
VBD/Sobol (none)	$N \times (M + 2)$	Sobol main/total effects
full factorial/grid (Sec. 3.2.2)	p^M	correlations, main effects

A **global LHS sampling** study is the most common follow-on study, ideally with $N = 10 \times M$, but possibly as few as $N = 2 \times M$, samples. Global LHS sampling has the benefit of reuse of the sample points for follow-on surrogate construction. Dakota directly outputs correlation coefficients, which when large can indicate parameters surely influencing the response; useful for inclusion-based screening. It also yields data for constructing scatter plots in post-processing analysis. Parameter effects can be confounded, so it can be hard to extract univariate effects, but one can get a good idea of the effect of joint variation with modest samples.

If point reuse or scatter plot diagnostics are not a primary consideration, **PSUADE/Morris** designs offer more inference power, with a similar cost, to LHS designs by facilitating quantitative detection of nonlinear or interaction effects in addition to main effects.

Variance-based decomposition (VBD/Sobol) analysis can offer even more information. This uses replicate LHS or a surrogate (possibly stochastic polynomial approximations as described in Section 6.1.2) to perform a variance-based decomposition that apportions output variance to input factors. The resulting Sobol indices for main and total effects (described in Section 3.1.2) can be helpful for up front sensitivity analysis, when an "80/20" principle applies: fewer than 20 percent of the parameters explain at least 80 percent of total output variance. In this case, one can quickly screen with relatively few model runs. However, this approach is challenging in the presence of strong interactions, due to the potentially substantial data requirements for accurate inference of Sobol indices. Strong interactions are suggested by differences in partial versus simple correlations or main versus total effects in Sobol indices.

For modest numbers of parameters and reasonable model run cost, one can conduct **full factorial** parameter studies with Dakota's grid/multidimensional parameter study or fractional factorial orthogonal array (OA) designs with DACE OA. These can assess main effects of each parameter, even when considering them jointly. A two-level Plackett-Burman design can be good for extremely slow codes, where the number of runs ($N = M + 1$) is severely limited. However these methods are not currently available in Dakota.

3.2.1 Centered Parameter Study

The centered parameter study executes multiple coordinate-based parameter studies, one per parameter, centered about the specified initial values. This is useful for investigation of function contours in the vicinity of a specific point and is a very common model exploration technique where each parameter is increased and decreased by a fixed increment. A set of widely-spaced points in a centered or multidimensional parameter study could be used to determine whether the response function variation is likely to be unimodal or multimodal. A set of closely-spaced points in a centered parameter study could also be used to assess the smoothness of the response functions in order to select a suitable finite difference step size for optimization/calibration (see an example of their use in Listing 5.1 in Section 5.2.1). After computing an optimum design, a parameter study could also be used for post-optimality analysis in verifying that the computed solution is actually at a minimum or constraint boundary and in investigating the shape of this minimum or constraint boundary. (In a parameter study, one may optionally enable Dakota's numerical gradient estimation to calculate local derivative values at each point in the parameter space, but the results are only published to the Dakota console text output, not the tabular data file.)

Dakota Input: This method requires two settings: (1) `step_vector`, a list of real values, each of which specifies the size of the increment or perturbation for a single variable; and (2) `steps_per_variable`, a list of integers that specifies the number of increments p_i per variable in each of the positive and negative coordinate directions. Centered parameter studies are typically conducted with $p_i = 5$ positive and negative increments of each parameter. The total number of samples required is $N = 1 + \sum_{i=1}^{M} 2p_i$. `step_vector` specifies absolute variable steps for continuous and discrete range variables, but for studies conducted over integer or real discrete set variables (see "Design Variables" in the Dakota Reference Manual [2]), specifies perturbations in index offsets to select from the possible set values. For example, with initial values of (1.0, 1.0), a `step_vector` of (0.1, 0.1), and a `steps_per_variable` of (2, 2), the center point is evaluated followed by four function evaluations (two negative deltas and two positive deltas) per variable. This set of points in parameter space is shown in Dakota screen output in Figure 3.1 and graphically in Figure 3.2.

Dakota Input for Cantilever: A sample Dakota input file for a centered parameter study with the cantilever beam application is shown in Listing 3.1. Note the previously discussed method controls for `centered_parameter_study` in lines 9–11, that all the variables are active (line 18), and that the evaluations will be saved to the tabular data file specified on line 3. The centered study is fully characterized by the initial values and steps for the variables (lines 21 and 25, 10 and 11).

Results and Discussion: The results of the study are depicted in Figure 3.3, where the tabular data generated by Dakota (`cantilever_centered.dat`) has been plotted with Matlab. These plots show that only w and t affect area /weight (the plots for these two variables overlay each other). For the stress and displacement, w and t have the strongest effect, and possibly a nonlinear one as evidence by the curvature in their traces. E and X have a small, but nonzero effect. All input/output relationships appear smooth (no noise or other oscillation is evident). These observations can be verified by studying the equations for the static cantilever problem.

3.2.2 Multidimensional Parameter Study

The multidimensional parameter study computes response data sets for an M-dimensional hypergrid of input points. This full factorial design is powerful in determining main effects and potential interactions among parameters, but the number of simulation runs quickly becomes prohibitive as the dimension M of the parameter space increases. It is presented here mainly because it's easily understandable and tractable/useful for small dimensional problems.

Listing 3.1: Dakota input file showing centered parameter study on the cantilever beam problem.

```
environment
  tabular_data
    tabular_data_file 'cantilever_centered.dat'
    custom_annotated header eval_id

method

# do a parameter study in coordinate directions over all 6 parameters
  centered_parameter_study
    step_vector 0.1 0.1 10 100 10 100
    steps_per_variable 2

variables

# by default, a parameter study won't operate on state parameters
# can change that default behavior be explicitly specifying which
# parameters to use (here "all")
  active all

  continuous_design = 2
    initial_point  1.0      1.0
    descriptors    'w'      't'

  continuous_state = 4
    initial_state  40000. 29.E+6 500. 1000.
    descriptors      'R'   'E'   'X'   'Y'

interface
  direct
    analysis_driver = 'mod_cantilever'

responses
  num_objective_functions = 3
    response_descriptors = 'area' 'stress' 'displacement'
  no_gradients
  no_hessians
```

```
Parameters for function evaluation 1:
                     1.0000000000e+00 d1
                     1.0000000000e+00 d2
Parameters for function evaluation 2:
                     8.0000000000e-01 d1
                     1.0000000000e+00 d2
Parameters for function evaluation 3:
                     9.0000000000e-01 d1
                     1.0000000000e+00 d2
Parameters for function evaluation 4:
                     1.1000000000e+00 d1
                     1.0000000000e+00 d2
Parameters for function evaluation 5:
                     1.2000000000e+00 d1
                     1.0000000000e+00 d2
Parameters for function evaluation 6:
                     1.0000000000e+00 d1
                     8.0000000000e-01 d2
Parameters for function evaluation 7:
                     1.0000000000e+00 d1
                     9.0000000000e-01 d2
Parameters for function evaluation 8:
                     1.0000000000e+00 d1
                     1.1000000000e+00 d2
Parameters for function evaluation 9:
                     1.0000000000e+00 d1
                     1.2000000000e+00 d2
```

Figure 3.1: Dakota output showing function evaluations for a centered parameter study with two positive and two negative steps per variable.

For these studies, each variable is partitioned into equally spaced intervals between its upper and lower bounds, and each combination of the values defined by these partitions is evaluated. The

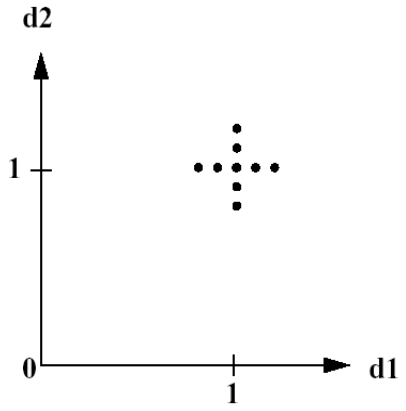


Figure 3.2: Notional example of centered parameter study over two parameters $d1$ and $d2$.

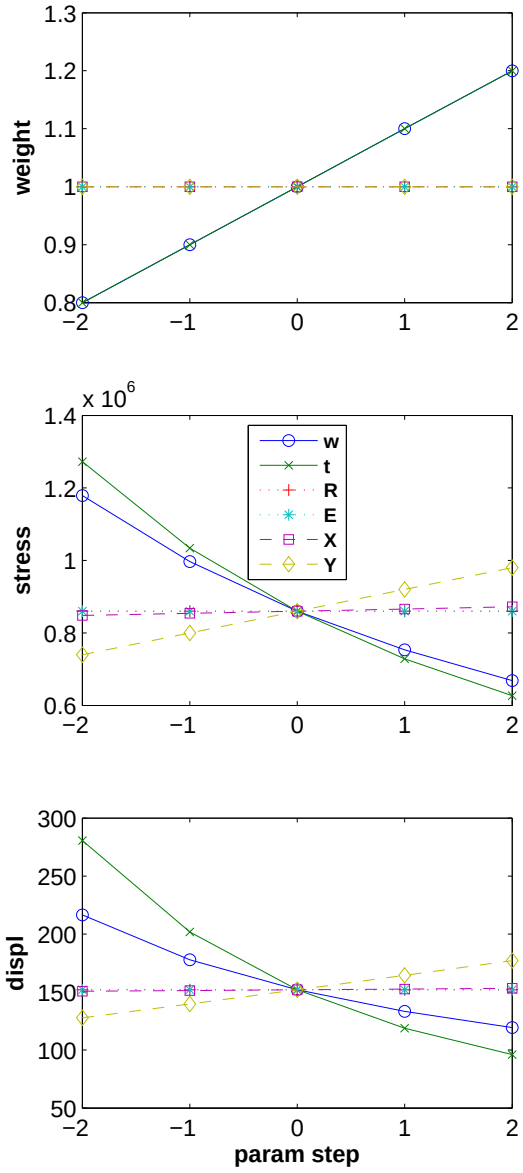


Figure 3.3: Cantilever beam: univariate effects from centered parameter study of each of six parameters on each of three responses.

number of function evaluations performed in the study is:

$$\prod_{i=1}^{M}(\texttt{partitions}_i + 1) \tag{3.7}$$

Dakota Input Example: The partitions information is provided using the `partitions` specification, which inputs an integer list of the number of partitions for each variable (i.e., $\texttt{partitions}_i$). Since the initial values will not be used, they need not be specified.

In a two variable example problem with `d1` $\in$ [0,2] and `d2` $\in$ [0,3] (as defined by the upper and lower bounds from the variables specification) and with `partitions` = (2,3), the interval [0,2] is divided into two equal-sized partitions and the interval [0,3] is divided into three equal-sized partitions. This two-dimensional grid, shown notionally in Figure 3.4, would result in the twelve function evaluations shown in Figure 3.5. See the first example in the Dakota User's Manual [1]: Tutorial for additional notes to understand this study.

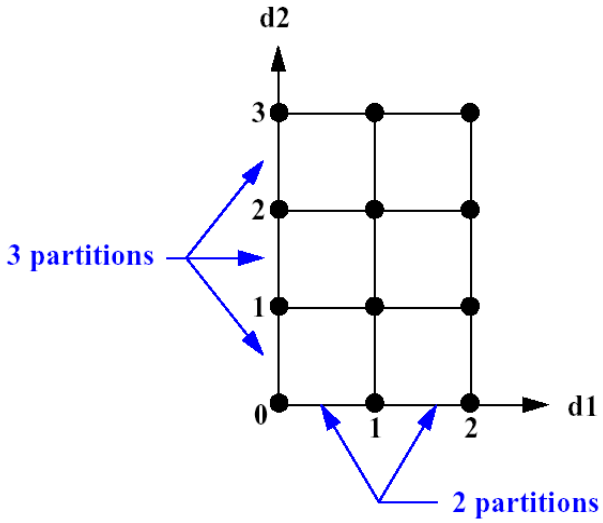


Figure 3.4: Example of multidimensional parameter study.

Dakota Input for Cantilever: Listing 3.2 shows a Dakota input file prescribing a multidimensional parameter study for the cantilever beam problem. On line 10, 9 partitions are specified for w and 6 for t, resulting in $10 \times 7 = 70$ total model evaluations in the (w, t) space. The parameters R, E, X, and Y are held at nominal values using Dakota's state variable mechanism, as `active all` is commented on line 16. In contrast to the centered parameter study, the active variables are characterized by their lower and upper bounds (lines 19–20).

Results and Discussion: When resources allow a grid parameter study to be conducted, one resulting advantage is that main effects can be calculated. An example is shown in Figure 3.6. In this example, the main effects for w and t are generated by post-processing Dakota's `cantilever_grid.dat` file using external statistical and plotting software. The left subplot shows the main effect of w, that is the relationship between w and the mean of the displacement, taken over all realization of the other variable t. We observe a smooth, nonlinear effect. Similar is true for the main effect of t in the right subplot.

The plot shown here was generated using Minitab statistical software, but SAS (`http://www.sas.com`), JMP (`www.jmp.com`), R (`http://www.r-project.org`), Minitab (`http://www.minitab.com`), Matlab (`http://www.mathworks.com/products/matlab`), and other tools can produce similar graphics. Therefore we recommend CASL analysts use the tools present in the computing environment at their institution. Grid parameter studies also output correlation coefficients, discussed at greater length in the next section.

3.2.3 Global LHS Sampling

Monte Carlo sampling methods, including Latin hypercube sampling (LHS), are discussed in more detail in Chapter 6, Uncertainty Quantification. For sensitivity analysis with global sampling,

Listing 3.2: Dakota input file showing grid parameter study on the cantilever beam problem.

```
environment
  tabular_data
    tabular_data_file 'cantilever_grid.dat'
    custom_annotated header eval_id

method

# conduct grid parameter study with 10 values for width, 7 for thickness
  multidim_parameter_study
    partitions = 9 6

variables

# default is to perform the study over the design variables, leaving
# state fixed; could override to do all variables with:
##active all

  continuous_design = 2
    lower_bounds   1.0      1.0
    upper_bounds   4.0      4.0
    descriptors    'w'      't'

  continuous_state = 4
    initial_state  40000. 29.E+6 500. 1000.
    descriptors      'R'   'E'   'X'   'Y'

interface
  direct
    analysis_driver = 'mod_cantilever'

responses
  num_objective_functions = 3
    response_descriptors = 'area' 'stress' 'displacement'
  no_gradients
  no_hessians
```

```
Parameters for function evaluation 1:
                      0.0000000000e+00 d1
                      0.0000000000e+00 d2
Parameters for function evaluation 2:
                      1.0000000000e+00 d1
                      0.0000000000e+00 d2
Parameters for function evaluation 3:
                      2.0000000000e+00 d1
                      0.0000000000e+00 d2
Parameters for function evaluation 4:
                      0.0000000000e+00 d1
                      1.0000000000e+00 d2
Parameters for function evaluation 5:
                      1.0000000000e+00 d1
                      1.0000000000e+00 d2
Parameters for function evaluation 6:
                      2.0000000000e+00 d1
                      1.0000000000e+00 d2
Parameters for function evaluation 7:
                      0.0000000000e+00 d1
                      2.0000000000e+00 d2
Parameters for function evaluation 8:
                      1.0000000000e+00 d1
                      2.0000000000e+00 d2
Parameters for function evaluation 9:
                      2.0000000000e+00 d1
                      2.0000000000e+00 d2
Parameters for function evaluation 10:
                      0.0000000000e+00 d1
                      3.0000000000e+00 d2
Parameters for function evaluation 11:
                      1.0000000000e+00 d1
                      3.0000000000e+00 d2
Parameters for function evaluation 12:
                      2.0000000000e+00 d1
                      3.0000000000e+00 d2
```

Figure 3.5: Dakota output function evaluations for a grid parameter study, the tensor product of three steps in $d1$ with four steps in $d2$.

variables are typically taken to be uniform on their support. Dakota will generate a space-filling sample design (shotgun blast of points into the M-dimensional parameter space). It will then run the model at these points, and analyze the resulting response data. Here we use a Latin hypercube design for consistency with the UQ recommendations. Latin hypercube designs have better space filling properties and 1-D projections. They will converge statistically at a faster rate than simple Monte Carlo designs.

Dakota Input for Cantilever: A Dakota input file example for conducting a sampling-based study on the cantilever beam is displayed in Listing 3.3. It generates a Latin hypercube design with 100 sample points (lines 11–12) in the six-dimensional parameter space. Note lines 35–38,

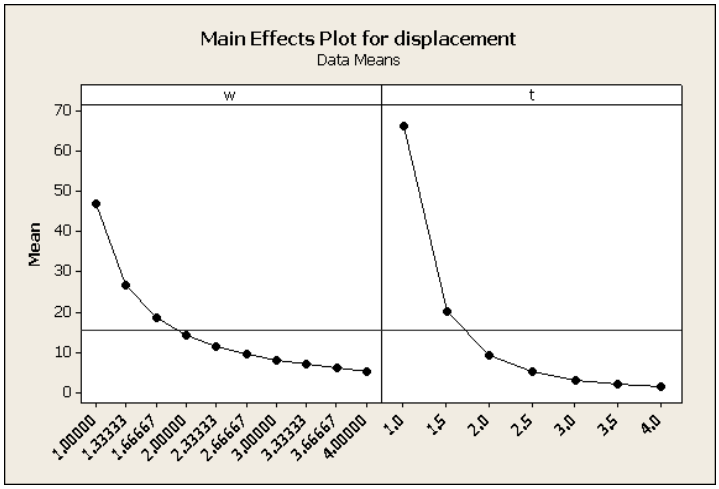


Figure 3.6: Cantilever beam: Minitab-generated main effects for w and t from grid study.

where the variables are characterized using uniform probability distributions. While not strictly necessary (bounded design variables can work too), this illustrates that Monte Carlo sampling can work with arbitrary probability distributions when needed. The seed on line 29 is specified for study repeatability. If the seed is omitted, Dakota will choose one at random, resulting in a different random design. This can be used to generate replicates to assess variability the statistics. Figure 3.7 shows sets of 2-D projections of one Dakota-generated sampling design.

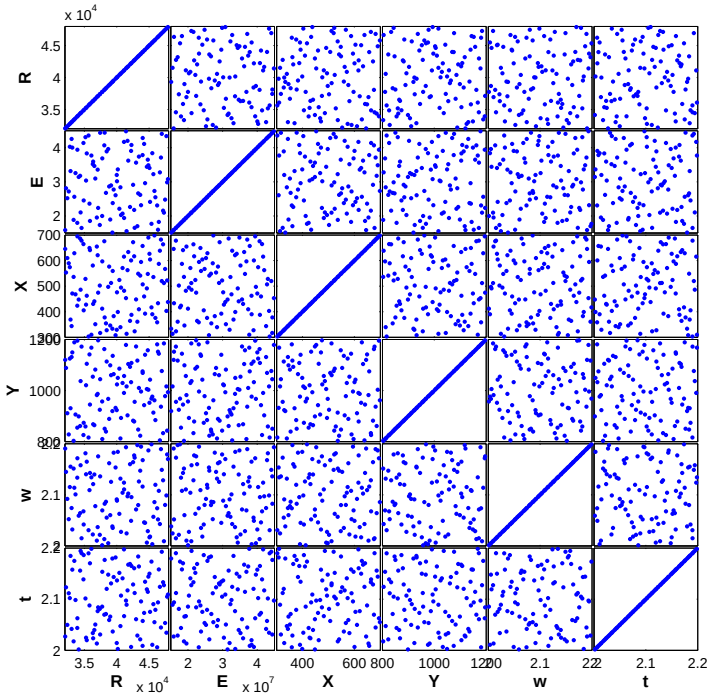


Figure 3.7: Latin hypercube design example: 2-D projections showing locations in the parameter space of 100 Monte Carlo samples using uniform distributions for all six parameters.

Listing 3.3: Dakota input file showing global sampling on the cantilever beam problem.

```
environment
  tabular_data
    tabular_data_file 'cantilever_sa.dat'
    custom_annotated header eval_id

method,

# do a sampling-based sensitivity study, optionally with
# variance-based decomposition to get higher order sensitivities
  sampling
    sample_type lhs
    samples = 100
#    variance_based_decomp

# do a design-of-experiments-based sensitivity using ANOVA to compute
#    the sensitivities
#    dace oas
#      main_effects
#    samples = 100

# do a Morris One-At-A-Time sensitivity study
#    psuade_moat
      # must be odd
#      partitions = 3
      # must be integer multiple of (num_vars + 1)
#      samples = 98

# need these for all methods; seed allows for repeatability
    seed = 52983

variables,

# By default DACE and Sampling methods sample over all variables

  uniform_uncertain = 6
    upper_bounds    48000.  45.E+6  700.  1200.  2.2   2.2
    lower_bounds    32000.  15.E+6  300.  800.   2.0   2.0
    descriptors      'R'     'E'    'X'   'Y'    'w'   't'

interface,
  direct
    analysis_driver = 'mod_cantilever'

responses,
  num_response_functions = 3
  response_descriptors = 'weight' 'stress' 'displ'
  no_gradients
  no_hessians
```

Results and Discussion: When performing Monte Carlo sampling, Dakota outputs correlations, including the partial correlations shown in the screen output in Figure 3.8. As discussed previously in Section 3.1.2, correlations often give a good quick indication of the overall input/output correlation as shown here. Values near 1 indicate strong positive correlation, -1 negative correlation, 0 no correlation. For sensitivity analysis, we typically rely on partial correlations as they account for the effect of other variables to provide a more reliable ranking mechanism.

```
Partial Correlation Matrix between input and output:
                    weight       stress        displ
           R  1.36556e-01 -9.89955e-01 -5.82547e-02
           E -2.59807e-02  1.51530e-02 -9.53598e-01
           X -8.58158e-03  9.96167e-01  3.12725e-01
           Y  5.15226e-02  9.96214e-01  7.35493e-01
           w  9.99659e-01 -9.84197e-01 -4.20681e-01
           t  9.99659e-01 -9.89246e-01 -5.24940e-01
```

Figure 3.8: Dakota output showing partial correlations for the cantilever beam problem.

One can visualize the correlation coefficients with external software to more easily see the relative impact. For large numbers of parameters, it is helpful to plot the relative magnitudes (Figure 3.9) or color code with conditional formatting in Excel (Figure 3.10) to more readily differentiate small from large correlations. Typically, correlations greater than 0.5 in magnitude indicate potentially significant input/output relationships (though specific guidance and interpretation depends on the number of samples, number of variables, and analysis tolerance). Values less than 0.5 should be more carefully studied for possible confounding factors or nonlinearities before discounting their importance.

As correlation coefficients are a linear measure of input/output relationship, it is critical to visualize scatter plots to check for nonlinear trends. Figure 3.11 shows scatter plots generated in Matlab, together with linear regression fits to the data. Some of the scatter plots exhibit nonlinear input/output trends in the cloud of data, for example the plots of E versus displ shows some curva-

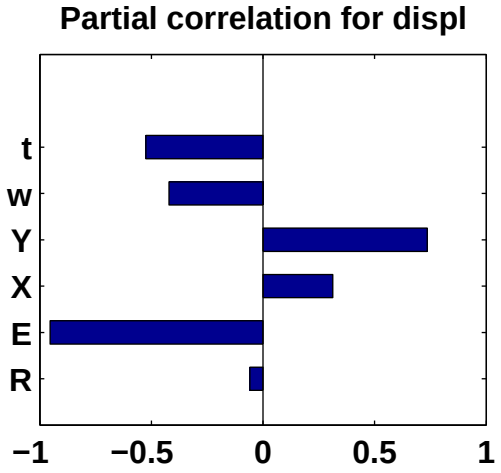


Figure 3.9: Cantilever beam: partial correlation between displacement and inputs.

Partial Correlations for Cantilever

	weight	stress	displ
R	0.14	-0.99	-0.06
E	-0.03	0.02	-0.95
X	-0.01	1.00	0.31
Y	0.05	1.00	0.74
w	1.00	-0.98	-0.42
t	1.00	-0.99	-0.52

Figure 3.10: Cantilever beam: partial correlation between displacement and inputs, conditionally formatted with Microsoft Excel.

ture. Overall, however, the correlations are a good representative indicator of the most important factors for these sample data.

Monte Carlo / LHS sampling for sensitivity analysis screening is typically conducted with number of samples equal to 10 times the number of variables, but budgets often push this down to a factor of two. Also, global Gaussian process models built in Dakota are typically constructed based on a Latin hypercube design. Guidance on the number of samples for constructing such surrogates is provided in Chapter 4.

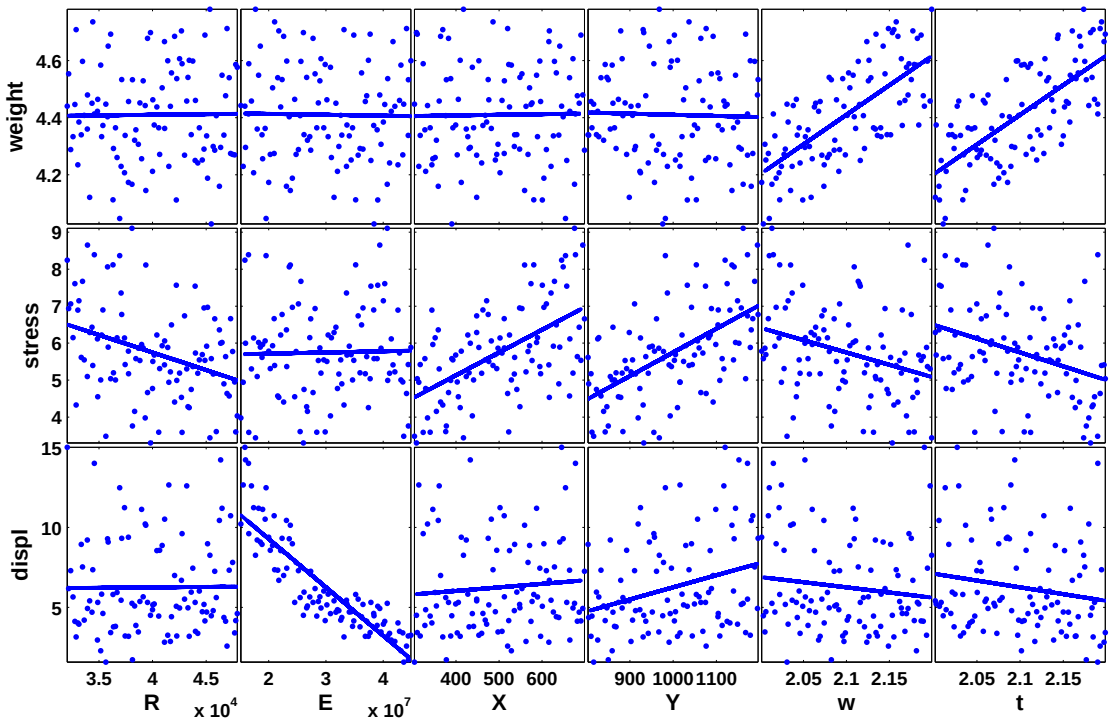


Figure 3.11: Cantilever beam: input/output scatter plots with correlations.

3.2.4 PSUADE/Morris Method

The Morris One-At-a-Time method, originally proposed by M. D. Morris [32], is a screening method, designed to explore a computational model to distinguish between input variables that have negligible, linear and additive, or nonlinear/interaction effects on the output. The computer experiments consist of individually randomized designs, which vary one input factor at a time to create a sample of its elementary effects. A more extensive discussion of the method and its metrics can be found in the "Design of Experiments Capabilities" chapter of the Dakota User's Manual [1].

The file `examples/SensitivityAnalysis/cantilever_morris.in` shows an alternate Dakota method specification for conducting a Morris screening experiment, resulting in the Morris modified mean and standard deviation of elementary effects, defined above in Section 3.1.2. This method often gives good insight for modest simulation budget. The changed input fragment is:

```
psuade_moat
  # must be odd
  partitions = 3
  # must be integer multiple of (num_vars + 1)
  samples = 98
```

In Figure 3.12, the Dakota screen output has been imported into Matlab to plot the modified mean μ^* versus standard deviation σ of the elementary effects. One can observe that for weight, w and t have a strong main effect as expected, and a small nonzero interaction effect. Other variables have no influence. For stress, R has a strong main/linear effect, X and Y have a stronger main effect than w and t, yet w and t have a stronger interaction effect. These are also evident from the cantilever equations. The displacement shows strong main and interaction dependence on E, weaker influence of w, t, X, Y, and no influence of R. This is likely an artifact of the magnitude of E, which is swamping the analysis, and emphasizes the importance of careful input scaling.

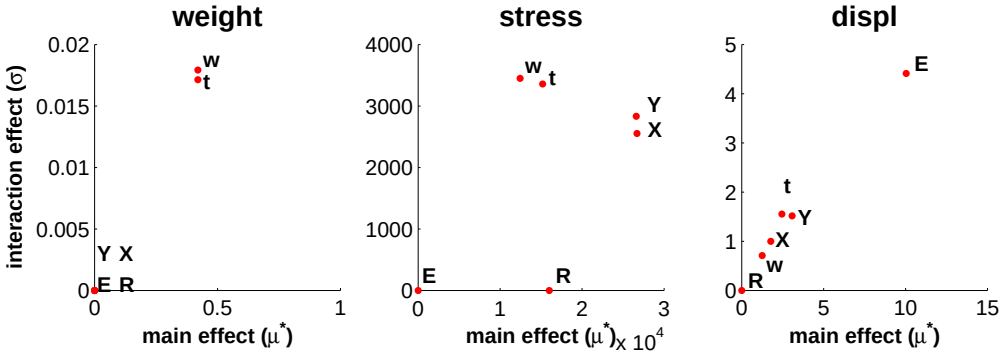


Figure 3.12: Cantilever beam: Morris elementary effects: modified mean (μ^*) and standard deviation (σ).

3.3 Summary and Additional Approaches

Parameter studies, design/analysis of computer experiments (DACE), and general sampling methods share the purpose of exploring the parameter space for sensitivity analysis. When a global space-filling set of samples is desired, then the design of experiments (DOE), DACE, and sampling methods

are recommended, with the particular choice depending on computational cost. These techniques are useful for scatter plot and variance analysis as well as surrogate model construction.

We draw a distinction between DOE and DACE methods. DOE are intended for physical experiments containing an element of stochasticity (and therefore tend to place samples at the extreme parameter vertices), whereas the latter are intended for deterministic computer experiments and are more space-filling in nature. Another distinction between DOE/DACE and sampling is based on the distributions of the parameters. DOE/DACE methods typically assume uniform distributions, whereas the sampling (and other uncertainty quantification) approaches in Dakota support a broad range of probability distributions.

Sensitivity analysis method selection recommendations for a broader array of Dakota methods are summarized in Table 3.2. Here are a few highlights to supplement the core recommendations above:

- The vector and list parameter study methods not described here are summarized in the "Parameter Study Capabilities" chapter of the Dakota User's Manual [1]. List parameter studies run the model at user-specified design points.
- The file `cantilever_sa.in` also demonstrates the use of DACE orthogonal array (lines 17–19) designs with minor changes to the input file. These output other helpful measures of sensitivity described in the "DACE Capabilities" chapter of the Dakota User's Manual [1].
- When implemented using replicate samples (specifying `method`, `sampling`, `variance_based_decomp` in the Dakota input), variance-based decomposition can require a prohibitive number of model runs. Surrogates such as Gaussian process models (see Section 4.2) are often used to mitigate this cost. Another advanced approach is to use polynomial chaos expansions (Section 6.1.2), which also produce Sobol indices. One may use a structured PCE design generated by Dakota, or import the points from a previous Monte Carlo sample to build the PCE and quickly calculate the variance-based decomposition.
- Other design types may be more appropriate for polynomial regression such as Box-Behnken [7] or central composite design [8] for quadratic polynomials.

Table 3.2: Guidelines for selection of parameter study, DOE, DACE, and sampling methods.

Method Classification	**Applications**	**Applicable Methods**
parameter study	sensitivity analysis, directed parameter space investigations	centered_parameter_study, list_parameter_study, multidim_parameter_study, vector_parameter_study
classical design of experiments	physical experiments (parameters uniformly distributed)	dace (box_behnken, central_composite)
design of computer experiments	variance analysis, space filling designs (parameters uniformly distributed)	dace (grid, random, oas, lhs, oa_lhs), fsu_quasi_mc (halton, hammersley), fsu_cvt, psuade_moat
sampling	space filling designs (parameters have general probability distributions)	sampling (Monte Carlo, LHS) with optional active view override

Chapter 4

Surrogate Models

This chapter introduces the basic theory and use of Dakota's polynomial regression and kriging surrogate models. Surrogate models are typically employed to provide computationally efficient approximate representations of trends and residual (error) processes in physical data or code output. The terms "emulator," "response surface," and "meta-model" refer to the generation of surrogate predictions with associated uncertainty quantification. In the Dakota context, surrogate models are automatically generated based on empirical samples of the true simulation model's input/output relationship. This type of surrogate can be contrasted with physics-based surrogates which make simplifying assumptions to create a simpler, faster-running simulation model.

In the following, $Y(x)$ denotes a surrogate for the physical or computational response of interest $f(x)$, $\tilde{\mu}(x)$ denotes the prediction of $Y(x)$ evaluated at the input x (emulator mean), and $\tilde{\sigma}(x)$ denotes the standard error of prediction evaluated at the input x (emulator standard error). Emulators are constructed from physical data or code output collected on a sample design of N runs in M input dimensions. Good practice requires the run size N to be at least as large as the number of surrogate model parameters requiring estimation (degrees of freedom). Dakota enforces this practice by exiting and returning an error message if N is too small. The degrees of freedom (and thus minimum N) will be stated for each surrogate model introduced in the ensuing sections.

The surrogate models considered in this chapter assume the distribution of residuals (difference between experimental or code output and emulator prediction) is modeled as mean-zero Gaussian. Diagnostics such as a normal probability plot of standardized emulator residuals can be used to check this assumption, as illustrated in Section 7.3.2. Under the assumption of Gaussian residuals, outputs used to construct emulators will then follow a multivariate Gaussian distribution, assuming all unknown surrogate model parameters are fixed. The *likelihood* of observing the given outputs for any given value of the surrogate model parameters is the associated value of this multivariate Gaussian density function. In this chapter, most surrogate model parameters are estimated by *maximum likelihood* in Dakota, by finding a value for these parameters that maximizes the likelihood function. This value is referred to as the maximum likelihood estimate (MLE). Parameter estimation via MLE differs from the Bayesian approach to parameter inference discussed in Uncertainty Quantification, Chapter 6, in that the latter results in a probability distribution for the surrogate model parameters. In practice, the contribution to overall emulator uncertainty induced by surrogate model parameter uncertainty from Bayesian inference is often small compared with the uncertainty arising from the residual error process itself. Computations in Dakota involving surrogate model indirection can thus be sped up considerably by utilizing a point estimate (the MLE) in place of a probability distribution for the surrogate model parameters.

Section 4.1 considers polynomial surrogates, which are typically used to model observed trends

in data from physical experiments resulting from perturbing input parameters that describe physical scenarios of interest. Polynomial surrogates model these trends as a regression relationship that is linear in the unknown coefficients (but not necessarily in the inputs themselves). Experimental errors are assumed to be independently distributed as mean-zero Gaussian distributions having common variance. Polynomial surrogates *smooth* the observed data by finding the best-fitting trend model of those in the class specified by the user.

Section 4.2 considers kriging surrogates, which are typically used to model observed trends in outputs from computationally intensive code runs resulting from input parameter perturbations. In this setting, input parameters describe physical scenarios of interest as well as uncertain initial or boundary conditions, or possibly uncertain closure model parameters that are calibrated to experimental data (see Optimization and Deterministic Calibration, Chapter 5 and Uncertainty Quantification, Chapter 6). Assuming the quantities of interest vary smoothly with input perturbations, the information available from a small number of code runs can be used effectively by the kriging surrogate to infer a correlated error structure. The kriging surrogate is thus able to borrow strength from the small set of code runs conducted to quickly predict code output at user specified input settings with quantified uncertainty. Code runs "closer" to a desired prediction site are weighted more heavily in constructing the kriging emulator than those further away with respect to the inferred correlation structure.

Typically, kriging surrogates *interpolate* the set of code runs made for the purpose of inferring the surrogate model parameters. That is, emulator predictions at input sites corresponding to available code runs are equal to the calculated code outputs with zero uncertainty. This interpolation property is often desirable for emulating deterministic codes that produce a single output value over repeated runs at the same input point. However, as described in Section 4.2 it is possible to relax this interpolation property when fitting a kriging model. This option is useful for leveraging the flexibility of kriging surrogates to model data from physical experiments (e.g. if polynomial surrogates do not provide an adequate fit), or to model output from stochastic codes or deterministic codes subject to numerical or high frequency noise.

As described in the ensuing sections, diagnostics are available to assess the predictive capability of a chosen surrogate model. We synthesize the discussion above to make the following recommendations on the initial choice of surrogate model. Polynomial surrogates are often employed to model outputs from physical experiments. Polynomial or kriging surrogates may be utilized to model code output. Polynomial surrogates are appropriate when very few input parameters are being varied or prior knowledge indicates the trend contains minimal contribution from input nonlinearity and interaction. Subject to these considerations regarding trend complexity, polynomial surrogates are particularly appropriate for modeling stochastic or noisy code outputs. Kriging surrogates are ideal for settings in which weak or nonexistent prior information exists about trend complexity. They are often able to successfully infer the trend from a modest set of training runs, yielding an advantage over the effort often required to successfully fit a suitable polynomial surrogate in the presence of little or no prior information about trends. Kriging surrogates are also capable of fitting nonlinear trends not easily represented by low order polynomial surrogates. For deterministic codes, the uncertainty quantification provided by kriging surrogates conforms to the notion that prediction uncertainty should diminish near input points at which training runs were conducted. This behavior does not occur with polynomial surrogates.

4.1 Polynomial Regression Models

Linear, quadratic, and cubic polynomial surrogate models are available in Dakota. The form of the linear polynomial model is

$$Y(x) = \beta_0 + \sum_{i=1}^{M} \beta_i x_i + \epsilon(x)\,;$$

the form of the quadratic polynomial model is

$$Y(x) = \beta_0 + \sum_{i=1}^{M} \beta_i x_i + \sum_{i=1}^{M} \sum_{j \geq i}^{M} \beta_{ij} x_i x_j + \epsilon(x)\,;$$

and the form of the cubic polynomial model is

$$Y(x) = \beta_0 + \sum_{i=1}^{M} \beta_i x_i + \sum_{i=1}^{M} \sum_{j \geq i}^{M} \beta_{ij} x_i x_j + \sum_{i=1}^{M} \sum_{j \geq i}^{M} \sum_{k \geq j}^{M} \beta_{ijk} x_i x_j x_k + \epsilon(x)\,.$$

In all of the polynomial models, $Y(x)$ is the response of the polynomial model plus a mean-zero Gaussian error model ($\epsilon(x)$); the x_i, x_j, x_k terms are the components of the M-dimensional input parameter values; and the β_0, β_i, β_{ij}, and β_{ijk} terms are the polynomial coefficients. The number of coefficients, N_β, depends on the order of the polynomial model. For the linear polynomial, $N_\beta = M + 1$; for the quadratic polynomial, $N_\beta = (M+1)(M+2)/2$; and for the cubic polynomial, $N_\beta = (M^3 + 6M^2 + 11M + 6)/6$. The errors $\epsilon(\cdot)$ associated with each output are assumed to be independently distributed and have constant variance σ^2.

There must be at least N_β data samples (i.e. $N \geq N_\beta$) in order to form a fully determined linear system and solve for the polynomial coefficients β. In most applications, the linear system will be over-determined (i.e. $N > N_\beta$). To solve such systems, Dakota employs a least-squares approach involving a QR factorization-based numerical method. Note that $N \geq N_\beta + 1$ is required to estimate both β and the error variance σ^2.

The maximum likelihood value of β is computed via ordinary least squares,

$$\hat{\beta} = \left(G^T G\right)^{-1} G^T y\,.$$

Here y is the N-vector of observed experimental data and G is an N by N_β matrix that contains evaluations of the polynomial basis functions at all runs in the N by M sample design X, $G_{ij} = g_j\left(X_i^T\right)$, $i = 1, \ldots, N$; $j = 1, \ldots, N_\beta$.

The emulator mean $\tilde{\mu}(x)$ evaluated at input x is a best linear unbiased predictor of the surrogate $Y(x)$,

$$\tilde{\mu}(x) = g^T(x)\hat{\beta}\,.$$

The emulator variance $\tilde{\sigma}^2(x)$ (here, the mean squared error of prediction) evaluated at input x provides a spatially (over the input parameter space) varying measure of prediction uncertainty,

$$\tilde{\sigma}^2(x) = \widehat{\sigma^2}\left(1 + g^T(x)\left(G^T G\right)^{-1} g(x)\right)\,,$$

where an unbiased estimate of σ^2 is

$$\widehat{\sigma^2} = \frac{\left(y - G\hat{\beta}\right)^T \left(y - G\hat{\beta}\right)}{N - N_\beta}\,.$$

The utility of polynomial models stems from two sources: (1) over a small portion of the parameter space, a low-order polynomial model is often an accurate approximation to the true data trends, and (2) the least-squares procedure provides a surface fit that smooths out noise in the data. For this reason, surrogate-based optimization is often successful when using polynomial models, particularly quadratic models. However, a polynomial surface fit may not be the best choice for modeling data trends over the entire parameter space, unless it is known a priori that the true data trends are close to linear, quadratic, or cubic. Furthermore, in general, polynomial models will not *interpolate*, i.e. predict with zero uncertainty, the data they are built from. If interpolation is desired (as with deterministic computational models subject to negligible numerical noise), or if lack of fit is observed, users should consider the kriging emulators described in the following section. See [34] for additional information on polynomial models.

4.1.1 Fitting Polynomial Surrogates in Dakota

In the following discussion, `typewriter font` indicates the names of Dakota input or output files, keywords, commands, or results. All input, output, and log files mentioned in this subsection can be found in `examples/SurrogateModels`. Listings 4.1 and 4.2 together show the Dakota input file `cantilever_polynomial_eval.in` for evaluating a quadratic polynomial emulator of the cantilever beam model outputs at user-specified values of the inputs x. The values of x at which to evaluate the emulator are given in the `list_parameter_study` block of the input file (line 21). The keyword `sampling` (line 57), together with `sample_type lhs` (line 59) tells Dakota to generate a Latin hypercube sample. Dakota then runs the cantilever beam model on the resulting 60 input settings (line 60), producing output for the three indicated response variables (area, stress, and displacement). The design and output results are then used by Dakota to fit a quadratic polynomial model to each response (line 36). Dakota also allows `linear` and `cubic` polynomial trend options. Finally, predicted responses corresponding to each user specified input setting are written to the file `cantilever_polynomial_evals.dat` (line 8), reproduced here in edited form:

```
%eval_id           R           E           X           Y           w           t        area      stress  displacement
       1 32887.97269 36889581.78 408.9821034   1148.6638 2.099541047 2.004037063 4.207558073 76546.06547   5.914253488
       2 43482.92921 16550393.66 639.6736606 1051.841799 2.014840025  2.19097364 4.414461384 64938.20937   11.96142127
       3 36220.72048 44845837.48 561.1743862 935.4138043 2.109386061 2.136922015 4.507593512 57481.41311   2.627795226
       4 47325.31136 25228026.72  348.841119  876.590786 2.190414084 2.082472944 4.561478066 28938.90198   5.259963951
```

By specifying a random number seed (line 58), the results obtained from running Dakota are repeatable, which is useful for regression testing purposes and for situations in which intermediate results (such as experimental designs) must be regenerated exactly in follow-on Dakota analyses.

In this example, we chose to generate a Latin hypercube sample to fit the quadratic surrogate model. Latin hypercube samples are frequently used to fit kriging surrogates (Section 4.2) for complex computational model quantities of interest due to their space-filling nature; however, they can be less efficient than various alternatives more commonly selected for fitting polynomial surrogates, particularly when modeling responses from physical experiments subject to observation error. Two such alternatives for obtaining quadratic emulators are available in Dakota, selected by replacing the `sampling` block with `dace central_composite` for central composite designs [8] or `dace box_behnken` for Box-Behnken designs [7]. Central composite designs generate $N = 1 + 2M + 2^M$ samples, while Box-Behnken designs generate $N = 1 + 4M(M-1)/2$ samples. A seed can also be specified with `dace` options for repeatability. The option `dace oas` allows for generation of orthogonal array designs [19] to support polynomial surrogate fits and main effects analysis (Chapter 3).

Dakota was run with the command `dakota -i cantilever_polynomial_eval.in >& reg.log &`. The log file `reg.log` provides leave-one-out cross-validation root mean squared prediction errors

Listing 4.1: Dakota input file producing predictions at user-specified inputs from a quadratic polynomial emulator for the cantilever beam problem.

```
# Build and evaluate a quadratic polynomial emulator of cantilever beam
# at a user specified set of points

# Top-level controls
environment
  method_pointer = 'EvalSurrogate'
  tabular_graphics_data
    tabular_graphics_file = 'cantilever_polynomial_evals.dat'

# Method to perform evaluations of the emulator
method
  id_method = 'EvalSurrogate'
  model_pointer = 'SurrogateModel'

  # Verbose will show the type form of the surrogate model
  output verbose

  # -----
  # Emulator evaluation option #1: Provide user specified inputs in the
  #                                Dakota input file
  list_parameter_study
    list_of_points
    #         R           E           X           Y           w           t
    32887.97269 36889581.78 408.9821034   1148.6638 2.099541047 2.004037063
    43482.92921 16550393.66 639.6736606 1051.841799 2.014840025  2.19097364
    36220.72048 44845837.48 561.1743862 935.4138043 2.109386061 2.136922015
    47325.31136 25228026.72  348.841119  876.590786 2.190414084 2.082472944
  # -----

# Surrogate model specification
model
  id_model = 'SurrogateModel'
  surrogate global
    dace_method_pointer = 'DesignMethod'
    # Quadratic polynomial model
    polynomial quadratic
    # compute and print diagnostics after build
    metrics 'rsquared' 'root_mean_squared'
      press
```

Listing 4.2: (Continued) Dakota input file producing predictions at user-specified inputs from a quadratic polynomial emulator for the cantilever beam problem.

```
variables,
  uniform_uncertain = 6
    upper_bounds    48000.  45.E+6  700.  1200.  2.2   2.2
    lower_bounds    32000.  15.E+6  300.  800.   2.0   2.0
    descriptors     'R' 'E' 'X' 'Y' 'w' 't'

responses
  response_functions = 3
  descriptors = 'area' 'stress' 'displacement'
  no_gradients
  no_hessians

# Method to generate a 60 run Latin hypercube design to build the emulator
method
  id_method = 'DesignMethod'
  model_pointer = 'SimulationModel'
  sampling
    seed = 20
    sample_type lhs
    samples = 60

# The true simulation model to evaluate to build the emulator
model
  id_model = 'SimulationModel'
  single
    interface_pointer = 'SimulationInterface'

interface,
  id_interface = 'SimulationInterface'
  direct
    analysis_driver = 'mod_cantilever'
```

(input file, line 39) for each output, which evaluated to `8.3476172226e-16`, `5.2744083954e+01`, and `3.7407615270e-01` for area, stress, and displacement respectively. The root mean squared prediction error (RMSPE) provides a good overall estimate of emulator out-of-sample predictive capability.

A K-fold cross-validation procedure is also available in Dakota by replacing the `press` option (line 39) with `cross_validation folds = K`. The K-fold cross-validation procedure involves randomly partitioning the training data into K subsets, each containing approximately $100/K\%$ of the data. For each subset, the $100/K\%$ of the design runs and their corresponding outputs are "held out", and the surrogate refitted with the remaining $100 * (K-1)/K\%$ of the design runs and associated outputs. The $100/K\%$ hold-out points are then predicted by the refitted emulator. Thus a total of K surrogate rebuilds are conducted, each leaving out $100/K\%$ of the original training data, and the RMSPE is computed for all of the hold-out design points over the whole procedure. Running Dakota with $K = 10$, the RMSPE evaluated to `6.1748098826e-16`, `5.3512924399e+01`, and `3.8966016172e-01` for area, stress, and displacement respectively. These results are qualitatively similar to the leave-one-out cross-validation results. Note that the leave-one-out cross-validation procedure is equivalent to N-fold cross-validation.

4.2 Kriging and Gaussian Process Models

The set of techniques known as kriging were originally developed in the geostatistics and spatial statistics communities to produce maps of underground geologic deposits based on a set of widely and irregularly spaced borehole sites [10]. Building a kriging model typically involves the

1. Choice of a trend function,
2. Choice of a correlation function, and
3. Estimation of correlation parameters.

Suppose outputs (e.g. deposits from the borehole sites) are to be collected at the N inputs $\{x_1, \ldots, x_N\}$ to train a kriging surrogate $Y(x)$ for the output at input x. The emulator used for prediction of $Y(x)$ is assumed to be linear in these outputs, $\hat{Y}(x) = \sum_{i=1}^{N} c_i(x)\, Y(x_i)$. The coefficient vector $c(x)$ is determined by minimizing the mean squared error of prediction $E\left[\left(Y(x) - \hat{Y}(x)\right)^2\right]$ for fixed correlation parameters, subject to the unbiasedness constraint $E\left[\hat{Y}(x)\right] = E\left[Y(x)\right]$. The notation $E[\,\cdot\,]$ indicates expected value. The optimized mean squared error of prediction quantifies the uncertainty in prediction of $Y(x)$.

Kriging surrogates can also be derived from Gaussian processes, which further assume that arbitrary collections of observed outputs follow multivariate Gaussian distributions for fixed trend, variance and correlation parameters. In Dakota, the Gaussian process (GP) framework is adopted to facilitate covariance parameter estimation through techniques such as maximum likelihood and uncertainty quantification via standard statistical inference methods.

A kriging surrogate, $Y(x)$, consists of a trend function (frequently a linear model $g^T(x)\beta$) plus a Gaussian process error model ($\epsilon(x)$) that modifies the trend function locally,

$$Y(x) = g^T(x)\beta + \epsilon(x)\,.$$

This specifies a stochastic process representation of the unknown true surface $f(x)$. The error process $\epsilon(x)$ is assumed initially to have mean zero and constant variance σ^2. Furthermore, correlated errors

in the input space are allowed by specifying a covariance function. In particular, the covariance between the errors at two arbitrary input locations x and x' is modeled as

$$\text{Cov}\left(Y(x), Y(x')\right) = \text{Cov}\left(\epsilon(x), \epsilon(x')\right) = \sigma^2 \; r\left(x, x'\right) \; ,$$

where $r\left(x, x'\right)$ is a correlation function (i.e. a symmetric, positive definite function satisfying $r\left(x, x\right) = 1$). In the following, this correlation function is assumed to depend on the values of unknown parameters ϕ, designated $r\left(x, x'|\phi\right)$.

When the true surface results from evaluation of a deterministic computational model, the error process $\epsilon(x)$ can be specified in such a way that the emulator will interpolate, with zero uncertainty, the model runs it was built from. This is accomplished through selection of a correlation function with requisite smoothness properties, discussed further below. The error process specification is modified as follows to accommodate physical experiments subject to measurement or replicate variability, stochastic computational models, or deterministic computational models for which numerical or high frequency noise is of concern,

$$\text{Cov}\left(\epsilon(x), \epsilon(x')\right) = \sigma^2 \; r\left(x, x'|\phi\right) + \Delta^2 \delta\left(x - x'\right) \; ,$$

where

$$\delta\left(x - x'\right) = \begin{cases} 1 & \text{if } x - x' = 0 \\ 0 & \text{otherwise} \end{cases}$$

and Δ^2 is the variance of the observational or numerical error. In the ensuing discussion, the term "nugget" refers to the ratio $\eta = \Delta^2/\sigma^2$.

Figure 4.1 illustrates the behavior of emulators that interpolate versus emulators that smooth output data. In each panel, the red filled circles represent observed outputs and the red curve designates the emulator mean. The gray shaded area outlines the ensemble of pointwise 95% prediction intervals as a function of the input x, computed using the emulator standard error. That is, for any specified x, there is 95% probability that a future sampled output would lie between the lower and upper bounds of the gray shaded area. The left panel illustrates an error process selected for interpolation. Note the emulator mean passes directly through the three observed outputs, and there is no emulator uncertainty at these points (zero width to the prediction intervals). Prediction uncertainty grows as the input level x moves away from any input location at which output is observed. The right panel illustrates an error process selected for smoothing. In this case, the emulator mean defines a smooth surface that predicts the four observed outputs without being required to recover their values exactly with zero uncertainty. The prediction intervals do not narrow as x approaches an input location at which output is observed. Kriging models with no nugget effect ($\eta = 0$) will interpolate, while polynomial models or kriging models with a positive nugget effect ($\eta > 0$) will smooth.

By convention, the terms simple kriging, ordinary kriging, and universal kriging are used to indicate the three most common choices for the trend function. In simple kriging, the trend is treated as a known constant, usually zero, $g^T(x)\beta \equiv 0$. Universal kriging [31] uses a general polynomial trend model $g^T(x)\beta$ with coefficients determined by generalized least squares regression. Dakota allows specification of linear or quadratic trend models. For quadratic models, the user can choose to include or omit the interaction terms among the input variables. Ordinary kriging is essentially universal kriging with a trend order of zero, i.e. the trend function is treated as an unknown constant and $g(x) = 1$. As before, N_β denotes the number of basis functions in $g(x)$ and therefore the number of elements in the vector β. Ordinary kriging is typically selected when emulating code output, as GP models often have sufficient flexibility to detect complex trends without the need for estimating the additional parameters required by a more complex trend function. However, if prior knowledge

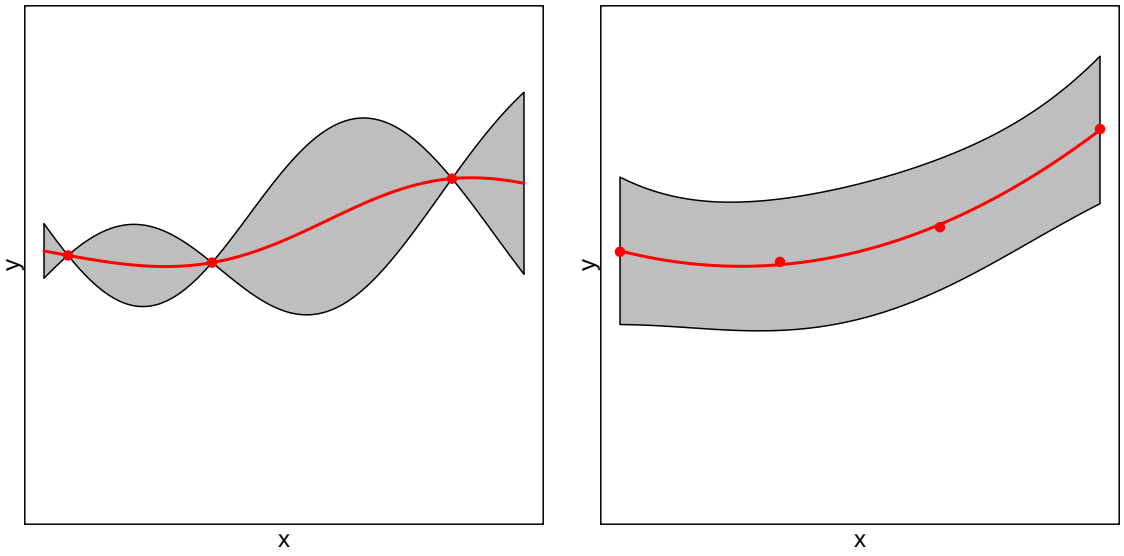


Figure 4.1: Emulator that interpolates (left) and smooths (right) observed output.

about more complex trends is available, it should be incorporated into trend function specification, particularly if the emulator is to be used for extrapolation.

The maximum likelihood value of β for fixed correlation parameters is computed via generalized least squares utilizing the correlation matrix $R(\phi)$ of the error process,

$$\hat{\beta}(\phi) = \left(G^T R^{-1}(\phi)\, G\right)^{-1} G^T R^{-1}(\phi)\, y\,.$$

Here G is a N by N_β matrix that contains evaluations of the polynomial basis functions at all runs in the N by M sample design X, $G_{ij} = g_j\left(X_i^T\right)$, $i = 1, \ldots, N$; $j = 1, \ldots, N_\beta$. The real, symmetric, positive-definite correlation matrix $R(\phi)$ from the error model contains evaluations of the correlation function $r(\cdot, \cdot|\phi)$ at all pairwise combinations of runs in the sample design X,

$$R_{ij}(\phi) = R_{ji}(\phi) = r\left(X_i^T, X_j^T|\phi\right) = r\left(X_j^T, X_i^T|\phi\right)$$

There is a single family of correlation functions implemented in Dakota, the power exponential:

$$r\left(x_i, x_j|\phi\right) = \exp\left(-\sum_{k=1}^{M} \phi_k \left|x_{ik} - x_{jk}\right|^\gamma\right)$$

where $0 < \gamma \leq 2$ and $\phi_k > 0$. The sample paths of a process equipped with this correlation function are everywhere continuous and nowhere differentiable for $0 < \gamma < 2$, and analytic for $\gamma = 2$. Dakota allows only this latter specification, referred to as the squared exponential or Gaussian correlation function. Note that $N \geq N_\beta + M + 1$ is required to estimate β, the error variance σ^2, and the correlation function parameters ϕ. Estimation of the nugget η in addition to all other surrogate parameters requires $N \geq N_\beta + M + 2$.

The emulator mean $\tilde{\mu}(x|\phi)$ evaluated at input x is a best linear unbiased predictor of the surrogate $Y(x)$,

$$\tilde{\mu}(x|\phi) = g^T(x)\hat{\beta}(\phi) + \bar{r}^T(x|\phi)R^{-1}(\phi)\left(y - G\hat{\beta}(\phi)\right)\,,$$

where $\bar{r}_i(x|\phi) = r\left(x, X_i^T|\phi\right)$. This emulator will interpolate the data that the kriging model was built from as long as no nugget is specified or fit and its correlation matrix $R(\phi)$ is numerically non-singular.

The emulator variance $\tilde{\sigma}^2(x|\phi)$ (here, the mean squared error of prediction) evaluated at input x provides a spatially varying measure of prediction uncertainty,

$$\tilde{\sigma}^2(x|\phi) = \widehat{\sigma^2}(\phi)\left(1 - \bar{r}^T(x|\phi)R^{-1}(\phi)\,\bar{r}(x|\phi) + \left(g(x) - G^T R^{-1}(\phi)\,\bar{r}(x|\phi)\right)^T \left(G^T R^{-1}(\phi)\,G\right)^{-1}\left(g(x) - G^T R^{-1}(\phi)\,\bar{r}(x|\phi)\right)\right),$$

where an unbiased estimate of σ^2 for fixed correlation parameters is

$$\widehat{\sigma^2}(\phi) = \frac{\left(y - G\hat{\beta}(\phi)\right)^T R^{-1}(\phi)\left(y - G\hat{\beta}(\phi)\right)}{N - N_\beta}.$$

Dakota completes construction of the kriging model by using optimization to find a set of correlation parameters ϕ (and if applicable, nugget η) that maximize the likelihood of observing the available data. This is equivalent to minimizing the following objective function,

$$\text{obj}(\phi) = \log\left(\widehat{\sigma^2}(\phi)\right) + \frac{\log\left(\det\left(R(\phi)\right)\right) + \log\left(\det\left(G^T R^{-1}(\phi)\,G\right)\right)}{N - N_\beta}.$$

Polynomial surrogates are a special case of kriging surrogates, resulting from setting $r\left(x, x'\right) = 1$ if $x = x'$ and 0 otherwise, and $\Delta = 0$.

4.2.1 Fitting Kriging Surrogates in Dakota

In the following discussion, `typewriter font` indicates the names of Dakota input or output files, keywords, commands, or results. All input, output, and log files mentioned in this subsection can be found in `examples/SurrogateModels`. Listings 4.3 and 4.4 together show the Dakota input file `cantilever_gp_eval.in` for evaluating a GP emulator of the cantilever beam model output at user specified input settings. This surrogate model assumes unknown constant trend (line 34), representing the default ordinary kriging specification. Universal kriging options are also available, by specifying a `linear`, `reduced_quadratic`, or `quadratic` trend. The reduced quadratic option fits a quadratic trend in the absence of interaction terms. This job shows how these input settings can be read from the user specified input file `cantilever_user_points.dat` (line 23), which has the following format:

%eval_id	interface	R	E	X	Y	w	t
1	NO_ID	32887.97269	36889581.78	408.9821034	1148.6638	2.099541047	2.004037063
2	NO_ID	43482.92921	16550393.66	639.6736606	1051.841799	2.014840025	2.19097364
3	NO_ID	36220.72048	44845837.48	561.1743862	935.4138043	2.109386061	2.136922015
4	NO_ID	47325.31136	25228026.72	348.841119	876.590786	2.190414084	2.082472944

The keyword `sampling` (line 55) together with `sample_type lhs` (line 57) tells Dakota to generate a Latin hypercube sample. Dakota then runs the cantilever beam model on the resulting 60 input settings (line 58), producing output for the three indicated response variables (area, stress, and displacement). The design and output results are then used by Dakota to fit a GP with estimated constant trend to each response, and finally predicted responses corresponding to each user specified input setting are written to the file `cantilever_gp_evals.dat` (line 8), reproduced here in edited form:

```
%eval_id          R          E          X          Y          w          t       area      stress  displacement
       1 32887.97269 36889581.78 408.9821034   1148.6638 2.099541047 2.004037063 4.209402759 76378.17522   5.360993694
       2 43482.92921 16550393.66 639.6736606 1051.841799 2.014840025  2.19097364 4.413155956  65000.4832   12.41942451
       3 36220.72048 44845837.48 561.1743862 935.4138043 2.109386061 2.136922015 4.506773804  57529.4144   2.511857535
       4 47325.31136 25228026.72  348.841119  876.590786 2.190414084 2.082472944 4.560423301 29135.16863   5.260298986
```

By specifying a random number seed (line 56), the results obtained from running Dakota are repeatable.

Listing 4.3: Dakota input file producing predictions at user specified inputs from a GP emulator with estimated constant trend for the cantilever beam problem.

```
1  # Build and evaluate a Gaussian process emulator of cantilever beam
   # at a user specified set of points
3
   # Top-level controls
5  environment
     method_pointer = 'EvalSurrogate'
7    tabular_graphics_data
       tabular_graphics_file = 'cantilever_gp_evals.dat'
9
11 # Method to perform evaluations of the emulator
   method
13   id_method = 'EvalSurrogate'
     model_pointer = 'SurrogateModel'
15
     # Verbose will show the type form of the surrogate model
17   output verbose

19   # -----
     # Emulator evaluation option #2: Provide user specified inputs in a
21   #                                separate file
     list_parameter_study
23     import_points_file 'cantilever_user_points.dat'
     # -----
25
   # Surrogate model specification
27 model
     id_model = 'SurrogateModel'
29   surrogate global
       dace_method_pointer = 'DesignMethod'
31     # GP model
       gaussian_process surfpack
33       trend
           constant
35     # compute and print diagnostics after build
       metrics 'rsquared' 'root_mean_squared'
37       press
```

A minimum run size was previously indicated based on the desire to obtain at least as many runs as unknown surrogate parameters. However, more conservative run sizes are often selected, such as a factor of 10 times the number of "active inputs." Since this latter quantity is often not known, $N = 10M$ is often selected [29]. With $M = 6$, this is the basis for the run size of 60 chosen for this example.

Listing 4.4: (Continued) Dakota input file producing predictions at user specified inputs from a GP emulator with estimated constant trend for the cantilever beam problem.

```
variables,
  uniform_uncertain = 6
    upper_bounds    48000.  45.E+6  700.  1200.  2.2   2.2
    lower_bounds    32000.  15.E+6  300.  800.   2.0   2.0
    descriptors     'R' 'E' 'X' 'Y' 'w' 't'

responses
  response_functions = 3
  descriptors = 'area' 'stress' 'displacement'
  no_gradients
  no_hessians

# Method to generate a 60 run Latin hypercube design to build the emulator
method
  id_method = 'DesignMethod'
  model_pointer = 'SimulationModel'
  sampling
    seed = 20
    sample_type lhs
    samples = 60

# The true simulation model to evaluate to build the emulator
model
  id_model = 'SimulationModel'
  single
    interface_pointer = 'SimulationInterface'

interface,
  id_interface = 'SimulationInterface'
  direct
    analysis_driver = 'mod_cantilever'
```

Dakota was run with the command `dakota -i cantilever_gp_eval.in >& gp.log &`. The log file `gp.log` provides the surrogate parameter estimates $\hat{\phi}$, $\hat{\beta}(\hat{\phi})$ and $\widehat{\sigma^2}(\hat{\phi})$, as well as the leave-one-out cross-validation RMSPEs (input file, line 37) for each output, which evaluated to `1.1264992653e-03`, `1.4913388456e+02`, and `1.1507679873e-01` for area, stress, and displacement respectively. For $K = 10$, the cross-validation RMSPEs for each output evaluated to `1.2088124274e-03`, `1.4905338700e+02`, and `1.4178321830e-01` for area, stress, and displacement respectively. Kriging fits can be adversely affected if the proportion of runs held out for cross-validation is too large (problem dependent), so it is typical to use $K = N$ (i.e. leave-one-out) in this setting.

Table 4.1 collects the four predictions of each output made by the quadratic polynomial and kriging emulators for comparison with direct code calculations. These predictions track variation in the code calculations well, an observation consistent with the small size of the cross-validation RMSPEs relative to the observed range in the calculations for each output.

Table 4.1: Polynomial and kriging emulator predictions compared with code calculations.

Area			
Case	Calculation	Polynomial	Kriging
1	4.207558073	4.207558073	4.209402759
2	4.414461384	4.414461384	4.413155956
3	4.507593512	4.507593512	4.506773804
4	4.561478066	4.561478066	4.560423301

Stress			
Case	Calculation	Polynomial	Kriging
1	76625.08025	76546.06547	76378.17522
2	64919.07096	64938.20937	65000.4832
3	57457.96055	57481.41311	57529.4144
4	28991.46319	28938.90198	29135.16863

Displacement			
Case	Calculation	Polynomial	Kriging
1	5.495297702	5.914253488	5.360993694
2	12.52267129	11.96142127	12.41942451
3	2.506561572	2.627795226	2.511857535
4	5.213217663	5.259963951	5.260298986

4.3 Summary

Table 4.2 summarizes the essential Dakota options for specifying polynomial regression or kriging models, with guidance on which method is to be preferred based on the nature of the physical/-computational experiment output and assumptions about the statistical modeling of residual error. Additional details on fitting kriging and other surrogates such as multivariate adaptive regression splines (MARS), simple artificial neural networks, or basic radial basis functions can be found in the Surfpack User's Manual [11].

Table 4.2: Guidelines for selection of surrogate methods.

Method Classification	**Applications**	**Applicable Methods**
polynomial regression	smooth fit to physical experiment response or stochastic/noisy computational experiment response [i.i.d. residual errors]	`polynomial` `linear` | `quadratic` | `cubic`
kriging	interpolation of deterministic smooth computational experiment response (specify trend option only) or smooth fit to stochastic/noisy computational experiment response (specify trend and nugget options) [correlated residual errors]	`gaussian_process surfpack` `trend` `constant` | `linear` | `reduced_quadratic` | `quadratic` `nugget` ($\eta > 0$) | `find_nugget`

Chapter 5

Optimization and Deterministic Calibration

The objective of optimization algorithms is to minimize (or maximize) an objective function, typically calculated by the user simulation code, subject to constraints on design variables and responses. Examples of optimization goals include:

- Identify system designs with maximal performance; e.g., case geometry that minimizes drag and weight, yet is sufficiently strong and safe.
- Determine operational settings that maximize system performance, e.g., fuel re-loading pattern yielding the smoothest nuclear reactor power distribution while maximizing output.
- Identify minimum-cost system designs/operational settings, e.g., delivery network that minimizes cost while also minimizing environmental impact.
- Identify best/worst case scenarios, e.g., impact conditions that incur the most damage.
- Calibration: Determine parameter values that maximize agreement between simulation response and target response.

The last goal is a critical use case for CASL. The calibration (parameter estimation, inverse problem) process involves adjusting input parameters to optimally fit a model to experimental or high-fidelity computational model data, find operational settings that best match a prescribed performance profile, or determine source terms for an observed phenomenon. Any Dakota optimization method can be applied to calibration problems, though some are tailored to efficiently address local least squares problem formulations. This chapter emphasizes deterministic model calibration, while non-deterministic approaches such as Bayesian methods are treated in Uncertainty Quantification, Chapter 6.

Available optimization approaches in Dakota include well-tested, proven gradient-based, derivative-free local, and global methods for use in science and engineering design applications. Dakota also offers more advanced algorithms, e.g., to manage multi-objective optimization or perform surrogate-based minimization (useful for noisy or expensive problems). A more extensive treatment of these can be found in "Optimization Capabilities," "Nonlinear Least Squares Capabilities," and "Surrogate-Based Minimization" chapters of the Dakota User's Manual [1]. This chapter continues by introducing optimization terminology needed to select from the available approaches.

5.1 Terminology and Problem Formulations

This section provides a basic introduction to the mathematical formulation of optimization problems. The primary goal of this section is to introduce terms relating to these topics, and is not intended to be a description of theory or numerical algorithms. For further details, consult [4], [15], [17], [35], and [50].

A general optimization problem is formulated as follows:

$$\begin{array}{rcl} \text{minimize:} & f(x) & \textit{objective function} \\ \text{over:} & x \in \Re^M & \textit{design variables} \\ \text{subject to:} & g_L \leq g(x) \leq g_U & \textit{nonlinear inequality constraints} \\ & h(x) = h_t & \textit{nonlinear equality constraints} \\ & a_L \leq A_i x \leq a_U & \textit{linear inequality constraints} \\ & A_e x = a_t & \textit{linear equality constraints} \\ & x_L \leq x \leq x_U & \textit{bound constraints} \end{array} \tag{5.1}$$

In this formulation, $x = [x_1, x_2, \ldots, x_M]$ is an M-dimensional vector of real-valued *design variables* or *design parameters*. The M-dimensional vectors x_L and x_U, are the lower and upper bounds, respectively, on the design parameters. These bounds define the allowable values for the elements of x, and the set of all allowable values is termed the *design space* or the *parameter space*. A *design point* or a *sample point* is a particular set of values within the parameter space.

The optimization goal is to minimize the *objective function*, $f(x)$, while satisfying the constraints. Constraints can be categorized as either linear or nonlinear and as either inequality or equality. The *nonlinear inequality constraints*, $g(x)$, are "2-sided," in that they have both lower and upper bounds, g_L and g_U, respectively. The *nonlinear equality constraints*, $h(x)$, have target values specified by h_t. The linear inequality constraints create a linear system $A_i x$, where A_i is the coefficient matrix for the linear system. These constraints are also 2-sided as they have lower and upper bounds, a_L and a_U, respectively. The linear equality constraints create a linear system $A_e x$, where A_e is the coefficient matrix for the linear system and a_t are the target values. The constraints partition the parameter space into feasible and infeasible regions. A design point is said to be *feasible* if and only if it satisfies all of the constraints. Correspondingly, a design point is said to be *infeasible* if it violates one or more of the constraints.

Many different methods exist to solve the optimization problem given by (5.1), all of which iterate on x in some manner. That is, an initial value for each parameter in x is chosen, the *response quantities*, $f(x)$, $g(x)$, $h(x)$, are computed, often by running a simulation, and some algorithm is applied to generate a new x that will either reduce the objective function, reduce the amount of infeasibility, or both. To facilitate a general presentation of these methods, three criteria will be used in the following discussion to differentiate them: optimization problem type, search goal, and search method.

The **optimization problem type** can be characterized both by the types of constraints present in the problem and by the linearity or nonlinearity of the objective and constraint functions. For constraint categorization, a hierarchy of complexity exists for optimization algorithms, ranging from simple bound constraints, through linear constraints, to full nonlinear constraints. By the nature of this increasing complexity, optimization problem categorizations are inclusive of all constraint types up to a particular level of complexity. That is, an *unconstrained problem* has no constraints, a *bound-constrained problem* has only lower and upper bounds on the design parameters, a *linearly-constrained problem* has linear constraints (and optionally bound constraints), and a *nonlinearly-constrained problem* may contain the full range of nonlinear, linear, and bound constraints (though

may omit linear or bound constraints if not applicable). If all of the linear and nonlinear constraints are equality constraints, then this is referred to as an *equality-constrained problem*, and if all of the linear and nonlinear constraints are inequality constraints, then this is referred to as an *inequality-constrained problem*.

Further categorizations can be made based on the linearity of the objective and constraint functions. A problem where the objective function and all constraints are linear is called a *linear programming (LP) problem*. These types of problems commonly arise in scheduling, logistics, and resource allocation applications. Likewise, a problem where at least some of the objective and constraint functions are nonlinear is called a *nonlinear programming (NLP) problem*. These NLP problems predominate in engineering applications and are the primary focus of Dakota.

The **search goal** refers to the ultimate objective of the optimization algorithm, i.e., either global or local optimization. In *global optimization*, the goal is to find the design point that gives the lowest feasible objective function value over the entire parameter space. In contrast, in *local optimization*, the goal is to find a design point that is lowest relative to a "nearby" region of the parameter space. In almost all cases, global optimization will be more computationally expensive than local optimization. Thus, the user must choose an optimization algorithm with an appropriate search scope that best fits the problem goals and the computational budget.

The **search method** refers to the approach taken in the optimization algorithm to locate a new design point that has a lower objective function or is more feasible than the current design point. The search method can be classified as either *gradient-based* or *nongradient-based*. In a gradient-based algorithm, gradients of the response functions are computed to find the direction of improvement. The Hessian (matrix of second derivatives of objectives and constraints with respect to parameters) can also be used in these methods to identify curvature to distinguish local minima from maxima. Gradient-based optimization is the search method that underlies many efficient local optimization methods. However, a drawback to this approach is that gradients can be computationally expensive, inaccurate, or even nonexistent (the situation for Hessians is typically even worse). In such situations, nongradient-based search methods may be useful. There are numerous approaches to nongradient-based optimization. Some of the more well known of these include pattern search methods (nongradient-based local techniques) and genetic algorithms (nongradient-based global techniques).

5.1.1 Special Considerations for Calibration

Any Dakota optimization algorithm can be applied to calibration problems arising in parameter estimation, system identification, and test/analysis reconciliation. However, nonlinear least-squares methods are optimization algorithms that exploit the special structure of a least squares or sum-of-squares objective function [15]. Here the misfit between vectors of model responses and simulation data is measured in the Euclidean or two-norm.

To exploit the problem structure, more granularity is needed in the response data than is required for a typical optimization problem. That is, rather than using the sum-of-squares objective function and its gradient, least-squares iterators require each term used in the sum-of-squares formulation along with its gradient. This means that the functions in the Dakota response data set consist of the N individual least-squares terms along with any nonlinear inequality and equality constraints. These individual terms are often called *residuals* when they denote differences of observed quantities from values computed by the model whose parameters are being estimated.

The enhanced granularity needed for nonlinear least-squares algorithms allows for simplified computation of an approximate Hessian matrix. In Gauss-Newton-based methods for example, the true Hessian matrix is approximated by neglecting terms in which residuals multiply Hessians

(matrices of second partial derivatives) of residuals, under the assumption that the residuals are zero in expected value under the nonlinear least-squares model. As a result, residual function value and gradient information (first-order information) is sufficient to define the value, gradient, and approximate Hessian of the sum-of-squares objective function (second-order information).

In practice, least-squares solvers will tend to be significantly more efficient than general-purpose optimization algorithms when the Hessian approximation is a good one, i.e., when the neglected component has negligible effect at the solution. Specifically, they can exhibit the quadratic convergence rates of full Newton methods, even though only first-order information is used. Gauss-Newton-based least-squares solvers may experience difficulty when the residuals at the solution are significant. Dakota has three solvers customized to take advantage of the sum of squared residuals structure in this problem formulation. Least squares solvers may experience difficulty when the residuals at the solution are significant, although experience has shown that Dakota's NL2SOL method can handle some problems that are highly nonlinear and have nonzero residuals at the solution.

Specialized least squares solution algorithms can exploit the structure of a sum-of-squares objective function for problems of the form:

$$\begin{array}{rcl}
\text{minimize:} & f(\theta) = \sum_{i=1}^{N} [T_i(\theta)]^2 = \sum_{i=1}^{N} [y_i(\theta) - d_i]^2 & \textit{least squares objective function} \\
\text{over:} & \theta \in \Re^M & \textit{calibration variables} \\
\text{subject to:} & g_L \leq g(x) \leq g_U & \textit{nonlinear inequality constraints} \\
& h(\theta) = h_t & \textit{nonlinear equality constraints} \\
& a_L \leq A_i\theta \leq a_U & \textit{linear inequality constraints} \\
& A_e\theta = a_t & \textit{linear equality constraints} \\
& \theta_L \leq \theta \leq \theta_U & \textit{bound constraints}
\end{array}$$

where $f(\theta)$ is the objective function to be minimized and $T_i(\theta)$ is the i-th least squares term. The bound, linear, and nonlinear constraints are the same as described previously for (5.1). Specialized least squares algorithms are generally based on the Gauss-Newton approximation. When differentiating $f(\theta)$ twice, terms of $T_i(\theta)T_i''(\theta)$ and $[T_i'(\theta)]^2$ result. Because $T_i(\theta)$ is zero in expected value under the nonlinear least-squares model, the Hessian matrix of second derivatives of $f(\theta)$ can be approximated using only first derivatives of $T_i(\theta)$. As a result, Gauss-Newton algorithms exhibit quadratic convergence rates near the solution for those cases when the Hessian approximation is accurate, i.e. the neglected component has negligible effect at the solution. Thus, by exploiting the structure of the problem, the second order convergence characteristics of a full Newton algorithm can be obtained using only first order information from the least squares terms. For problems with nonsmooth gradients or poor finite difference approximations, see [23].

A common example for $T_i(\theta)$ might be the difference between experimental data and model predictions for a response quantity at a particular location and/or time step, i.e.:

$$T_i(\theta) = y_i(\theta) - d_i$$

where $y_i(\theta)$ is the response quantity predicted by the model and d_i is the corresponding experimental data. In this case, θ would have the meaning of model parameters which are not precisely known and are being calibrated to match available data. This class of problem is known by the terms parameter estimation, system identification, model calibration and test/analysis reconciliation, for example.

This overview of optimization problem formulations and goals approaches underscores that no single optimization method or algorithm works best for all types of optimization problems. The following section offers some basic guidelines for choosing one for specific optimization problems.

5.2 Recommended Methods

In selecting an optimization method, important considerations include the type of variables in the problem (continuous, discrete, mixed), whether a global search is needed or a local search is sufficient, and the required constraint support (unconstrained, bound constrained, or generally constrained). Less obvious, but equally important, considerations include the efficiency of convergence to an optimum (i.e., convergence rate) and the robustness of the method in the presence of challenging design space features (e.g., nonsmoothness). Sensitivity analysis (described in Chapter 3) is a critical precursor to assess problem characteristics prior to choosing and applying an optimization method.

For example the cantilever beam optimization problem posed in (2.3) in Section 2.1 has continuous design variables only, nonlinear inequality constraints, and bound constraints. Sensitivity analysis in Section 3.2 indicated that the objective and constraints are smooth functions of the design variables. This can also be directly inferred from the algebraic physics equations for the cantilever beam (2.1), though cannot for a general physics simulation.

Table 5.1 highlights a few key Dakota optimization approaches and problems for which they are suited. The following sections offer more details on the approaches and input file examples. Without considering specific problem knowledge or characteristics, a derivative-free local pattern search approach is typically a good starting point. If it doesn't find good solutions, move to a genetic algorithm. If it is too costly, move to a surrogate-based approach. However, when problems are smooth and not too multimodal, a local algorithm will outperform these other approaches.

Table 5.1: Guidance for selecting from the top recommended Dakota optimization algorithms.

algorithm type / Dakota method	**variable type**	**cost (samples)**	**goal and characteristics**
gradient-based local/ OPT++ Quasi-Newton	continuous	low/ medium	single local solution/improvement, assumes smooth input/output mapping
local calibration / OPT++ Gauss Newton	continuous	low/ medium	same as previous line, but tailored to least-squares calibration
derivative-free local / Coliny Pattern Search	continuous	medium/ high	single local solution; better when can't estimate derivatives
local w/surrogate / Surrogate-based Local	continuous	medium	same as "derivative-free local," but for noisier or more expensive simulations
global / Coliny Evolutionary Algorithm	continuous or discrete	high	global optimality, with ranked family of best solutions
global w/surrogate / Efficient Global	continuous	medium	same as "global," but for more expensive simulations

5.2.1 Gradient-Based Local Methods

Gradient-based optimizers are best suited for efficient navigation to a local minimum in the vicinity of the initial point. They are not intended to find global optima in nonconvex design spaces. For global optimization methods, see Section 5.2.3. Gradient-based optimization methods are highly efficient, with the best convergence rates of all of the local optimization methods, and are the methods of choice when the problem is smooth, unimodal, and well-behaved. However, these methods can be among the least robust when a problem exhibits nonsmooth, discontinuous, or multimodal

behavior. Figure 5.1 depicts a multimodal function on which gradient-based optimizers will typically find only a nearby local minimum. The derivative-free methods described in Section 5.2.2 are more appropriate for problems with some of these characteristics.

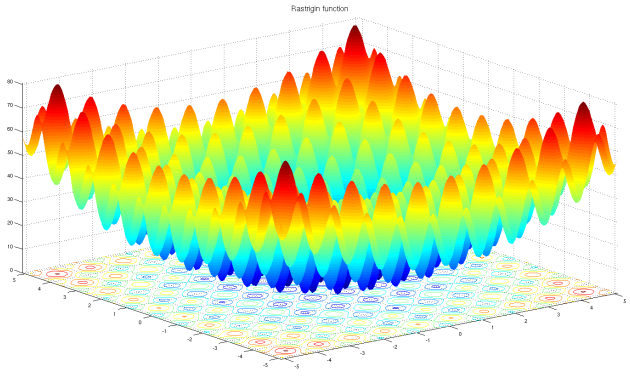


Figure 5.1: Surface plot with contours of an example function that is locally smooth, but globally multimodal.

Newton methods are representative of gradient-based optimization methods. These can be derived by applying Newton's method for root finding to $\nabla f(x) = 0$ to find a local minimum of the objective function. The resulting progression from current iterate x_n to next iterate x_{n+1} is then

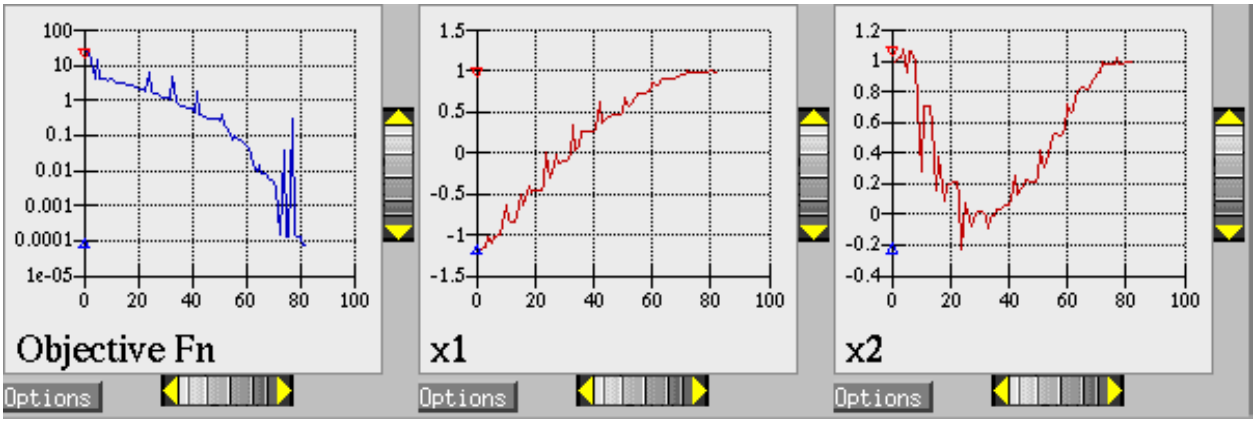
$$x_{n+1} = x_n - \left(\nabla^2 f(x_n)\right)^{-1} \nabla f(x_n).$$

This naive iteration directly requires the gradient and Hessian ($\nabla^2 f(x)$) of the objective function. In practice this basic iteration is enhanced with strategies to choose an appropriate step length to achieve expected decrease with each iteration, approximate the action of the Hessian-vector product when not directly available, and handle nonlinear constraints, for example via penalty methods.

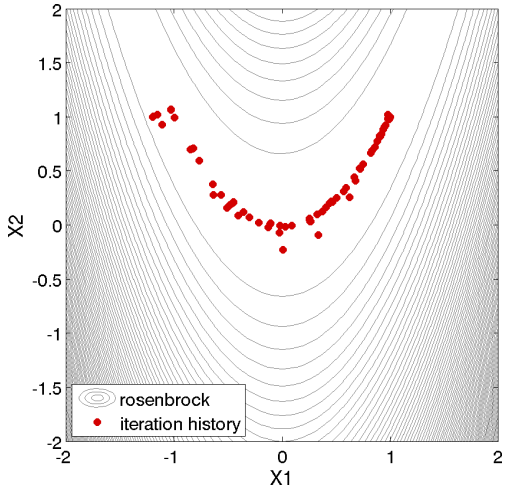
Gradient accuracy is a critical factor for gradient-based optimizers, as inaccurate derivatives will often lead to failures in the search or premature termination of the method. Analytic gradients and Hessians are ideal but often unavailable. If analytic gradient and Hessian information can be provided by an application code, a full Newton method will achieve quadratic convergence rates near the solution. If only gradient information is available and the Hessian information is approximated from an accumulation of gradient data, the superlinear convergence rates can be obtained. It is most often the case for engineering applications, however, that a finite difference method will be used by the optimization algorithm to estimate gradient values. Dakota allows the user to select the step size for these calculations, as well as choose between forward-difference and central-difference algorithms. The finite difference step size should be selected as small as possible, to allow for local accuracy and convergence, but not so small that the steps are "in the noise." This requires an assessment of the local smoothness of the response functions using, for example, a parameter study method. Central differencing will generally produce more reliable gradients than forward differencing but at roughly twice the expense.

A typical iteration history and search path for a gradient-based optimization solver is shown in Figure 5.2. Notice the algorithm efficiently going downhill with respect to the contours of the notional objective function.

Recommended method: The recommended local derivative-based solver in Dakota comes from the OPT++ package. A Dakota input example for the cantilever beam optimization problem



(a)



(b)

Figure 5.2: Gradient-based unconstrained optimization example: (a) screen capture of the Dakota graphics and (b) sequence of design points (dots) evaluated (line search points omitted).

(see Section 2.1) is shown in Listing 5.1. The iteration starts at the initial iterate for (w, t) specified in `initial_point` on line 17. The state variables are held fixed at their given values. The algorithm seeks to minimize the cantilever objective function (area), within the bound constraints specified on lines 16 and 18, subject to constraints on stress and displacement specified on line 31. The algorithm will terminate when the convergence criteria from line 8 is met. Notice the responses section specifies numerical gradients (line 35), indicating that Dakota should approximate the derivatives of model responses with respect to parameters via finite differences. Dakota's problem scaling is used to help better condition the constraints for presentation to the optimizer. If the response is a rough or strongly nonlinear function of the parameters, these approximations can be poor and yield bad performance or results. Figure 5.3 shows an excerpt of the Dakota output, showing the optimal design point found.

Variation for calibration: Listing 5.2 shows the Dakota input variation to directly treat calibration with a least-squares specific gradient-based solver. This example tunes the active variables (line 16) $\theta = (E, w, t)$ to match synthetic experimental data from a file, with (line 32) or without (line 31) added noise. The data were generated using the following tuning parameter val-

Listing 5.1: Dakota input file showing local gradient-based optimization on the cantilever beam problem.

```
environment
  tabular_data
    tabular_data_file 'cantilever_opt_grad.dat'
    custom_annotated header eval_id

method
  optpp_q_newton
    convergence_tolerance 1.0e-4
  scaling

model
  single

variables
  continuous_design = 2
    upper_bounds   4.0     4.0
    initial_point  2.5     2.5
    lower_bounds   1.0     1.0
    descriptors    'w'     't'
  continuous_state = 4
    initial_state  40000. 29.E+6 500. 1000.
    descriptors    'R'    'E'    'X'  'Y'

interface
  direct
    analysis_driver = 'mod_cantilever'

responses
  objective_functions = 1
  # constraints assumed <= 0 unless bounds given
  nonlinear_inequality_constraints = 2
    scale_types 'value'
    scales 1.0e3 1.0e-1
  descriptors = 'area' 'stress' 'displacement'
  numerical_gradients forward
    fd_step_size 1.0e-6
  no_hessians
```

```
<<<<< Function evaluation summary: 33 total (33 new, 0 duplicate)
<<<<< Best parameters          =
                     2.3520094791e+00 w
                     3.3263488002e+00 t
                     4.0000000000e+04 R
                     2.9000000000e+07 E
                     5.0000000000e+02 X
                     1.0000000000e+03 Y
<<<<< Best objective function  =
                     7.8236039089e+00
<<<<< Best constraint values   =
                    -6.4109754517e+02
                    -4.8144720935e-05
<<<<< Best data captured at function evaluation 31
```

Figure 5.3: Dakota output showing optimal local gradient-based optimization result for cantilever beam.

ues: $E = 2.85\text{E}7$, $w = 2.5$, $t = 3.0$, and fixed state parameter values: $R = 40000$, $X = 500$, $Y = 1000$. The input file specifies `calibration_terms` instead of `objective_functions` at line 30, indicating to Dakota that it should treat these responses as terms in a least-squares calibration problem. The data here is a set of three observations, one each for area, stress, and displacement. The NL2SOL solver is used for this example as it performs better than the default-recommended OPT++ Gauss-Newton solver.

The calibrated parameter values from the Dakota output are shown in Figure 5.4. Dakota has recovered the true values of the parameters, verifying the operation of the algorithm. The output from local calibration methods also includes confidence intervals on the parameters. With 95% confidence the true value of each parameter lies in the interval specified, given the misfit between the model and corresponding data. When this problem is exercised with noisy data, the confidence intervals expectedly grow larger.

For problems not suitable for local gradient-based optimization, any of the optimization methods discussed in the following sections can be applied to Dakota responses with `calibration_terms`. Dakota will automatically compute the objective function as the sum of squared residuals for presentation to the optimization algorithm.

5.2.2 Derivative-Free Local Methods

Derivative-free methods can be more robust and more inherently parallel than gradient-based approaches. They can be applied in situations were gradient calculations are too expensive or unreliable. In addition, some derivative-free methods can be used for global optimization which gradient-based techniques (see 5.2.1), by themselves, cannot. For these reasons, derivative-free methods are often go-to methods when the problem may be nonsmooth, multimodal, or poorly behaved. It is important to be aware, however, that they exhibit much slower convergence rates for finding an optimum, and as a result, tend to be much more computationally demanding than gradient-based methods. They often require from several hundred to a thousand or more function evaluations for local methods, depending on the number of variables, and may require from thousands to tens-of-thousands of function evaluations for global methods. Given the computational cost, it is often

Listing 5.2: Dakota input file showing deterministic local calibration with a least-squares solver on the cantilever beam problem.

```
environment
  tabular_data
    tabular_data_file 'cantilever_calibration.dat'
    custom_annotated header eval_id

method
  nl2sol
    convergence_tolerance 1.0e-6
#  output verbose

model
  single

variables
  active design
  continuous_design 3
    upper_bounds 31000000 10 10
    initial_point 29000000 4 4
    lower_bounds 27000000 1 1
    descriptors 'E' 'w' 't'
  continuous_state 3
    initial_state 40000 500 1000
    descriptors 'R' 'X' 'Y'

interface
  direct
    analysis_driver = 'mod_cantilever'

responses
  calibration_terms 3
    calibration_data_file = 'dakota_cantilever_examples.clean.dat'
    #calibration_data_file = 'dakota_cantilever_examples.error.dat'
    freeform
  descriptors = 'area' 'stress' 'displacement'
  analytic_gradients
  no_hessians
```

```
<<<<< Function evaluation summary: 18 total (17 new, 1 duplicate)
<<<<< Best parameters          =
                     2.8499999995e+07 E
                     2.5000000002e+00 w
                     2.9999999997e+00 t
                     4.0000000000e+04 R
                     5.0000000000e+02 X
                     1.0000000000e+03 Y
<<<<< Best residual norm =  2.9442886479e-06; 0.5 * norm^2 =  4.3344178210e-12
<<<<< Best residual terms =
                    -2.3943602656e-10
                     2.9442885534e-06
                     7.0639355476e-10
<<<<< Best model responses =
                     7.4999999998e+00
                     2.6666666696e+03
                     3.0864943605e-01
<<<<< Best data captured at function evaluation 17

Confidence Interval for E is [  2.8499511022e+07,  2.8500488969e+07 ]
Confidence Interval for w is [  2.4999189436e+00,  2.5000810568e+00 ]
Confidence Interval for t is [  2.9999176961e+00,  3.0000823033e+00 ]
```

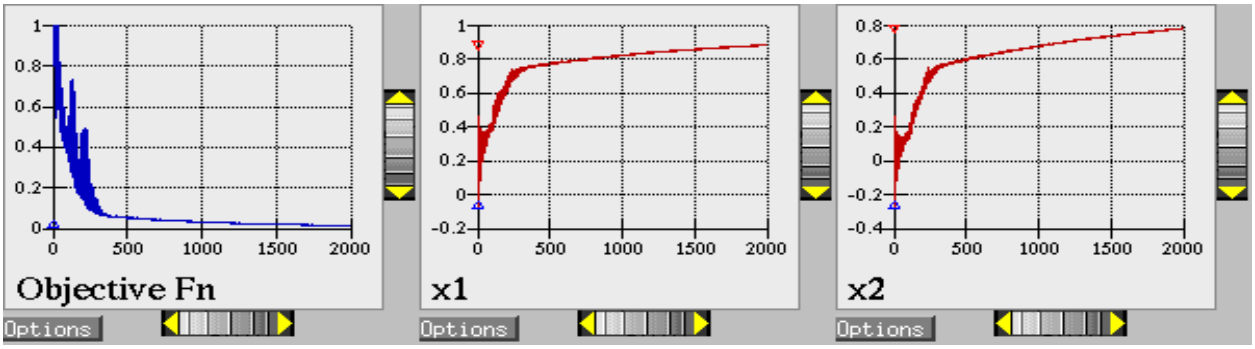
Figure 5.4: Dakota output showing optimal local gradient-based calibration result for cantilever beam, clean data.

prudent to use derivative-free methods to identify regions of interest and then use gradient-based methods to hone in on the solution. In addition to slow convergence, nonlinear constraint support in derivative-free methods is an open area of research and, while supported by many methods in Dakota, is not as refined as constraint support in gradient-based methods.

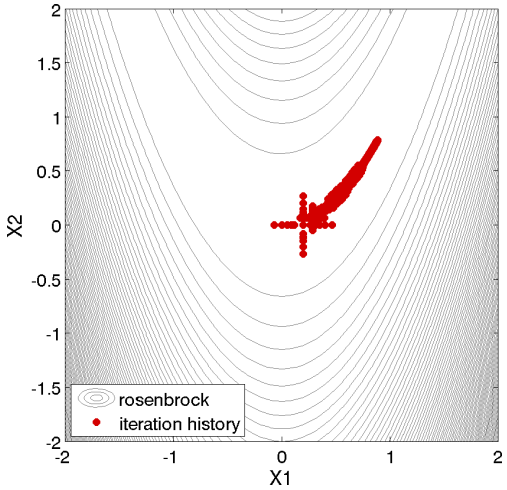
Local pattern search algorithms work by sampling the objective function at points on a stencil, often chosen to align with the coordinate axes. The stencil moves, expands, and contracts as the algorithm progresses. A typical pattern search iteration history is provided in Figures 5.5(a) and (b), which show the locations of the function evaluations used in the pattern search algorithm. Figure 5.5(c) provides a close-up view of the pattern search function evaluations used at the start of the algorithm (from a starting point $(x_1, x_2) = (0.0, 0.0)$. The coordinate pattern is clearly visible at the start of the iteration history, and the decreasing size of the coordinate pattern is evident at the design points move toward $(x_1, x_2) = (1.0, 1.0)$.

While pattern search algorithms are useful in many optimization problems, this example shows some of the drawbacks to this algorithm. While a pattern search method may make good initial progress towards an optimum, it is often slow to converge, here not fully converging after 2000 iterations. On a smooth, differentiable function such as that depicted, a nongradient-based method will not be as efficient as a gradient-based method. However, there are many engineering design applications where gradient information is inaccurate or unavailable, which renders gradient-based optimizers ineffective. Thus, pattern search algorithms are often good choices in complex engineering applications when the quality of gradient data is suspect.

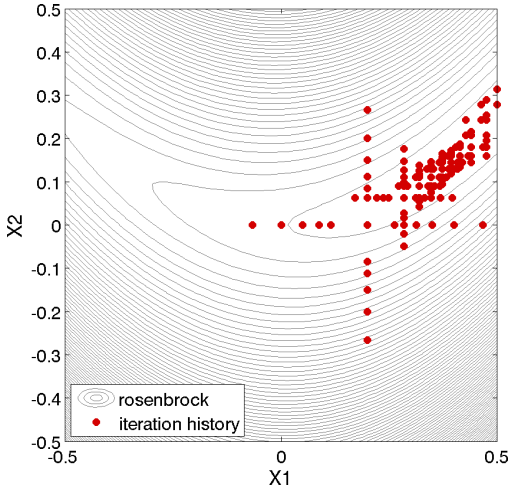
Recommended method: We recommend the Coliny Pattern Search algorithm for a derivative-free local method. A Dakota input file shown in Listing 5.3 applies a pattern search method to



(a)



(b)



(c)

Figure 5.5: Pattern search optimization example: (a) screen capture of the Dakota graphics, (b) sequence of design points (dots) evaluated and (c) close-up view illustrating the shape of the coordinate pattern used.

the cantilever beam optimization problem to minimize the objective function (area), subject to constraints on stress and displacement (see Section 2.1). The input is similar to the input file for the gradient-based optimization, except it has a different set of keywords in the method block of the input file (line 6–15), and the gradient specification in the responses block has been changed to `no_gradients` (line 39). The pattern search optimization algorithm used, `coliny_pattern_search` is part of the SCOLIB library [18]. See the Dakota Reference Manual [2] for more information on the *method* block commands that can be used with SCOLIB algorithms.

The tailing portion of the Dakota output shows the final design point and constraint values; see Figure 5.6. A solution similar to the gradient-based approach was found, though the constraint for displacement was slightly violated (this can often be addressed through larger constraint penalties in derivative-free methods; see commented option in the input file). However, the algorithm required about an order of magnitude more function evaluations.

5.2.3 Derivative-Free Global Methods

Dakota has a number of global optimization algorithms, including DiRECT, genetic algorithms, and surrogate-based approaches. Here we discuss genetic algorithms. In contrast to pattern search

Listing 5.3: Dakota input file showing local derivative-free optimization on the cantilever beam problem.

```
environment
  tabular_data
    tabular_data_file 'cantilever_opt_ps.dat'
    custom_annotated header eval_id

method
  max_iterations = 1000
  max_function_evaluations = 2000
  coliny_pattern_search
    solution_accuracy = 1e-4
    initial_delta = 0.5
    threshold_delta = 1e-4
    exploratory_moves basic_pattern
    contraction_factor = 0.75
#    constraint_penalty = 10000

model
  single

variables
  continuous_design = 2
    upper_bounds   4.0     4.0
    initial_point  2.5     2.5
    lower_bounds   1.0     1.0
    descriptors    'w'     't'
  continuous_state = 4
    initial_state  40000. 29.E+6 500. 1000.
    descriptors    'R'    'E'    'X'  'Y'

interface
  direct
    analysis_driver = 'mod_cantilever'

responses
  objective_functions = 1
  # constraints assumed <= 0 unless bounds given
  nonlinear_inequality_constraints = 2
  descriptors = 'area' 'stress' 'displacement'
  no_gradients
  no_hessians
```

```
<<<<< Function evaluation summary: 149 total (149 new, 0 duplicate)
<<<<< Best parameters          =
                     2.3875000000e+00 w
                     3.2493181357e+00 t
                     4.0000000000e+04 R
                     2.9000000000e+07 E
                     5.0000000000e+02 X
                     1.0000000000e+03 Y
<<<<< Best objective function  =
                     7.7577470490e+00
<<<<< Best constraint values   =
                    -1.6832688252e-01
                     4.1748455156e-02
<<<<< Best data captured at function evaluation 142
```

Figure 5.6: Dakota output showing results from local derivative-free optimization with a pattern search algorithm.

algorithms, which are local optimization methods, evolutionary algorithms (EA) are global optimization methods. EAs are best suited to optimization problems that have multiple local optima, and where gradients are either too expensive to compute or are not readily available.

Evolutionary algorithms work by generating an initial random sample in the parameter space, computing function values/constraints at those points, and then determining the next places to sample based on biological genetic selection principles of mutation and fitness. A simplified evolutionary or genetic algorithm for optimization would include:

1. Initialize a random population (sample) of individual x values.
2. Evaluate the fitness with respect to optimization objective and constraints.
3. Select more fit individuals as parents to reproduce.
4. Recombine and mutate to create a new population.
5. Iterate to (2.) until convergence is reached.

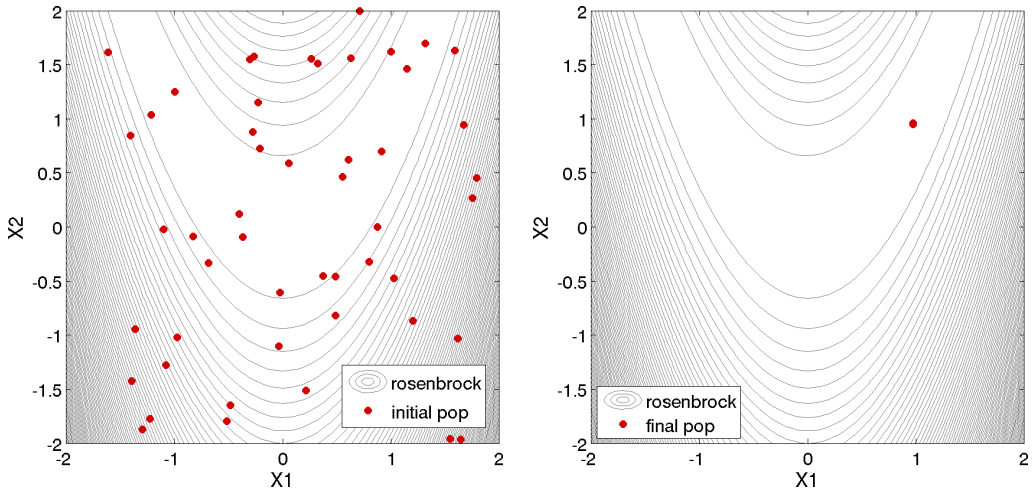
Here the implementation details of each step, for example how to select parents, how to combine and mutate, vary greatly among implementations, but are usually biologically inspired. Figure 5.7(a) shows the population of 50 randomly selected design points that comprise the first generation of the EA, and Figure 5.7(b) shows the final population of 50 design points, where most of the 50 points are clustered near $(x_1, x_2) = (0.98, 0.95)$.

Listing 5.4 shows a Dakota input file that uses an EA to solve the cantilever beam optimization problem as described in Section 2.1. Each generation of the EA has a population size of 50 (line 11). The algorithm will take at most 100 iterations comprising no more than 2000 function evaluations (lines 7 and 8). The EA software available in Dakota provides the user with much flexibility in choosing the settings used in the optimization process. The details of all the settings are not discussed here; see [2] and [18].

On completion, the file `cantilever_opt_ea.dat` provides a listing of the design parameter values and objective function values for all 2,000 design points evaluated during the running of the EA. The final solution is shown in Figure 5.8.

Listing 5.4: Dakota input file showing global optimization on the cantilever beam problem with an evolutionary algorithm.

```
environment
  tabular_data
    tabular_data_file 'cantilever_opt_ea.dat'
    custom_annotated header eval_id

method
  max_iterations = 100
  max_function_evaluations = 2000
  coliny_ea
    seed = 11011011
    population_size = 50
    fitness_type merit_function
    mutation_type offset_normal
    mutation_rate 1.0
    crossover_type two_point
    crossover_rate 0.0
    replacement_type chc = 10

model
  single

variables
  continuous_design = 2
    upper_bounds   4.0     4.0
    initial_point  2.5     2.5
    lower_bounds   1.0     1.0
    descriptors    'w'     't'
  continuous_state = 4
    initial_state  40000. 29.E+6 500. 1000.
    descriptors    'R'    'E'    'X'  'Y'

interface
  direct
    analysis_driver = 'mod_cantilever'

responses
  objective_functions = 1
  # constraints assumed <= 0 unless bounds given
  nonlinear_inequality_constraints = 2
  descriptors = 'area' 'stress' 'displacement'
  no_gradients
  no_hessians
```



(a) (b)

Figure 5.7: Evolutionary algorithm optimization example: 50 design points in the (a) initial and (b) final populations selected by the evolutionary algorithm.

```
<<<<< Function evaluation summary: 2007 total (2007 new, 0 duplicate)
<<<<< Best parameters          =
                     2.3547960771e+00 w
                     3.3249245714e+00 t
                     4.0000000000e+04 R
                     2.9000000000e+07 E
                     5.0000000000e+02 X
                     1.0000000000e+03 Y
<<<<< Best objective function  =
                     7.8295193371e+00
<<<<< Best constraint values   =
                    -6.8024421496e+02
                    -3.4463344891e-03
<<<<< Best data captured at function evaluation 1982
```

Figure 5.8: Dakota output showing optimal point found by a global evolutionary algorithm.

EAs are better suited to optimization problems where conventional gradient-based optimization fails, such as situations where there are multiple local optima and/or gradients are not available. In such cases, the computational expense of an EA is warranted since other optimization methods are not applicable or impractical. In many optimization problems, EAs often quickly identify promising regions of the design space where the global minimum may be located. However, an EA can be slow to converge to the optimum. For this reason, it can be an effective approach to combine the global search capabilities of an EA with the efficient local search of a gradient-based algorithm in a *hybrid optimization* strategy. In this approach, the optimization starts by using a few iterations of an EA to provide the initial search for a good region of the parameter space (low objective function and/or feasible constraints), and then it switches to a gradient-based algorithm (using the best design point found by the EA as its starting point) to perform an efficient local search for an optimum design point. More information on this hybrid approach is provided in the "Hybrid Minimization" section

of the "Advanced Methods" chapter of the Dakota User's Manual [1].

Another effective method for global optimization, especially for costlier computational models, is the Efficient Global Optimization (EGO) approach. This is discussed in the section "Efficient Global Minimization" of the Dakota User's Manual [1].

5.3 Summary and Additional Approaches

In selecting an optimization method, important considerations include the type of variables in the problem (continuous, discrete, mixed), whether a global search is needed or a local search is sufficient, and the required constraint support (unconstrained, bound constrained, or generally constrained). Less obvious, but equally important, considerations include the efficiency of convergence to an optimum (i.e., convergence rate) and the robustness of the method in the presence of challenging design space features (e.g., nonsmoothness).

Table 5.2 provides a more extensive reference for selecting from among all of Dakota's optimization methods. Here blank fields inherit the values from above. With respect to constraint support, the methods with more advanced constraint support are also applicable to the lower constraint support levels; they are listed only at their highest level of constraint support for brevity. For example, all methods listed as supporting nonlinear constraints also support bound constraints.

Because of the computational cost of running simulation models, surrogate-based optimization (SBO) methods are often used to reduce the number of actual simulation runs. In SBO, a surrogate or approximate model is automatically constructed by Dakota based on a limited number of simulation runs. The optimization is then performed on the surrogate model. Dakota has an extensive framework for managing a variety of local, multipoint, global, and hierarchical surrogates for use in optimization. Finally, sometimes there are multiple objectives that one may want to optimize simultaneously instead of a single scalar objective. In this case, one may employ multi-objective methods described in "Optimization Capabilities" in the Dakota User's Manual to either form a single composite objective, or assess the trade-off between multiple objectives directly.

Table 5.2: Detailed guidelines for selecting from among all Dakota optimization methods. Blank fields inherit the values from above.

Method Classification	**Desired Problem Characteristics**	**Applicable Methods**
Gradient-Based Local	smooth; continuous variables no constraints	optpp_cg
	smooth; continuous variables; bound constraints	dot_bfgs, dot_frcg, conmin_frcg
	smooth; continuous variables; bound constraints, linear and nonlinear constraints	npsol_sqp, nlpql_sqp, dot_mmfd, dot_slp, dot_sqp, conmin_mfd, optpp_newton, optpp_q_newton, optpp_fd_newton, weighted sums (multiobjective), pareto_set strategy (multiobjective)
Gradient-Based Global	smooth; continuous variables; bound constraints, linear and nonlinear constraints	hybrid strategy, multi_start strategy
Derivative-Free Local	nonsmooth; continuous variables; bound constraints	optpp_pds
	nonsmooth; continuous variables; bound constraints, linear and nonlinear constraints	asynch_pattern_search, coliny_cobyla, coliny_pattern_search, coliny_solis_wets, surrogate_based_local
	nonsmooth; continuous variables; discrete variables; bound constraints, nonlinear constraints	mesh_adaptive_search
Derivative-Free Global	nonsmooth; continuous variables; bound constraints	ncsu_direct
	nonsmooth; continuous variables; bound constraints, linear and nonlinear constraints	coliny_direct, efficient_global, surrogate_based_global
	nonsmooth; continuous variables, discrete variables; bound constraints, linear and nonlinear constraints	coliny_ea, soga, moga (multiobjective)

Chapter 6

Uncertainty Quantification

At a high level, uncertainty quantification (UQ) constitutes the process of characterizing input, numerical, and experimental uncertainties – consisting of both measurement errors and variability in replicate data, propagating these uncertainties through a computational model, and performing statistical or interval assessments on the resulting responses. This process determines the effect of uncertainties and assumptions on model responses or quantities of interest (QoI). In Section 6.1, we summarize techniques to propagate input uncertainties through models whereas in Section 6.2, we discuss Bayesian techniques to quantify input uncertainties.

For this discussion, inputs collectively refer to model parameters, initial conditions, boundary conditions, or exogenous forces. For models in which inputs are derived from closure, constitutive or phenomenological relations, one must employ model calibration techniques to estimate means, moments, or ultimately probability density functions (PDF) for these calibrated inputs based on experimental data or high-fidelity codes. This is often termed *inverse uncertainty quantification*. For CASL applications, inputs requiring calibration include cross-section values, closure and phenomenological parameters, and initial and boundary conditions.

The quantification of response or output uncertainties facilitates optimal design and decision making and is necessary to ensure robustness, performance or safety margins. For example, outputs specified for emergency core cooling systems include peak clad temperatures and maximum local cladding oxidation. The manner in which output statistics are employed depends on the application. For the assessment of design margins, the sample mean $\bar{x}$ and sample variance s^2 can be used to construct 2σ confidence intervals $\bar{x} \pm 2s$ whereas a predictive distribution for the output might be compared to replicate data to assess validation.

Wilks' formula constitutes a classical approach for assessing output uncertainty [36, 53]. In this coverage approach, the code is run N times for randomly selected input values chosen from expert-specified intervals. The outputs are then ranked to establish tolerance bounds for the response. This approach has the advantage that the number of required code evaluations is independent of the number of parameters. However, the resulting tolerance bounds can be overly conservative and the techniques detailed in Sections 6.1 and 6.2 yield more precise bounds or densities for inputs and outputs.

UQ is related to sensitivity analysis, detailed in Chapter 3, in that the common goal is to gain an understanding of how variations in the parameters or inputs affect the response functions of the engineering design problem. However, for UQ, some or all of the components of the parameter vector are considered to be uncertain as specified by particular probability distributions (e.g., normal, exponential, extreme value), or other uncertainty structures. By assigning specific distributional structure to the inputs, distributional structure for the outputs (i.e., response statistics)

can be inferred. UQ can thus be defined as the process of quantifying the imprecision of computed model responses or quantities of interest whereas sensitivity analysis ascertains how uncertainty in model outputs can be apportioned to uncertainties in model inputs when taken either singly or in combination over the range of input values.

6.1 Uncertainty Propagation

Whereas Dakota provides a number of options for propagating uncertainties through models, we focus on sampling and stochastic polynomial methods. The selection of these techniques to employ can be based on the following criteria:

- Sampling methods are applicable for nonsmooth and/or multi-modal response functions and general input densities including those for correlated parameters, initial conditions, boundary conditions or exogenous forces. Since sampling methods require numerous model evaluations – e.g., hundreds to millions – they necessitate that models be efficient to evaluate or the use of suitable surrogate models. Sampling methods yield response samples, which can be post-processed – e.g., using kernel density estimation (KDE) – to construct response densities.
- Stochastic polynomial methods require smooth response functions. Stochastic collocation (SC) methods are applicable for general input densities whereas nonintrusive polynomial chaos expansions (PCE) require the specification or construction of orthogonal polynomials. For normal or uniform input densities, Hermite or Legendre polynomials are employed. For densities that do not correspond to members in the Askey family of polynomials, Dakota provides the capability of employing empirical histograms to generate orthogonal polynomials. Evaluation of the quantity of interest often necessitates that inputs are mutually independent, which is generally not the case, but can often be achieved through Nataf transformations. These methods utilize Dakota sparse grid routines to provide highly efficient evaluation of response moments and, for sensitivity analysis, Sobol global sensitivity indices.

We note that Dakota provides a number of other techniques – including local and global reliability methods, interval methods, and mixed UQ algorithms – to propagate uncertainties through models. Readers are referred to the "Uncertainty Quantification Capabilities" chapter of [1] for details about these methods.

We do not differentiate between aleatoric uncertainties, which are inherent to a problem or experiments and are intrinsically probabilistic in nature, and epistemic uncertainties, which are due to lack of knowledge. We refer readers to [43] for details regarding the nature of these uncertainties and to the "Uncertainty Quantification Capabilities" chapter of [1] for a description of how Dakota algorithms accommodate these two classes of uncertainties. Further details regarding various uncertainty propagation techniques are provided in [43, 46].

6.1.1 Sampling Methods

Sampling-based methods are the most robust uncertainty techniques available, are applicable to almost all simulations, and possess rigorous error bounds. Consequently, they should be used whenever the function is relatively inexpensive to compute and adequate sampling can be performed. In the case of computationally expensive simulations, however, the number of function evaluations required by traditional techniques such as Monte Carlo and Latin hypercube sampling (LHS) quickly becomes prohibitive, especially if tail statistics are needed. We note that the issues associated with tail statistics can be mitigated through the use of importance sampling.

Alternatively, one can apply the traditional sampling techniques to a surrogate function approximating the expensive computational simulation. However, if this approach is selected, the user should be aware that it is difficult to assess the accuracy of the results. Unlike the case of surrogate-based local minimization, there is no simple pointwise calculation to verify the accuracy of the approximate results. This is due to the functional nature of uncertainty quantification; i.e., the accuracy of the surrogate over the entire parameter space needs to be considered, not just around a candidate optimum as in the case of surrogate-based local optimization. This issue especially manifests itself when trying to estimate low probability events such as the catastrophic failure of a system.

Due to the computational complexity of CASL codes, sampling methods will generally need to be applied to surrogate models rather than physics-based codes. Hence the issues associated with establishing the accuracy of the surrogate must be addressed or the accuracy of the surrogate-based sampling results verified for these codes. Techniques to establish surrogate accuracy include "leave-one-out" cross-validation for Gaussian processes [6] and the Dakota K-fold cross-validation capability, which admits "leave-one-out" as a special case. Details regarding the use of the Dakota sampling capability are provided in Section 6.1.5.

6.1.2 Stochastic Polynomial Methods

Stochastic polynomial methods comprise a second class of forward propagation techniques, also available in Dakota. The development of these techniques mirrors that of deterministic Galerkin and finite element analysis utilizing the notions of projection, collocation, orthogonality, and weak convergence [13, 14]. Rather than providing point estimates, they form an approximation to the functional relationship between random inputs and response functions, which provides a representation of output uncertainties for multi-code simulations. Expansion methods include nonintrusive polynomial chaos expansions (PCE), which employ multivariate orthogonal polynomials that are tailored to particular input probability distributions, and stochastic collocation (SC), which employs multivariate interpolation polynomials.

For certain applications, sampling-based models can be efficiently combined with stochastic polynomial methods. For example, it is often advantageous to employ Latin hypercube sampling directly or to construct a general surrogate, and then subsequently use the evaluated samples with a regression-based nonintrusive polynomial chaos expansion.

Nonintrusive Polynomial Chaos Expansions (PCE)

To motivate nonintrusive PCE methods, which for certain implementation regimes are also termed pseudo-spectral or discrete projection methods, we consider a parameter-dependent response $Y(X)$ where $X \in \Re^M$. For random inputs X, this response is represented by the truncated expansion

$$Y(X) = \sum_{j=0}^{J} \alpha_j \Psi_j(X) \tag{6.1}$$

where $\Psi_j(X)$ are polynomials that are orthogonal with respect to inner products corresponding to common probably density functions. For example, Hermite and Legendre polynomials with weights $e^{-x^2/2}$ and 1 are respectively used to represent single-variate normal and uniform distributions. As detailed in the "Stochastic Expansion Methods" chapter of the Dakota Theory Manual [3] and Chapter 10 of [43], tensored polynomials are constructed as basis functions for multivariate densities.

If we denote the density by $\rho(x)$ and note that $\Psi_0(X) = 1$, if follows that

$$E[\Psi_0(X)] = 1$$

and

$$\begin{aligned} E[\Psi_i(X)\Psi_j(X)] &= \int_\Gamma \Psi_i(x)\Psi_j(x)\rho(x)dx \\ &= \delta_{ij}\gamma_i \end{aligned}$$

where $\Gamma = [0,1]^M$ for scaled Legendre polynomials, δ_{ij} is the Kronecker delta and the normalization factor is

$$\gamma_i = E[\Psi_i^2(X)].$$

We note that γ_i can be computed analytically for each polynomial in the Askey family, which includes Hermite and Legendre polynomials.

Based on these orthogonality properties, it follows that the mean and variance of Y are

$$\begin{aligned} E[Y(X)] &= \alpha_0 \\ \mathrm{Var}[Y(X)] &= \sum_{j=1}^{J} \alpha_j^2 \gamma_j. \end{aligned} \tag{6.2}$$

Hence these values, as well as higher order moments, can be computed very efficiently once one has constructed the coefficients α_j.

For the nonintrusive PCE method, one takes the weighted inner product of (6.1) with respect to Ψ_j and enforces orthogonality to obtain

$$\alpha_j = \frac{1}{\gamma_j} \int_\Gamma Y(x)\Psi_j(x)\rho(x)dx\,.$$

Hence the determination of the coefficients α_j requires numerical quadrature over $\Gamma \subset \Re^M$. In Dakota, this is achieved using tensored Gaussian or sparse grid quadrature techniques. The evaluation of

$$\alpha_j \approx \frac{1}{\gamma_j} \sum_{i=1}^{N_q} Y(x_i)\Psi_j(x_i)\rho(x_i)w_i \tag{6.3}$$

thus employs codes nonintrusively, or as a black box, to evaluate the response at parameter values x_i.

As detailed in [3], the number of terms N_t in a polynomial chaos expansion of total order p, involving M random variables, is $N_t = \frac{(M+p)!}{M!p!}$. For high expansion orders and a large number of random variables, N_t can be huge, which can lead to large memory requirements and potential segmentation faults. To address this, the maximal expansion order should be chosen so that the number of terms does not exceed 10 times the number of training points.

Details regarding this method can be found in Chapter 10 of [43]. Alternatively, one can determine coefficients using Dakota's linear regression capabilities as detailed in the "Stochastic Expansion Methods" chapter of the Dakota Theory Manual [3]. The implementation of this technique in Dakota is illustrated in Section 6.1.5.

Stochastic Collocation (SC)

In the stochastic collocation method, one represents the response for random inputs X as

$$Y(X) = \sum_{j=1}^{J} r_j L_j(X)$$

where $r_j = Y(x_j)$ is the response value at the interpolation point x_j and $L_j(x)$ is a Lagrange polynomial. In 1-D, the Lagrange polynomial can be represented as

$$L_j(x) = \prod_{\substack{k=1 \\ k \neq j}}^{J} \frac{x - x_k}{x_j - x_k}$$

which highlights the property that $L_j(x_i) = \delta_{ij}$. For moderate parameter dimensionality, Dakota provides the capability for implementing multivariate interpolation on Smolyak sparse grids. As detailed in the "Stochastic Expansion Methods" chapter of the Dakota Theory Manual [3], Dakota also provides the capability for implementing local or global interpolating polynomials and either value-based or gradient-enhanced representations. The implementation of stochastic collocation in Dakota is illustrated in Section 6.1.5.

Table 6.1 provides a reference for choosing a Dakota propagation method or strategy based on the properties of the model.

Table 6.1: Guidelines for uncertainty propagation method selection.

Method Classification	**Desired Problem Characteristics**	**Applicable Methods**
Sampling	Nonsmooth and/or multi-modal response functions; general densities; computationally efficient models or surrogates	`sampling`
Stochastic Polynomial	Smooth response functions; can be combined with Dakota sparse grid routines	`polynomial_chaos` `stoch_collocation`

6.1.3 Verification

Verification comprises a critical component of uncertainty propagation when quantifying response variability. For many problems, one or more of the following steps can be employed to verify response uncertainties. The use of this verification strategy is illustrated in Section 6.1.5.

(i) Compare the mean, variance, skewness and kurtosis provided by noninvasive PCE and stochastic collocation.

(ii) Construct the response densities using the sampling methods discussed in Section 6.1.1 and compare the resulting moments with those computed in (i).

(iii) Employ energy statistics, as developed in [47, 48], to quantify whether or not response densities computed via sampling, noninvasive PCE and stochastic collocation correspond to the same distribution.

(iv) For linearly parameterized problems with uncorrelated Gaussian inputs $X_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, the response $Y = \sum_{i=1}^{N} a_i X_i$ is normally distributed with mean and variance

$$E(Y) = \sum_{i=1}^{N} a_i \mu_i \ , \ \text{var}(Y) = \sum_{i=1}^{N} a_i^2 \sigma_i^2. \tag{6.4}$$

These moments can be compared with those constructed in (i) and (ii).

6.1.4 Prediction Intervals

One goal when propagating uncertainties is to construct credible or prediction intervals for the model response or quantity of interest. This can be achieved using the methods of Section 6.1.1 to sample from input densities, constructed either experimentally or using the Bayesian model calibration techniques detailed in Section 6.2. The propagation of input density information alone yields credible intervals, which quantify the accuracy of the model. The simultaneous propagation of input and experimental uncertainties – e.g., using the statistical model (6.5) – yields prediction intervals, which quantify the probability of observing the next experimental measurement or numerical simulation. For example, the interval (a, b) is a 95% prediction interval if the probability that a future experimental observation y_i falls within (a, b) is at least 0.95. For this reason, prediction intervals are typically preferable to credible intervals when experimentally validating model behavior. Details regarding posterior predictive distributions can be found in [12].

6.1.5 Uncertainty Propagation: Cantilever Beam Example

We illustrate the sampling and stochastic polynomial methods for the cantilever beam example detailed in Section 2.1. Specifically, we illustrate the use of random sampling, noninvasive polynomial chaos expansions, and stochastic collocation to propagate parameter uncertainties through the model to quantify uncertainties in the output area, stress, and displacement. In all cases, parameters were assumed to have the normal distributions $R \sim \mathcal{N}(4.0E5, 4.0E6)$, $E \sim \mathcal{N}(2.9E7, 2.1025E12)$, $X \sim \mathcal{N}(5.0E2, 1.0E4)$ and $Y \sim \mathcal{N}(1.0E3, 1.0E4)$. Additional details regarding the performance of random sampling, PCE, and stochastic collocation, for this example, are provided in [46].

Random Sampling

We first illustrate the use of random sampling to construct densities for the output based on 10^6 samples from the assumed normal input distributions. The input file is shown in Listing 6.1.

An excerpt of the random sampling output is shown in Figure 6.1. First the mean, standard deviation, skewness, and kurtosis moments are calculated for each of the three response functions, as well as 95% confidence intervals for each response mean and standard deviation. Correlation matrices among all inputs and outputs are provided to determine the degree of linear relationships among variables. For those cases in which responses do not exhibit a dependence on inputs – e.g., area is not a function of R, E, X, Y – NaN is returned to indicated that there is no variability with respect to those inputs. Finally, the full output provides a listing of CDF probabilities.

Nonintrusive Polynomial Chaos Expansions (PCE)

A typical Dakota input file for performing uncertainty propagation using nonintrusive PCE is shown in Listing 6.2. In this example, we compute CDF probabilities for 17 probability levels of the cantilever beam equations. Due to the low parameter dimensionality, we select Gaussian quadrature with a total of 57 function evaluations to compute the nonintrusive PCE coefficients (6.3), using tensor product quadrature points. The tensor product generates all combinations of values from each individual dimension so it is an all-way pairing of points. We note that one would replace tensored Gaussian quadrature with sparse grid techniques for moderate parameter dimensionality; e.g., $M = 5$ to approximately $M = 40$, depending on the regularity of the modeling equations.

Once the expansion coefficients have been calculated, some statistics can be computed analytically – e.g., via (6.2) – whereas others must be evaluated numerically. For the numerical portion, the input file specifies the use of 10000 samples, which will be evaluated on the expansion to compute

Listing 6.1: UQ input for random sampling.

```
# Dakota Input File: cantilever_sampling.in

environment
  tabular_data
    tabular_data_file = 'cantilever_sampling.dat'
    custom_annotated header eval_id

method
  sampling
    sample_type random
    samples = 1000
    seed = 17
    response_levels = 1. 5. 10.
    10000. 20000. 40000.
    1. 2. 3.
model
  single

variables
  active uncertain
  continuous_design = 2
    initial_point    2.5    2.5
    descriptors      'w'    't'
  normal_uncertain = 4
    means             =  40000. 29.E+6  500. 1000.
    std_deviations    =  2000.  1.45E+6 100. 100.
    descriptors       =  'R'    'E'     'X'  'Y'

interface
  direct
    analysis_driver = 'mod_cantilever'

responses
  response_functions = 3
  descriptors = 'area' 'stress' 'displacement'
  no_gradients
  no_hessians
```

```
Statistics based on 1000 samples:

Moment-based statistics for each response function:
                           Mean           Std Dev          Skewness          Kurtosis
         area  6.2500000000e+00  0.0000000000e+00  0.0000000000e+00 -3.0000000000e+00
       stress  1.7353650406e+04  5.8085356413e+03 -6.5053618765e-03  7.3520733236e-02
  displacement  1.7161216049e+00  4.1203461736e-01  1.3498700788e-01 -1.6987570074e-02

95% confidence intervals for each response function:
                    LowerCI_Mean      UpperCI_Mean    LowerCI_StdDev    UpperCI_StdDev
         area  6.2500000000e+00  6.2500000000e+00  0.0000000000e+00  0.0000000000e+00
       stress  1.6993203553e+04  1.7714097260e+04  5.5646488747e+03  6.0749462045e+03
  displacement  1.6905529262e+00  1.7416902836e+00  3.9473425169e-01  4.3093273234e-01

Simple Correlation Matrix among all inputs and outputs:
             R            E            X            Y         area       stress   displacement
     R  1.00000e+00
     E -1.09891e-03  1.00000e+00
     X -1.89011e-03  3.84943e-04  1.00000e+00
     Y  2.28006e-03  1.47231e-03 -1.28899e-04  1.00000e+00
  area          nan          nan          nan          nan      nan
stress -3.46903e-01  1.62695e-03  6.50987e-01  6.74942e-01      nan  1.00000e+00
 disp.  1.99216e-03 -4.87018e-01  3.80836e-01  7.81233e-01      nan  7.74982e-01  1.00000e+00

Partial Correlation Matrix between input and output:
                     area       stress displacement
           R          nan -1.00000e+00  5.02670e-03
           E          nan          nan -9.87430e-01
           X          nan  1.00000e+00  9.79611e-01
           Y          nan  1.00000e+00  9.95042e-01

Simple Rank Correlation Matrix among all inputs and outputs:
              R            E            X            Y         area       stress   displacement
     R   1.00000e+00
     E   1.29417e-02  1.00000e+00
     X  -1.02171e-02  3.68951e-03  1.00000e+00
     Y   3.49896e-03  5.36687e-03  4.26256e-03  1.00000e+00
  area          nan          nan          nan          nan       nan          nan
stress  -3.31948e-01 -7.20001e-08  6.39440e-01  6.52457e-01       nan  1.00000e+00
  disp  -1.33517e-03 -4.74628e-01  3.68797e-01  7.57213e-01       nan  7.57396e-01  1.00000e+00

Partial Rank Correlation Matrix between input and output:
                  area          stress    displacement
           R       nan    -7.97285e-01     2.39053e-02
           E       nan    -6.40293e-03    -8.86932e-01
           X       nan     9.31104e-01     8.26712e-01
           Y       nan     9.34429e-01     9.49696e-01
```

Figure 6.1: Excerpt of UQ output for random sampling.

the CDF probabilities. We summarize in Figure 6.2 excerpts of the output. The full output lists a summary of the PCE coefficients, which reproduce the function for a Hermite polynomial basis. The analytic statistics for mean, standard deviation, and covariance are then presented for each of the response functions: area, stress, and displacement. Finally, we note the numerical results for the CDF probabilities based on 10^5 samples performed on the expansion. For example, approximately 50% of the displacement samples are determined to be less than or equal to 1.709 inches.

Listing 6.2: UQ input for nonintrusive polynomial chaos expansions.

```
# Dakota Input File: cantilever_uq_pce.in

environment
    tabular_data
        tabular_data_file = 'cantilever_uq_pce.dat'
    graphics

method
  polynomial_chaos
    sparse_grid_level = 2 #non_nested
    sample_type lhs
    seed 12347
    samples = 10000
    num_probability_levels = 0 17 17
    probability_levels =
      .001 .01 .05 .1 .15 .2 .3 .4 .5 .6 .7 .8 .85 .9 .95 .99 .999
      .001 .01 .05 .1 .15 .2 .3 .4 .5 .6 .7 .8 .85 .9 .95 .99 .999
    cumulative distribution

model
  single

variables
  active uncertain
  continuous_design = 2
    initial_point    2.5    2.5
    descriptors      'w'    't'
  normal_uncertain = 4
    means             =  40000. 29.E+6  500. 1000.
    std_deviations    =  2000.  1.45E+6 100. 100.
    descriptors       =  'R'    'E'     'X'  'Y'

interface
  direct
    analysis_driver = 'mod_cantilever'

responses
  response_functions = 3
  descriptors = 'area' 'stress' 'displacement'
  no_gradients
  no_hessians
```

```
Statistics derived analytically from polynomial expansion:
Moment-based statistics for each response function:
                            Mean           Std Dev          Skewness          Kurtosis
stress
  expansion:    1.7600000000e+04  5.7871581973e+03
  numerical:    1.7600000000e+04  5.7871581973e+03  9.4100742175e-15  6.2172489379e-15
displacement
  expansion:    1.7201243431e+00  4.0644795983e-01
  numerical:    1.7201243431e+00  4.0644787032e-01  1.5009952217e-01  4.9005496977e-02
Covariance matrix for response functions:
[[  7.3191614098e-30  5.3786836096e-13  6.5282177146e-17
    5.3786836096e-13  3.3491200000e+07  1.8163897036e+03
    6.5282177146e-17  1.8163897036e+03  1.6519994405e-01 ]]

Cumulative Distribution Function (CDF) for stress:
     Response Level  Probability Level  Reliability Index  General Rel Index
     --------------  -----------------  -----------------  -----------------
   2.4921421856e+02   1.0000000000e-03
   4.1489075797e+03   1.0000000000e-02
   7.9708753041e+03   5.0000000000e-02
   1.0090342657e+04   1.0000000000e-01
   1.1589780322e+04   1.5000000000e-01
   1.2731567123e+04   2.0000000000e-01
   1.4564078343e+04   3.0000000000e-01
   1.6151010310e+04   4.0000000000e-01
   1.7689441098e+04   5.0000000000e-01
   1.9129203866e+04   6.0000000000e-01
   2.0683233939e+04   7.0000000000e-01
   2.2457356004e+04   8.0000000000e-01
   2.3589089220e+04   8.5000000000e-01
   2.4920875151e+04   9.0000000000e-01
   2.7044322788e+04   9.5000000000e-01
   3.0752664401e+04   9.9000000000e-01
   3.5331778223e+04   9.9900000000e-01
Cumulative Distribution Function (CDF) for displacement:
     Response Level  Probability Level  Reliability Index  General Rel Index
     --------------  -----------------  -----------------  -----------------
   5.8392829293e-01   1.0000000000e-03
   8.2796204947e-01   1.0000000000e-02
   1.0598405267e+00   5.0000000000e-02
   1.2097152423e+00   1.0000000000e-01
   1.2968601525e+00   1.5000000000e-01
   1.3747889667e+00   2.0000000000e-01
   1.4999245316e+00   3.0000000000e-01
   1.6076201989e+00   4.0000000000e-01
   1.7093348267e+00   5.0000000000e-01
   1.8118345186e+00   6.0000000000e-01
   1.9241773807e+00   7.0000000000e-01
   2.0596369386e+00   8.0000000000e-01
   2.1417150361e+00   8.5000000000e-01
   2.2453314112e+00   9.0000000000e-01
   2.3964502080e+00   9.5000000000e-01
   2.7290918315e+00   9.9000000000e-01
   3.0882954345e+00   9.9900000000e-01
```

Figure 6.2: Excerpt of UQ output for nonintrusive polynomial chaos expansion.

Stochastic Collocation

Here we illustrate the use of stochastic collocation built on an anisotropic sparse grid defined from numerically-generated orthogonal polynomials; see Chapters 10 and 11 of [43] for details regarding stochastic collocation. The input file is shown in Listing 6.3. In this example, we again compute CDF probabilities of stress and displacement for varying response levels. This example requires 233 function evaluations to compute the interpolating polynomials used for stochastic collocation.

Listing 6.3: UQ input for stochastic collocation.

```
# Dakota Input File: cantilever_uq_sc.in

environment
  tabular_data
    tabular_data_file = 'cantilever_uq_sc.dat'


method
  stoch_collocation
    sparse_grid_level = 3
    #dimension_preference = 2 1
    samples = 10000 seed = 12347 rng rnum2
    response_levels = 1. 5. 10. 15. 20. 50.
    1. 10. 5000. 10000. 50000. 100000.
    .1 1. 2. 3. 5. 10.
    variance_based_decomp #interaction_order = 1
    output silent

variables
  #lognormal_uncertain = 2
   #     means             =  1.   1.
    #std_deviations    =  0.5  0.5
    #descriptors       = 'x1' 'x2'

  active uncertain
  continuous_design = 2
    initial_point    2.5    2.5
    descriptors      'w'    't'
  normal_uncertain = 4
    means             =  40000. 29.E+6  500. 1000.
    std_deviations    =  2000.  1.45E+6 100. 100.
    descriptors       =  'R'    'E'     'X'  'Y'

interface
  direct
    analysis_driver = 'mod_cantilever'

responses
  response_functions = 3
  descriptors = 'area' 'stress' 'displacement'
        no_gradients
  no_hessians
```

Once the expansion coefficients have been calculated, some statistics are available analytically and others must be evaluated numerically. For the numerical portion, the input file specifies the use of 10000 samples, which will be evaluated on the expansion to compute the CDF probabilities. We

summarize in Figure 6.3 excerpts from the output. We first note the moment statistics for mean, standard deviation, skewness, and kurtosis computed by numerical integration (see the "Analytic Moments" section in the "Stochastic Expansion Methods" chapter in the Dakota Theory Manual [3]), where the numerical row corresponds to integration using the original response values and the expansion row corresponds to integration using values from the interpolant. The response covariance and global sensitivity indices (Sobol indices) are presented next. This example shows that for stress, the variables R, X, and Y all play significant roles, but that the interactions between them are relatively negligible. For displacement, E, X, and Y have significant influence, while the interactions between E and X, E and Y, X and Y, and the three way interaction have significantly less effect. Finally, in the full output, we see the numerical results for the CDF probabilities based on 10000 samples performed on the expansion.

Method Verification

We illustrate here the verification techniques detailed in Section 6.1.3. We compile in Tables 6.2 and 6.3 the first four moments provided by random sampling, noninvasive PCE and stochastic collocation. Additionally, since stress is a linear function of the normally distributed inputs X, Y and R, the relations (6.2) can be used to compute analytic values for the stress moments.

We note that the moments for each of the three methods are in close agreement, with the exception of the third and fourth moments in the stress output. Since the stress is linearly dependent on the parameters, it is expected that the PCE and collocation methods should produce exact results for these two moments, neglecting rounding error. By comparison, the skewness and kurtosis, obtained with random sampling, are non-negligible but have values that are still five to seven orders of magnitude smaller than the sample mean and variance.

The stress density constructed by random sampling is compared in Figure 6.4(a) with normal densities whose mean and variance (6.2) are constructed using coefficients α_j determined by nonintrusive PCE or stochastic collocation. The displacement density constructed by random sampling is plotted in Figure 6.4(b). We cannot compare to Gaussian representations with mean and variance

Stress				
	Mean	Std. Deviation	Skewness	Kurtosis
Analytic	1.760e+04	5.787e+03	–	–
Sampling	1.735e+04	5.809e+03	-6.505e-03	7.352e-02
PCE	1.760e+04	5.787e+03	9.410e-15	6.217e-15
Collocation	1.760e+04	5.787e+03	-1.842e-14	5.773e-15

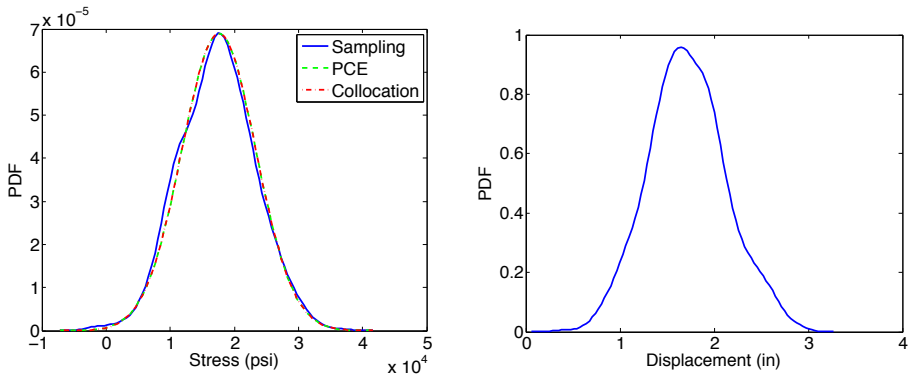
Table 6.2: Comparison of moments for stress from random sampling, polynomial chaos expansion, and stochastic collocation methods.

Displacement				
	Mean	Std. Deviation	Skewness	Kurtosis
Sampling	1.716	4.120e-01	1.350e-01	-1.700e-02
PCE	1.720	4.064e-01	1.501e-01	4.901e-02
Collocation	1.720	4.064e-01	1.504e-01	6.878e-02

Table 6.3: Comparison of moments for displacement from random sampling, polynomial chaos expansion, and stochastic collocation methods.

```
Statistics derived analytically from polynomial expansion:

Moment-based statistics for each response function:

                           Mean           Std Dev          Skewness          Kurtosis
stress
  expansion:    1.7600000000e+04  5.7871581973e+03 -1.3819173712e-14  5.3290705182e-15
  numerical:    1.7600000000e+04  5.7871581973e+03 -1.8416777990e-14  5.7731597281e-15
displacement
  expansion:    1.7201241681e+00  4.0644979045e-01  1.5036497320e-01  6.8782204635e-02
  numerical:    1.7201241681e+00  4.0644979045e-01  1.5036497320e-01  6.8782204635e-02

Covariance matrix for response functions:
[[  7.8886090522e-31 -3.2350298255e-26 -6.5675038652e-31
   -3.2350298255e-26  3.3491200000e+07  1.8163966854e+03
   -6.5675038652e-31  1.8163966854e+03  1.6520143216e-01 ]]

Global sensitivity indices for each response function:

stress Sobol indices:
                                  Main             Total
                       1.1943435888e-01  1.1943435888e-01 R
                      -4.9831897744e-14  1.3881742943e-13 E
                       4.4028282056e-01  4.4028282056e-01 X
                       4.4028282056e-01  4.4028282056e-01 Y
                            Interaction
                      -2.3136238238e-14 R E
                       7.1188425348e-15 R X
                       7.1188425348e-14 E X
                      -8.1866689150e-14 R Y
                       1.2635945499e-13 E Y
                      -9.0765242319e-14 X Y
                       1.6017395703e-14 R E X
                       2.4915948872e-14 R E Y
                       1.8508990590e-13 R X Y
                      -2.6695659506e-14 E X Y
displacement Sobol indices:
                                  Main             Total
                       3.0376298835e-13 -1.3978473800e-13 R
                       2.4260687756e-01  2.4452448915e-01 E
                       1.5243149082e-01  1.5378097334e-01 X
                       6.0208388530e-01  6.0457471845e-01 Y
                            Interaction
                       2.4193512347e-13 R E
                      -2.5537596366e-13 R X
                       3.8691316858e-04 E X
                      -8.0376224352e-13 R Y
                       1.5282637946e-03 E Y
                       9.6013473508e-04 X Y
                      -1.6666641839e-13 R E X
                       1.9623626681e-13 R E Y
                       3.4408550893e-13 R X Y
                       2.4346209458e-06 E X Y
```

Figure 6.3: Excerpt of UQ output for stochastic collocation.



(a) (b)

Figure 6.4: (a) Stress density constructed by random sampling and normal densities with mean and variance (6.2) computed using coefficients α_j determined by nonintrusive PCE and stochastic collocation. (b) Displacement density constructed by random sampling.

computed using nonintrusive PCE or collocation coefficients since the nonlinear displacement dependence on inputs yields a non-Gaussian output as demonstrated by the magnitude of the skewness and kurtosis coefficients in Table 6.3. However, we could employ the nonintrusive PCE or collocation representations as surrogate models, which could subsequently be used with random sampling to more efficiently construct the non-Gaussian displacement density.

The comparison in Figure 6.4(a), obtained by sampling, nonintrusive PCE, and stochastic collocation provides a qualitative comparison of the methods. To quantify whether the probability density functions correspond to the same distribution, we compute energy statistics as presented in [47, 48]. We take the null hypothesis H_0 to be the statement that the three PDF sample sets are from the same distribution. We reject H_0, at a $100(1-\alpha)\%$ confidence level, if the p-value computed using resampled replicates is less than α. We employed $n = 1000$ samples to compute each PDF and $M = 499$ bootstrap samples. For these values the test statistic is $T = 1.3 \times 10^4$ whereas the p-value is 0.7. Because this value is greater than $\alpha = 0.01$, we do not reject the null hypothesis that the PDF samples are from the same distribution.

6.2 Bayesian Model Calibration

We noted in the introduction to this chapter that uncertainty quantification is broadly comprised of two steps: (i) quantification of uncertainties associated with models, inputs and experiments, and (ii) propagation of these uncertainties through models to quantify uncertainties in responses or quantities of interest. Here inputs refer to model parameters, initial conditions, boundary conditions or exogenous forces. We discuss here techniques to calibrate inputs derived from closure, constitutive or phenomenological relations for which values derived from fundamental principles are lacking.

The deterministic calibration techniques detailed in Chapter 5 provide point estimates for calibration inputs, but generally no measure of uncertainty. An exception is gradient-based methods, which produce asymptotic 95% confidence intervals for each input. The usefulness of these intervals is tied to the validity of asymptotic assumptions applied to the problem at hand. As noted in the introduction to this chapter, approaches such as Wilks' formula employ uniform input distributions, which are generally based on expert opinion. These intervals are often based on qualitative, rather than quantitative, knowledge and hence they are typically conservative.

Bayesian inference provides a framework for probabilistic model calibration based on the assump-

tion that calibration inputs are random variables having associated PDFs. These PDFs quantify both the support, or admissible parameter values, and the plausibility of each admissible parameter value. In Bayesian model calibration, one employs a likelihood, which incorporates measured data and computed model information, to update prior density information to obtain a more accurate posterior parameter density, which is consistent with experimental uncertainties.

Input densities or bounds, constructed in this manner, are tighter and contain more information than uniform densities constructed solely to bound potential input values. Propagation of these input densities using the sampling, nonintrusive polynomial chaos expansions, or stochastic collocation techniques of Section 6.1 will provide reduced response uncertainties and hence tighter robustness, performance or safety margins. For example, these densities could be employed in Wilks' formula to construct tighter tolerance bounds than those obtained using conservative, non-inference based input densities.

We summarize pertinent details required for implementation of the methods and refer readers to [21, 43] for additional examples and details regarding the theory and algorithms.

6.2.1 Direct Implementation of Bayes' Relation

To set notation, we consider a random calibration parameter vector $\Theta = [\Theta_1, \ldots, \Theta_M]$ with the realization $\theta = [\theta_1, \ldots, \theta_M]$. We consider the statistical model

$$D_i = y_i(\Theta) + \varepsilon_i \ , \ i = 1, \ldots, N \tag{6.5}$$

where $D = [D_1, \ldots, D_N]$ denotes unobserved (random) data, $y(\Theta) = [y_1(\Theta), \ldots, y_N(\Theta)]$ is the parameter-dependent model, and $\varepsilon = [\varepsilon_1, \ldots, \varepsilon_N]$ is a random vector, which represents experimental and model errors. Throughout this discussion, we assume that ε_i are independently and identically distributed (i.i.d.) and $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ where the experimental error variance σ^2 is determined experimentally or estimated through the inference procedure. We note that the model may additionally depend on spatial or temporal independent variables – e.g., $y_i = y(x_i, \Theta)$ or $y_i = y(t_i, \Theta)$ – but we simplify notation by suppressing these latter dependencies since model calibration focuses on uncertain parameters.

In Bayesian inference, one employs Bayes' relation

$$\pi(\theta|d) = \frac{\mathcal{L}(\theta; d)\pi_0(\theta)}{\displaystyle\int_{\Re^M} \mathcal{L}(\theta; d)\pi_0(\theta)d\theta} \tag{6.6}$$

for observed data d to update a prior density $\pi_0(\theta)$, using the likelihood $\mathcal{L}(\theta; d)$, to obtain a more informative posterior density $\pi(\theta|d)$.

Prior Density. The prior density $\pi_0(\theta)$ incorporates any knowledge that one has about parameters prior to obtaining observations d. This could come from previous similar experiments or analysis regarding similar models. If prior knowledge is of questionable accuracy, it is better to use a noninformative prior, which is often taken as an improper uniform density posed on the parameter support. For example, one would employ $\pi_0(\theta) = \chi_{(0,\infty)}(\theta)$ for positive parameters, where $\chi_{(0,\infty)}(\theta)$ is the characteristic function having a value of 1 for $\theta \in (0, \infty)$ and 0 for $\theta \in (-\infty, 0]$. In analyses assuming random experimental error variance σ^2, a standard noninformative prior for σ^2 is the Jeffreys prior $\pi_0(\sigma^2) \propto (1/\sigma^2)$.

Likelihood Function. The likelihood $\mathcal{L}(\theta; d)$ incorporates information provided by the samples and model and constitutes the mechanism through which data informs the posterior density. The

likelihood can be interpreted as quantifying the probability of obtaining the observations d for a given value θ of the parameter Θ. The likelihood can generally be written as

$$\mathcal{L}(\theta; d) = f[y(\theta) - d]$$

where the function f can be constructed to emphasize specific relations between the model and data. For the statistical model (6.5) with i.i.d. errors ε_i that are normally distributed, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, one employs the likelihood relation

$$\begin{aligned} \mathcal{L}(\theta; d) = & \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-(d_i - y_i(\theta))^2/2\sigma^2} \\ = & \frac{1}{(2\pi\sigma^2)^{N/2}} e^{-SS_q/2\sigma^2} \end{aligned} \tag{6.7}$$

where

$$SS_q = \sum_{i=1}^{N} [d_i - y_i(\theta)]^2$$

denotes the sum of squares error. We note that the likelihood and prior density must be specified by users when employing sampling-based Bayesian algorithms.

Posterior Density. For small parameter dimensions M and fixed or estimated σ^2, the posterior density can be constructed by employing quadrature rules to approximate the denominator of (6.6). If we let w^i and ζ^i denote the quadrature weights and points and assume a noninformative prior, the posterior can be approximated by

$$\pi(\theta|d) \approx \frac{1}{\sum_{i=1}^{N_q} e^{-(SS_{\zeta^i} - SS_q)/2\sigma^2} w^i}. \tag{6.8}$$

The algebraic reformulation of the sum of squares in the denominator is made to avoid numerical $0/0$ errors. For $M = 1$ through roughly 4, one can employ tensored Gaussian quadrature relations whereas sparse grid techniques can be employed for moderate dimensionality. We refer to this as *direct Bayesian calibration.*

6.2.2 Sampling Based Metropolis Algorithms

The difficulties associated with approximating the denominator of (6.6) or constructing marginal posterior densities, for moderate to large dimensional parameter spaces, can be partiality circumvented by employing Markov chain Monte Carlo (MCMC) techniques. The goal with these algorithms is to construct sampling-based chains whose stationary distribution is the posterior distribution. The capabilities currently provided in Dakota are: Delayed Rejection Adaptive Metropolis (DRAM) (and variants) via the and QUESO (Quantification of Uncertainty for Estimation, Simulation, and Optimization) library and DiffeRential Evolution Adaptive Metropolis (DREAM). An additional capability, Gaussian Process Models for Simulation Analysis (GPMSA), is under development in QUESO and will be deployed through a future Dakota release.

Delayed Rejection Adaptive Metropolis (DRAM)

In Metropolis algorithms, parameters are sampled using a proposal function that reflects, to the degree possible, the geometry of the unknown posterior density. For example, one can propose candidate samples $\theta^* \sim \mathcal{N}(\theta^{k-1}, V)$, where θ^{k-1} is the previous chain element and V is the proposal

covariance matrix. Proposed candidates are rejected or accepted with a probability that reflects the degree to which candidates increase the likelihood. The goal is to construct a chain whose stationary distribution is the posterior density.

DRAM is a variation of the Metropolis algorithm whose robustness is improved in two ways. First, adaptation allows the algorithm to update the proposal covariance matrix to reflect accepted candidates. In this manner, information acquired about the posterior distribution through accepted chain candidates is used to update the proposal distribution. Secondly, delayed rejection provides a mechanism for efficiently constructing alternative candidates when the current candidate is rejected. In combination, these two mechanisms provide the algorithm with substantial robustness and efficiency [16]. We note that parallel versions of DRAM have recently been developed [44].

DiffeRential Evolution Adaptive Metropolis (DREAM)

There are regimes in which DRAM algorithms are often inefficient. These include problems in which posterior densities are multi-modal, highly complex, or have heavy tails. For these cases, the single DRAM chain will be slow to traverse the posterior which can significantly diminish its efficiency. Moreover, the computational overhead associated with complex models can preclude the construction of burned-in single chains whereas one can often compute shorter parallel chains using massively parallel architectures.

DREAM algorithms can circumvent some of these limitations. In these algorithms, candidates are randomly generated using differential evolution algorithms. These algorithms are inherently parallel and have the advantage that chains can learn from each other. Details are provided in [52, 51].

Gaussian Process Models for Simulation Analysis (GPMSA)

GPMSA provides additional capability for Bayesian calibration. A key part of GPMSA is the construction of a surrogate model or emulator from simulation runs collected at various settings of input parameters; see Chapter 4 for details regarding the construction of surrogate models including Gaussian process representations. The emulator is a statistical model of the system response, and it is used to incorporate the observational data to improve system predictions and constrain or calibrate the unknown parameters. The GPMSA code draws heavily on the theory developed in the seminal Bayesian calibration paper by Kennedy and O'Hagan [25]. The particular approach under development in QUESO is described in [20]. GPMSA uses Gaussian process models in the emulation, but for functional responses the emulator is actually a set of basis functions (e.g., from a singular value decomposition) which have GPs as the coefficients. One major difference between GPMSA and the QUESO implementation in Dakota is that the QUESO implementation does not have an explicit "discrepancy" function δ which models the difference between the simulation and the observational data results in addition to the error term ε, but GPMSA has a sophisticated model for the discrepancy term.

Table 6.4 provides a reference for choosing a Bayesian model calibration method based on the properties of the model.

6.2.3 Model Calibration and Surrogate Models

Dakota provides various capabilities for combining Bayesian model calibration algorithms with surrogate models. The first is to employ a surrogate model or emulator when constructing chains and posterior densities for inputs. For computationally intense codes, this will be necessary to acquire the 10^3 to 10^5 model solutions required to burn-in chains and obtain a statistically relevant

Table 6.4: Guidelines for Bayesian method selection.

Method Classification	Desired Problem Characteristics	Applicable Methods
DRAM	Unimodal or weakly multi-modal posterior densities; computationally efficient models or surrogates	`bayes_calibration queso`
DREAM	Multi-modal, complex or heavy tailed posterior densities; inherently parallel simulation codes	`bayes_calibration dream`
GPMSA	Bayesian calibration using a Gaussian process emulator; can accommodate certain model discrepancy relations	n/a

number of chain elements. Upon completion, the QUESO GPMSA package will provide one alternative for Bayesian calibration using a Gaussian process emulator. Alternatively, the techniques of Chapter 4 can be used to construct a surrogate model, which is then employed in Dakota-QUESO DRAM or Dakota DREAM for Bayesian model calibration. One can apply Bayesian methods to a general surrogate by using a `model_pointer` to point the Bayesian routine to a model of type `surrogate`. For the nonintrusive PCE or stochastic collocation surrogates, discussed in Section 6.1, or Gaussian process surrogates, one can shortcut this process by specifying the emulator as one of `gaussian_process, pce`, etc.

Once input distributions have been constructed, one can employ a surrogate model to implement the sampling methods discussed in Section 6.1.1 for quantities of interest that may not have been used for calibration. Alternatively, one can employ the stochastic polynomial methods of Section 6.1.2 to propagate uncertainties. We note that these comprise a form of interpolation or regression-based surrogate models, which complement the kriging and Gaussian process-based methods detailed in Chapter 4.

6.2.4 Verification

We summarize here a general framework for verifying model calibration results for CASL codes implemented in VERA via Dakota. This framework is generally applicable to codes with nonlinear parameter dependencies and experimental or synthetic data.

(i) Test algorithms using a linearly-parameterized model where analytic uncertainty relations can be computed. Whereas this is not generally possible for CASL codes with nonlinear parameter dependencies, code verification in this manner provides a first step for verifying the capabilities of the model calibration framework and it may be used in certain nearly linear operating regimes.

(ii) Compare to direct numerical implementation of Bayes' formula (6.6) for small to moderate input or parameter dimensions M; e.g., $M \leq 20$ to 30. For the likelihood relation (6.7) and a noninformative prior $\pi_0(\theta)$, this involves the evaluation of the relation (6.8). This also comprises code verification.

(iii) Compare to other packages that implement DRAM to perform code-to-code verification.

(iv) Compare DRAM and DREAM results. These comparisons can be quantified using the energy statistics tests detailed in [47, 48].

(v) Compare to sampling distributions provided by frequentist analysis. Whereas this approach can guide verification, is must be used with care since the underlying assumptions for frequentist and Bayesian inference differ significantly; see Chapter 4 of [43]. For example, asymptotic analysis often yields Gaussian sampling distributions which will obviously be inaccurate if the true distribution is highly non-Gaussian.

(vi) Check the convergence of the algorithms by increasing the number of quadrature points used in (6.8) or number of iterations in DRAM or DREAM chains to establish solution verification.

6.2.5 Synthetic Data

One often employs synthetic data when testing model calibration algorithms since it provides a regime in which errors are constructed, and hence known, and it can be employed when experimental data is not readily available. We illustrate the construction of synthetic data for the statistical model (6.5) but note that the procedure may vary for other statistical models.

For a nominal input $\tilde{\theta} = [\tilde{\theta}_1, \ldots, \tilde{\theta}_M]$, one calculates a nominal model response $y(\tilde{\theta})$. For a specified variance σ^2, one then generates realizations $\epsilon_1, \ldots, \epsilon_N$ from a normal distribution $\mathcal{N}(0, \sigma^2)$, which yields the synthetic data

$$d_i = y(\tilde{\theta}) + \epsilon_i \ , \ i = 1, \ldots, N.$$

For multiple responses, the standard deviation σ is typically scaled by the magnitude of each response in the manner illustrated in the cantilever beam example of Section 6.2.6.

6.2.6 Bayesian Calibration Examples

We illustrate here the performance and verification of the Dakota Bayesian model calibration packages for the cantilever beam example of Section 6.2.4 and linear verification example of Section 2.2 and Appendix A. The use of the algorithms for COBRA-TF is illustrated in Section 7.3.

Cantilever Beam

We employ the cantilever beam example of Section 2.1 to illustrate the implementation of the DRAM and DREAM algorithms for Bayesian model calibration along with the verification framework summarized in Section 6.2.4. The posterior parameter densities constructed in this manner can subsequently be employed with the uncertainty propagation techniques detailed in Section 6.1 to quantify response uncertainties.

Case I. We consider first the case in which the Young's modulus E and width w are considered unknown, the remaining parameters and inputs t, R, L, D_0, X and Y are assumed known and fixed, and data are taken to be displacement and stress measurements d and s. To construct synthetic data using the techniques of Section 6.2.5, based on the assumption of independent and identically distributed (i.i.d.) observation errors $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, we compute the nominal displacement and stress values

$$\tilde{d} = 0.3086 \, , \tilde{s} = 2.6667 \times 10^3$$

using the fixed and nominal input values in Table 6.5. The standard deviations are taken to be

$$\sigma_d = 0.1 \cdot \tilde{d} \ , \ \sigma_s = 0.1 \cdot \tilde{s}$$

t	R	L	D_0	X	Y	$\tilde{E}$	$\tilde{w}$
3	4×10^4	100	2.2535	500	100	2.85×10^7	2.5

Table 6.5: Known values for t, R, L, D_0, X, Y and nominal values $\tilde{E}$ and $\tilde{w}$.

d ($\times 10^{-1}$)	3.2075	2.7005	2.7939	2.8578	2.9298	2.9875	3.0903	2.1515	2.9454	3.4700
s ($\times 10^3$)	2.3829	3.0943	2.9959	2.6054	2.2650	2.5481	2.6251	2.7403	2.5970	2.7849

Table 6.6: Synthetic displacement and stress data employed for Bayesian model calibration.

which yields the synthetic data compiled in Table 6.6 when observation errors ε_{d_i} and ε_{s_i} are drawn from normal distributions $\mathcal{N}(0, \sigma_d^2)$ and $\mathcal{N}(0, \sigma_s^2)$.

We first employ the discretized Bayes relation (6.8) to directly construct marginal posterior densities for E and w. Due to the simplicity of the algebraic model, we employ a tensored trapezoid quadrature rule, which yields the convergence results compiled in Table 6.7. For more computationally intensive models and codes, one would employ tensored Gaussian routines for low parameter dimensions for $M = 1$ to approximately 6 and sparse grid techniques for moderate dimensionality of M up to 30 or 40 where the upper limit depends on the regularity of the likelihood.

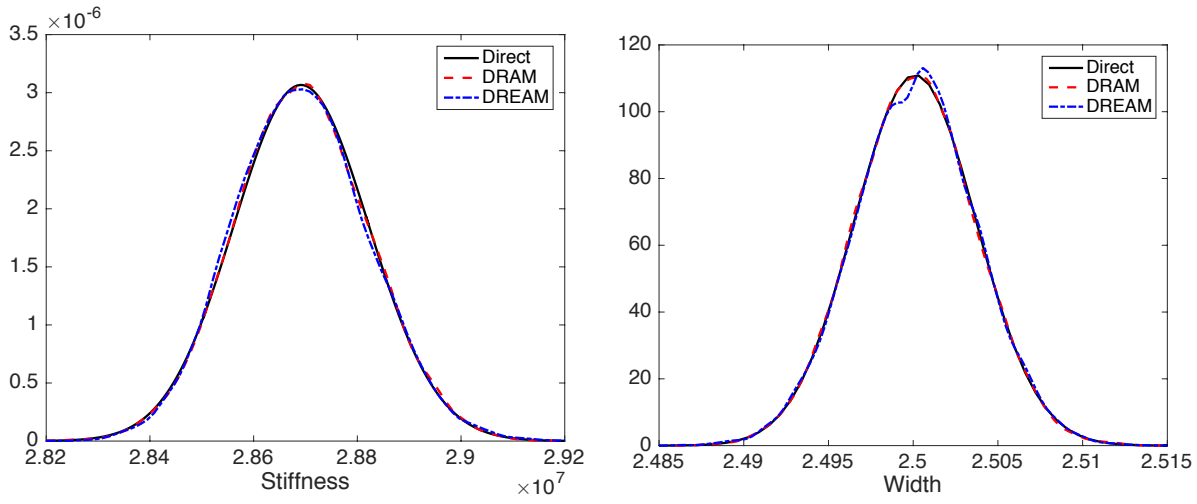
The direct results are compared with posterior densities constructed using the Dakota-QUESO DRAM and Dakota DREAM algorithms in Figure 6.5. The corresponding input decks are provided in Listings 6.4 and 6.5. The file `dakota_cant_withsigma.dat` contains the calibration data of Table 6.6 along with nominal observation error variances corresponding to each observation. Each row represents an individual experiment, and the associated observations are placed in the first set of columns followed by the nominal variances in the same order. The posterior means and standard deviations are compiled in Table 6.8. The joint sample points plotted in Figure 6.6 demonstrate that E and w are correlated but identifiable. For the DRAM algorithm, we constructed a chain of length 60,000 and saved the last 50,000 elements to ensure burn-in. For DREAM, we employed 10 chains of length 6,000 and employed the last 5000 entries when computing statistics and marginal

N_q	μ_E	σ_E	μ_w	σ_w
40	2.8751e+07	2.6685e+05	2.5024	1.2589e-02
80	2.8688e+07	1.6430e+05	2.5004	6.3164e-03
160	2.8693e+07	1.3087e+05	2.5001	3.6669e-03
320	2.8693e+07	1.3011e+05	2.5001	3.5940e-03

Table 6.7: Convergence of the direct numerical Bayes relation.

	μ_E	σ_E	μ_w	σ_w
Direct	2.8693e+07	1.3011e+05	2.5001	3.5940e-03
DRAM	2.8693e+07	1.2911e+05	2.5001	3.5699e-03
DREAM	2.8692e+07	1.2945e+05	2.5001	3.5994e-03

Table 6.8: Posterior means and standard deviations provided by the direct, DRAM and DREAM algorithms.



(a) (b)

Figure 6.5: Marginal posterior densities for (a) E and (b) w generated through direct solution of Bayes' relation (6.8), Dakota-QUESO DRAM and Dakota DREAM.

and joint densities. In both cases, we employed noninformative priors.

From Figure 6.5, we note that the direct, DRAM and DREAM results match very closely for the stiffness E. The width plot demonstrates that in some cases, the marginal density computed using DREAM can differ slightly from that computed using DRAM. This is a very qualitative comparison since kernel density estimates (KDE), used to compute the marginal densities from the sampled chains, have a smoothing effect on the distribution. We note that the DREAM chains also exhibit some variability for different seed values.

To quantify the comparison between the DRAM and DREAM chains, we employ the energy statistics detailed in [47, 48] and illustrated in Section 6.1. Here the null hypothesis H_0 is that DRAM and DREAM are sampling from the same distribution. We thinned the chains to $n = 5000$ to ensure that the samples are uncorrelated and employed $M = 499$ resampled replicates to compute p-values. We first tested the chains for E and w individually. In this case, the p-value for E is 0.024 whereas it is 0.142 for w. Because both p-values are greater than $\alpha = 0.01$, this supports the conclusion that the DRAM and DREAM samples for E and w are from the same distributions. We then computed a p-value of 0.022 for the joint samples E and w. For this case, we again have insufficient evidence to reject the null hypothesis H_0.

To apply energy statistics to the direct method, one could construct a spline representation, based on the computed posterior values, and sample from this representation. Due to the very close qualitative comparisons with the DRAM marginal distributions, we do not illustrate this method for quantitatively comparing sampled-based methods to the direct implementation.

In summary, we recommend that, when possible, at least two of the techniques be compared to verify the accuracy of the inference procedure.

Case II. Secondly, we consider the case when E, t and w are considered uncertain and synthetic data is taken to be displacement, stress and area measurements generated in a manner analogous to Case I. The joint sample points constructed using Dakota-QUESO DRAM are plotted in Figure 6.7. The nearly single-valued relation between t and w indicates that these parameters are essentially nonidentifiable, which is a manifestation of nonunique input-output maps. This is consistent with the observation that the product $A = w \cdot t$ appears in the displacement, stress and area relations.

Listing 6.4: Input for Dakota-QUESO DRAM.

```
# DAKOTA INPUT FILE - cantilever_bayes_dram.in

method
  bayes_calibration queso
    logit_transform
    dram
    proposal_covariance prior
    chain_samples = 60000
    seed = 348
    export_chain_points_file = 'cant_dram.txt'

variables
  uniform_uncertain 2
    initial_point 2.85e7 2.5
    upper_bounds  1.e8 10.0
    lower_bounds  1.e6  0.1
    descriptors 'E' 'w'
  continuous_state 4
    initial_state 3 40000 500 1000
    descriptors 't' 'R' 'X' 'Y'

interface
  system
    analysis_driver = 'mod_cantilever'

responses
  calibration_terms = 2
  descriptors = 'stress' 'displacement'
  calibration_data_file = 'dakota_cant_withsigma.dat' freeform
  num_experiments = 10
  variance_type = 'scalar'

  no_gradients
  no_hessians
```

Listing 6.5: Input for Dakota DREAM.

```
# DAKOTA INPUT FILE - cantilever_bayes_dream.in

method
  bayes_calibration dream
    chain_samples = 60000
    chains = 10
    seed = 348
    export_chain_points_file = 'cant_dream.txt'

variables
  uniform_uncertain 2
    initial_point 2.85e7 2.5
    upper_bounds  1.e8 10.0
    lower_bounds  1.e6  0.1
    descriptors 'E' 'w'
  continuous_state 4
    initial_state 3 40000 500 1000
    descriptors 't' 'R' 'X' 'Y'

interface
  system
    analysis_driver = 'mod_cantilever'

responses
  calibration_terms = 2
  descriptors = 'stress' 'displacement'
  calibration_data_file = 'dakota_cant_withsigma.dat' freeform
  num_experiments = 10
  variance_type = 'scalar'

  no_gradients
  no_hessians
```

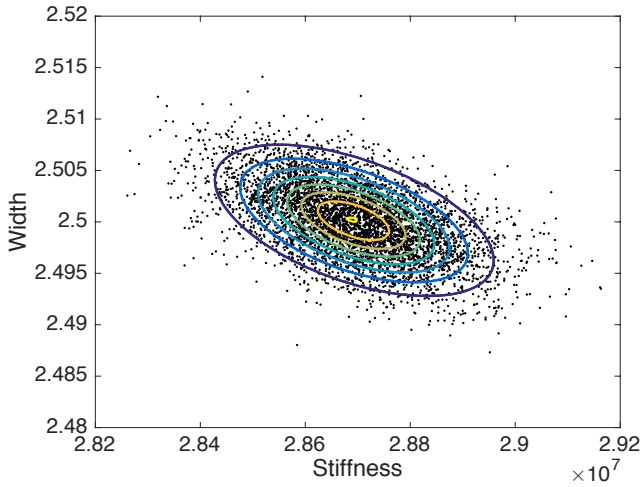


Figure 6.6: Joint posterior sample points for E and w constructed using the direct method (contours) and DRAM (points).

We note that for this example, DREAM more accurately quantifies the nearly single-valued relation between E and w and E and t as compared with DRAM. The implementation of Bayesian calibration techniques for nonidentifiable parameter sets will generally be problematic unless informative prior specification is provided. If such prior information is not available, parameter selection based on global sensitivity analysis or reduced order modeling techniques should be employed to determine the set of identifiable or influential parameters. Details regarding parameter selection techniques can be found in Chapter 6 of [43].

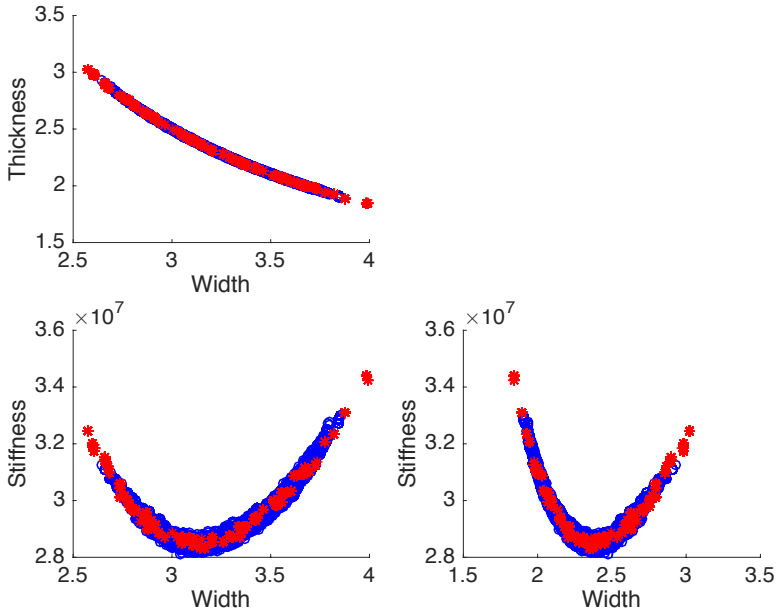


Figure 6.7: Joint posterior sample points for E, w and t using DRAM (o) and DREAM ($*$).

General Linear Model Verification Tests

We illustrate here aspects of the general linear model verification test suite described in Chapter 2 and Appendix A. This constitutes step (i) in the verification framework detailed in Section 6.2.4. All results were obtained using the QUESO implementation of DRAM within Dakota. The input files mentioned in this subsection can be found in `examples/LinearModel`.

Three examples of QUESO DRAM verification are examined. The first two demonstrate Case 2 scenarios with the "no correlation" and "AR(1) correlation" specifications, respectively. The third represents Case 1 with the AR(1) correlation model.

A simple linear regression model, $g(x) = (1, x_1, x_2, \ldots, x_M)^T$, is specified for the three verification examples treated in this section. The dimension of β is therefore $N_\beta = M + 1$. Our examples assume $M = 2$ covariate dimensions and therefore $N_\beta = 3$. For the purpose of generating calibration data using the process described in Appendix A.1, the nominal regression parameter vector β_0 was set to $(0.2, -0.3, 0.1)$, and the nominal precision (inverse variance) λ_0 and correlation parameter ϕ_0 in the AR(1) correlation model were set to 400 and 0.8, respectively. The regression matrix G used by all three examples is given as follows,

$$G = \begin{pmatrix} 1 & -1.3134 & -0.230852 \\ 1 & 0.865439 & -0.708232 \\ 1 & -1.24733 & 0.443547 \\ 1 & 0.598521 & -1.47469 \\ 1 & -1.22409 & -0.347091 \end{pmatrix},$$

which is contained in the file `g.in`.

Data of sample size $N = 5$ were generated via (2.4) as described in Appendix A.1. The largest number of unknown parameters, $N_\beta + 1 = 4$ encountered in the Case 2 examples, is therefore exceeded by the amount of available data. Data generated for the "no correlation" and "AR(1) correlation" specifications are provided in Table 6.9.

Table 6.9: Calibration data for the "no correlation" and "AR(1) correlation" specifications.

No correlation	AR(1) correlation
0.597953	0.553559
−0.123317	−0.178668
0.556432	0.602756
−0.178478	−0.149112
0.541521	0.540218

Both Case 2 examples assume the noninformative prior for β and the Gamma prior for λ with $a = 64$ and $b = 4/25$. In the AR(1) correlation scenarios, ϕ_0 is fixed at 0.8 in the calibration analysis. The Case 1 example adopts the informative prior for β having $\mu_0 = \mathbf{0}_3$, $\lambda_0 = 400$, and $\Sigma = 0.25 * \mathcal{I}_3$ for $\mathbf{0}_{N_\beta}$ the $N_\beta \times 1$ vector of zeros and $\mathcal{I}_{N_\beta}$ the $N_\beta \times N_\beta$ identity matrix.

Example 1. We first illustrate Case 2 with data generated under the "no correlation" specification. Listing 6.6 shows the Dakota input file used to generate DRAM samples from the joint posterior distribution of the regression and variance parameters.

There are several notable features in Listing 6.6. Line 7 of the input file indicates that 260,000 iterations (accepted chain samples) of DRAM are conducted. In the analysis of results, the first

Listing 6.6: Input for the first stochastic verification example.

```
environment
  tabular_data
    tabular_data_file = 'verif1_cal_tabular.dat'

method
  bayes_calibration queso
    chain_samples = 260000
    seed = 503
    logit_transform
    export_chain_points_file = 'verif1_cal_mcmc.dat'
    dram
    proposal_covariance
      diagonal values 1.  1.  1.
    calibrate_error_multipliers one
      hyperprior_alphas = 64
      hyperprior_betas  = 64
    output verbose

variables
  uniform_uncertain 3
    initial_point   0.    0.    0.
    upper_bounds  100.  100.  100.
    lower_bounds -100. -100. -100.
    descriptors   'q1'  'q2'  'q3'

interface
  fork
    analysis_driver = 'simulator_script'
    work_directory named 'workdir'
      directory_tag
      file_save
      copy_files = 'lm.template' 'g.in'
      parameters_file = 'params.in'
      results_file    = 'results.out'

responses
  calibration_terms = 5
    calibration_data_file 'y_1.dat' freeform
    num_experiments = 1
    variance_type = 'scalar'

  no_gradients
  no_hessians
```

10,000 DRAM samples were discarded as burn-in, and the remaining 250,000 samples were thinned by taking every 25th sample, resulting in a total of 10,000 samples used. This ensured that the samples ultimately used for verification purposes were approximately uncorrelated as required by the adopted statistical methods. In line 9, the `logit_transform` option transforms all bounded parameters to an unbounded domain for the purpose of facilitating the generation of legitimate proposals in the DRAM algorithm.

Line 20 of the input file specifies the prior distribution of the regression parameters β as independently uniform. Dakota does not currently allow specification of the flat noninformative prior for β, so this is approximately accomplished by specifying uniform distributions on domains much wider than the anticipated ranges of values for these parameters, as done in lines 22 and 23 here.

Line 40 of the input file instructs Dakota to adopt the `'scalar'` structure for the variance of the j-th response in the i-th experiment σ^2_{ij},

$$\sigma^2_{ij} = m_{ij}\sigma^2_{n,ij}\,,$$

where the multipliers m_{ij} are independent random variables assigned Inverse Gamma prior distributions and the nominal variances $\sigma^2_{n,ij}$ are provided in the input data file `y_1.dat` (Line 38). Line 14 further restricts this specification to a common multiplier $m_{ij} = m$ across experiments and responses. In this example, since data from only one experiment is available, the variance of the j-th response is therefore given by

$$\sigma^2_j = m\sigma^2_{n,j}\,,$$

where the multiplier m is a random variable assigned an Inverse Gamma prior distribution. The format of the input data file requires one row per experiment and one column per response, with optional additional columns containing the nominal variances $\{\sigma^2_{n,ij}\}$ of each response. In this example, `y_1.dat` therefore contains one row with the first five columns giving the observed responses (first column of Table 6.9) and the second five columns giving the nominal variances of each response, where $\sigma^2_{n,j} = 0.0025 = 1/\lambda_0$ for $j = 1, 2, \ldots, 5$. In other words, a common variance $\sigma^2 = m\sigma^2_n$ with $\sigma^2_n = 0.0025$ is assumed across all responses here.

Lines 15 and 16 of the input file specify hyperparameter values for the shape (`hyperprior_alphas`) and rate (`hyperprior_betas`) parameters of the Inverse Gamma prior distribution for m, namely 64 and 64, respectively. Note that this variance specification is equivalent to the common precision $\lambda = 1/\sigma^2$ of Appendix A having a Gamma prior distribution with $a = 64$ and $b = 4/25$, as indicated earlier in the introduction.

Lines 12 and 13 of the input file instruct Dakota to adopt specified values for the diagonal entries of the initial covariance matrix used in the Gaussian proposal density of the DRAM algorithm. Normally these can be adopted from the prior specification of the β parameters in the `variables` block, but this default was overridden here as the noninformative nature of the β prior in this example renders a proposal covariance matrix that is "too large," negatively impacting the performance of the DRAM algorithm.

The `interface` block of the input file instructs Dakota to run `simulator_script` to generate model output. This shell script first calls `dprepro`, a Perl program distributed with Dakota that sets up an input deck for each run by replacing tags in the templated input deck `lm.template` with the actual parameter values utilized. The next step is to run the code providing the input to output map, in this case a Python program called `lm.py` supplied by the user. Finally the output of this run is postprocessed into the results needed for Dakota consumption.

Figure 6.8 presents comparisons of the marginal posterior densities for the three regression parameters and the precision sampled via DRAM with the analytical solution derived from Appendix A.1. In all cases, a kernel density estimator using a Gaussian kernel is used to approximate

the marginal posterior densities from the DRAM samples. It is observed visually that the DRAM algorithm implemented in QUESO produces marginal posterior distributions that closely match analytical results. This is confirmed by conducting the energy test of Appendix A.3, which seeks evidence against the hypothesis that QUESO DRAM is sampling from a distribution equal to the analytical posterior distribution. Sample sets of size 100 from both distributions were used to conduct the test, which produced a p-value of 0.661 suggesting insufficient evidence against the desired outcome of equal joint distributions. The 100 QUESO DRAM samples were obtained by taking every 100th sample from the thinned set of 10,000 samples described previously.

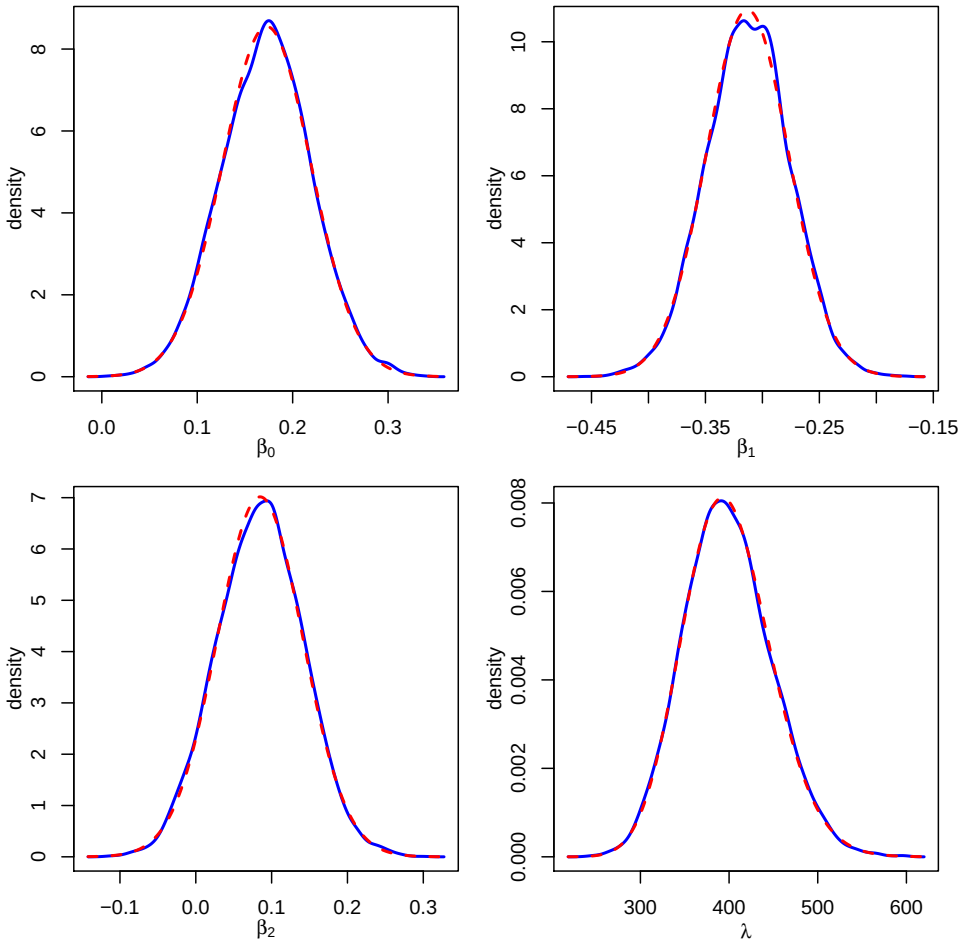


Figure 6.8: Marginal posterior distributions of regression parameters β and error precision λ computed from DRAM samples (solid blue) and analytical results (dashed red).

Example 2. We next illustrate Case 2 with data generated under the "AR(1) correlation" specification. Listing 6.7 shows the Dakota input file used to generate DRAM samples from the joint posterior distribution of the regression and variance parameters.

This example is very similar to the first example, with the main difference being the covariance specification. In particular, the first example assumes the individual responses are mutually independently distributed, while this example assumes they are correlated. Line 43 of the input file instructs Dakota to adopt the `'matrix'` structure for the covariance Σ_i of the responses associated

Listing 6.7: Input for the second stochastic verification example.

```
environment
  tabular_data
    tabular_data_file = 'verif2_cal_tabular.dat'

method
  bayes_calibration queso
    chain_samples = 260000
    seed = 603
    logit_transform
    export_chain_points_file = 'verif2_cal_mcmc.dat'
    dram
    proposal_covariance
      diagonal values 1.  1.  1.
    calibrate_error_multipliers one
      hyperprior_alphas = 64
      hyperprior_betas  = 64
    output verbose

variables
  uniform_uncertain 3
    initial_point   0.    0.    0.
    upper_bounds  100.  100.  100.
    lower_bounds -100. -100. -100.
    descriptors   'q1'  'q2'  'q3'

interface
  fork
    analysis_driver = 'simulator_script'
    work_directory named 'workdir'
      directory_tag
      file_save
      copy_files = 'lm.template' 'g.in'
      parameters_file = 'params.in'
      results_file    = 'results.out'

responses
  descriptors 'y_2'
  calibration_terms = 1
    field_calibration_terms = 1
      lengths = 5
    calibration_data
      num_experiments = 1
      variance_type = 'matrix'

  no_gradients
  no_hessians
```

with the i-th experiment,

$$\Sigma_i = m_i \Sigma_{n,i} \,,$$

where the multipliers m_i are independent random variables assigned Inverse Gamma prior distributions and the nominal covariance matrices $\Sigma_{n,i}$ are input to Dakota from files having the following nomenclature: `<descriptor>.i.sigma`. Here, the descriptors are identified in line 37 of the input file, while the `i` indicates the experiment number. As before, line 14 further restricts this specification to a common multiplier $m_i = m$ across experiments,

$$\Sigma_i = m \Sigma_{n,i} \,,$$

where the multiplier m is a random variable assigned an Inverse Gamma prior distribution. In this example, as indicated on line 42 there is only one experiment which consists of five responses (Line 40). Therefore, the nominal covariance matrix $\Sigma_{n,1}$ must be input from the file `y_2.1.sigma`, given by

```
0.0025   0.002   0.0016 0.00128 0.001024
0.002    0.0025  0.002  0.0016  0.00128
0.0016   0.002   0.0025 0.002   0.0016
0.00128  0.0016  0.002  0.0025  0.002
0.001024 0.00128 0.0016 0.002   0.0025
```

which must have dimensions conforming to the number of responses corresponding to the associated experiment (5×5). The matrix $\Sigma_{n,1}$ above was constructed from the AR(1) correlation function described in Appendix A.2 using $\phi_0 = 0.8$, followed by scaling to obtain a nominal marginal variance of $0.0025 = 1/\lambda_0$ for each of the five responses as in the previous example.

The data are input to Dakota as column vectors from files having the following nomenclature: `<descriptor>.i.dat`. In this example, the second column of Table 6.9 must be saved to the file `y_2.1.dat` (absent any column header) prior to running Dakota.

Figure 6.9 presents comparisons of the marginal posterior densities for the three regression parameters and the precision sampled via DRAM with the analytical solution derived from Appendix A.1. As in the previous example, it is observed visually that the DRAM algorithm implemented in QUESO produces marginal posterior distributions that closely match analytical results. This is confirmed by the energy test, which produced a p-value of 0.503 supporting the conclusion of equal joint distributions.

It is not yet possible to work with covariance structures in Dakota having a more complicated parameterization than that considered above (for example, $\Sigma_{n,i} = R(\phi_i)$ with $R(\cdot)$ having the AR(1) structure of Appendix A.2 and ϕ_i random is not currently permitted).

Example 3. We conclude by illustrating Case 1 with data generated under the "AR(1) correlation" specification. Listing 6.8 shows the Dakota input file used to generate DRAM samples from the joint posterior distribution of the regression parameters.

This example adapts the previous example by assuming an informative prior for the regression parameters β and by fixing the multiplier in the covariance structure to be $m = 1$. The `variables` block starting on line 16 of the input file specifies a Gaussian prior distribution for β having means 0, variances 0.01, and no pairwise correlation among the elements of β. The nominal covariance matrix of the responses `y_2.1.sigma` is carried over from the previous example, as is the data vector `y_2.1.dat`. The fixed multiplier $m = 1$ is specified by simply removing the block

```
calibrate_error_multipliers one
  hyperprior_alphas = 64
  hyperprior_betas  = 64
```

Listing 6.8: Input for the third stochastic verification example.

```
environment
  tabular_data
    tabular_data_file = 'verif3_cal_tabular.dat'

method
  bayes_calibration queso
    chain_samples = 260000
    seed = 703
    logit_transform
    export_chain_points_file = 'verif3_cal_mcmc.dat'
    dram
    proposal_covariance
      diagonal values 1.  1.  1.
    output verbose

variables
  normal_uncertain 3
    initial_point   0.   0.   0.
    means           0.   0.   0.
    std_deviations  0.1  0.1  0.1
    descriptors    'q1' 'q2' 'q3'

interface
  fork
    analysis_driver = 'simulator_script'
    work_directory named 'workdir'
      directory_tag
      file_save
      copy_files = 'lm.template' 'g.in'
      parameters_file = 'params.in'
      results_file    = 'results.out'

responses
  descriptors 'y_2'
  calibration_terms = 1
    field_calibration_terms = 1
      lengths = 5
    calibration_data
      num_experiments = 1
      variance_type = 'matrix'

  no_gradients
  no_hessians
```

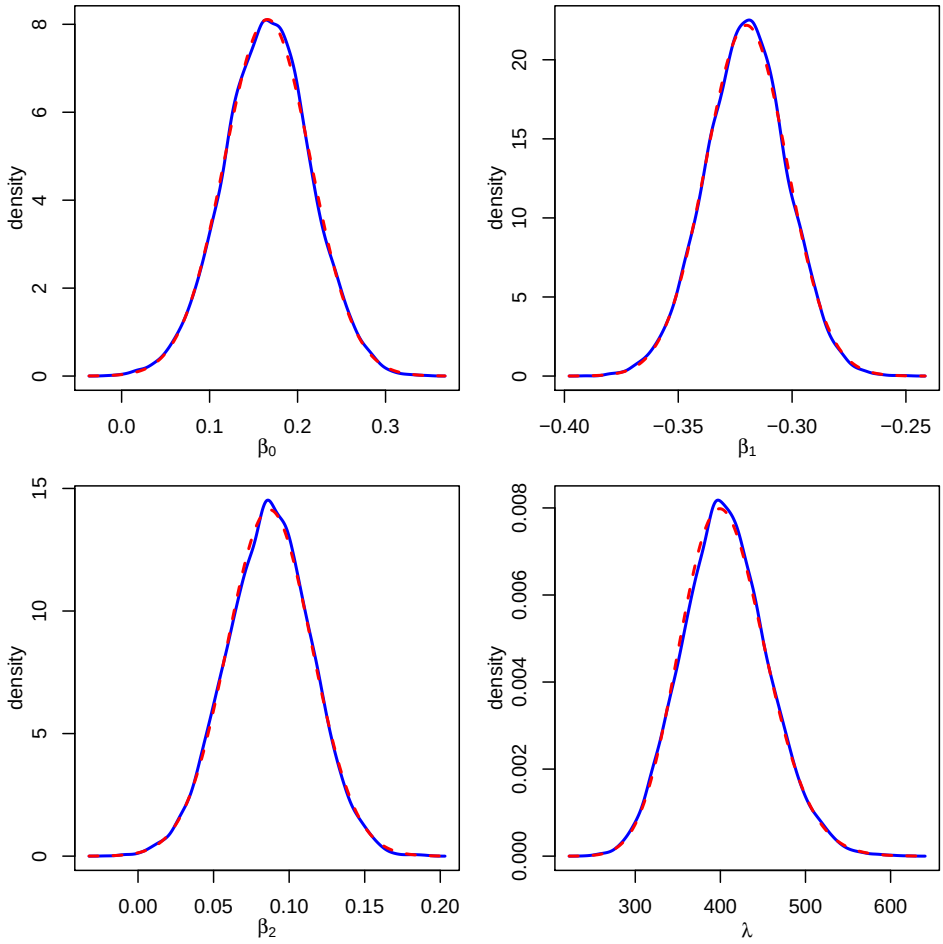


Figure 6.9: Marginal posterior distributions of regression parameters β and error precision λ computed from DRAM samples (solid blue) and analytical results (dashed red).

from Listing 6.7.

Figure 6.10 presents comparisons of the marginal posterior densities for the three regression parameters sampled via DRAM with the analytical solution derived from Appendix A.1. As in the previous examples, it is observed visually that the DRAM algorithm implemented in QUESO produces marginal posterior distributions that closely match analytical results. The energy test again confirms this result, as the p-value of 0.455 supports the conclusion of equal joint distributions.

Dakota does not currently facilitate the informative joint prior distribution for (β, λ) adopted by Case 2 in Appendix A.1. Hence, verification of informative prior specifications for β must currently be conducted under Case 1.

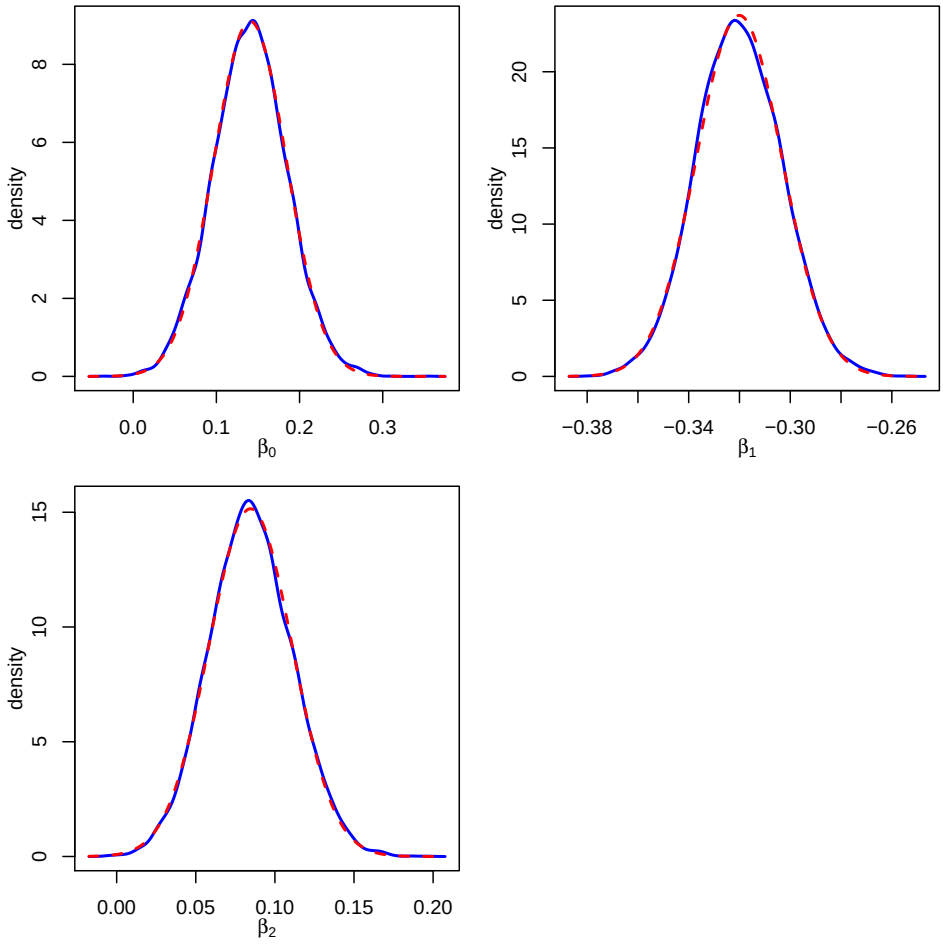


Figure 6.10: Marginal posterior distributions of regression parameters β computed from DRAM samples (solid blue) and analytical results (dashed red).

Chapter 7

COBRA-TF VUQ Studies

This chapter demonstrates the use of Dakota to complete an overall VUQ process for CASL Progression Problem 6, simulated with COBRA-TF, as described in Section 2.3. The workflow demonstration includes the following Dakota studies:

1. Initial **centered parameter studies** to exercise the COBRA-TF model with two coupled physics scenarios and verify the Dakota/COBRA-TF interfaces, resulting in adding and removing some parameters from the admissible set. (Section 7.1)

2. **Sensitivity analysis** using parameter study, LHS and Morris methods to identify the most important of 33 parameters. Initial LHS studies revealed code robustness issues under joint variation, resulting in adjusting the range of one parameter. Screening based on these studies resulted in three significant parameters. Another LHS study with 30 samples was conducted over these parameters and used in subsequent activities. (Section 7.2)

3. **Deterministic and Bayesian calibration** to estimate the values or distributions of the three significant parameters using synthetic data. The deterministic calibration demonstrates gradient-based local calibration using the COBRA-TF simulation model directly, while the Bayesian calibration uses a surrogate model constructed from the 50 LHS samples. (Section 7.3)

4. **Surrogate construction** and validation to assess the quality and applicability of a response surface model. (Section 7.3.2)

The total Progression Problem 6 parameter set for consideration is indicated in Table 2.6 in Section 2.3. The two parameters marked with an asterisk in this table, `ql` and `qv`, were initially included to assess the effects of heat transfer across the fuel pin surface into the channel liquid and vapor phases, respectively. However, it was quickly discovered that for the steady-state Progression Problem 6, perturbing either of these produces an inherent thermal imbalance precluding any steady thermal behavior. Accordingly, these parameters were excluded from the following studies. For all studies in this chapter, the total pressure drop through the fuel rod assembly (here indicated by `TotalPressure`) was used as the quantity of interest.

7.1 Initial Parameter Studies with Two Power Distributions

Two initial centered parameter studies were conducted to verify the Dakota/COBRA-TF interface, assess code robustness, and generate initial results. Both employed a Dakota centered parameter

study (Section 3.2.1) over 33 parameters for which the shift values were zero; e.g. $ka_p = 0$, and the scaling values were allowed to vary by ±5% around unity; e.g. $k_p = 1.0 \pm 5\%$ in increments of 1%. The two studies differ in the power distribution input to the thermal hydraulics code. The first uses a uniform value specified via input, while the second uses an axially varying power distribution representing a converged steady-state solution from a previous full simulation of Progression Problem 6. A spatially-varying power distribution is provided by a file, "ss_power.txt," which if present in the run directory gets used instead of the uniform power specification. The second study represents a parameter sensitivity study performed around the actual solution to the neutronics component of the problem.

The initial centered parameter studies each involve 33 parameters evaluated at 10 pertubed values in addition to the baseline (nominal) evaluation. This amounts to 331 total runs for each study. Dakota provides a concurrent execution facility, which for these studies, enables 60 independent runs to execute simultaneously on the the CASL fissile machines (e.g. `james007`, `boris natasha` or `anasova`). Each run requires between 5 and 7 minutes so that each centered parameter study completes in just under 1 hour assuming available capacity on the machine.

Table 7.1 summarizes the results for the uniform power distribution, and Table 7.2 summarizes results for the spatially varying power distribution. All values are reported as percentages of the difference in total pressure drop across the fuel assembly compared to each respective baseline value representing unperturbed parameter values. The difference in the baseline values can be considered as a measure of the sensitivity of total pressure drop to the spatial power distribution. All parameters were perturbed in the same manner; e.g., the scaling coefficient was adjusted -5% to +5% in increments of 1%. These initial results demonstrate the iterative exploratory process of conducting Dakota studies on models, so detailed discussion is omitted. The results in the remainder of this chapter are based on a spatially uniform power distribution.

7.2 COBRA-TF Sensitivity Studies

Initial studies reduced the admissible parameter set to 33 key parameters, for which the VUQ process is demonstrated in this section. The first step in sensitivity analysis is to perform a centered parameter study to assess the effect of individual parameters on the simulation response. Then we conduct Dakota analyses that jointly vary the parameters to better assess global sensitvities for complex models. The Latin hypercube and Morris methods complement the parameter study results to screen the parameter set.

7.2.1 Centered Parameter Study

Dakota Input: A Dakota input file `dakota_centered_33.in` for a 33 variable centered parameter study is shown in Listing 7.1. The `method` section (line 5) prescribes the study with five each positive and negative parameter steps of 0.01. The 33 COBRA-TF parameters are specified in the variables section, with an `initial_point` of 1.0 (lines 17-21), indicating nominal input values for the simulation. The `interface` (line 30) specifies use of the `dakota-vera-analysis` driver (line 34), which implements the Dakota/COBRA-TF interface descibed in Appendix B. This analysis workflow accepts values of the 33 parameters from Dakota, runs the simulation, and extracts the desired response metric, `TotalPressure`. Dakota's `work_directory` feature (line 40) will cloister each COBRA-TF simulation in a separate working directory to permit concurrent model evaluations.

Results and Discussion: Figure 7.1 displays data from the Dakota-generated tabular data file `dakota_centered_33.dat`, revealing the univariate effects of each parameter on the `TotalPressure` response. The parameters `k_cd`, `k_xkwlx` and to a lesser degree `k_rodqq` have significant effect, and

Table 7.1: Percent difference of total pressure drop compared to the baseline value of 1.17231 bar using uniform power distribution input to the COBRA-TF thermal hydraulics code.

	percent perturbation in parameter									
parameter	−5%	−4%	−3%	−2%	−1%	+1%	+2%	+3%	+4%	+5%
cd	−2.02	−1.62	−1.21	−0.81	−0.4	0.41	0.81	1.21	1.62	2.02
cdfb	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
cond	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
eta	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
gama	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
qliht	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
qradd	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
qradv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
qvapl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
rodqq	−0.26	−0.21	−0.16	−0.11	−0.05	0.05	0.11	0.16	0.22	0.27
sdent	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
sent	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
spht s	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmasg	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmasl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmasv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmome	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmoml	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmomv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tnrgl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tnrgv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
wkr	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkge	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkle	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkvls	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkwew	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkwlw	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkwlx	−3.83	−3.07	−2.3	−1.53	−0.77	0.77	1.53	2.3	3.07	3.83
xkwvw	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkwvx	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

there is a strong linear relationship between each of them and `TotalPressure`. The parameter `k_tmoml` induces a small change in the response, and the remainder of the parameters have zero effect. These sensitivity results make physical sense based on assessment of the test problem. The total pressure drop should depend strongly on the loss coefficient (`k_cd`) and wall friction (`k_xkwlx`) in the dominant flow direction, with a minor dependency on the turbulent mixing between channels

(`k_tmoml`). The dependence on the externally supplied heat rate (`k_rodqq`) indicates the possibility of boiling, but this has not been explored in depth.

Table 7.2: Percent difference of total pressure drop compared to the baseline value of 1.17561 bar using a power distribution from a previous steady-state neutronics solution to Progression Problem 6.

	percent perturbation in parameter									
parameter	−5%	−4%	−3%	−2%	−1%	+1%	+2%	+3%	+4%	+5%
cd	−2.03	−1.62	−1.22	−0.81	−0.41	0.41	0.81	1.22	1.62	2.03
cdfb	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
cond	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
eta	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
gama	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
qliht	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
qradd	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
qradv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
qvapl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
rodqq	−0.28	−0.23	−0.17	−0.11	−0.06	0.06	0.12	0.17	0.23	0.29
sdent	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
sent	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
spht s	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmasg	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmasl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmasv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmome	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmoml	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tmomv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tnrgl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tnrgv	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
wkr	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xk	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkes	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkge	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkl	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkle	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkvls	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkwew	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkwlw	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkwlx	−3.84	−3.07	−2.31	−1.54	−0.77	0.77	1.54	2.3	3.07	3.84
xkwvw	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
xkwvx	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Listing 7.1: Dakota input file for centered parameter study, with ±5% variation in each of 33 parameters.

```
environment
  tabular_graphics_data
    tabular_graphics_file 'dakota_centered_33.dat'

method
  # 11 total evaluations over range [0.95,1.05]
  centered_parameter_study
  step_vector           0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
                        0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
                        0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
                        0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
                        0.01
  steps_per_variable  5

variables
  continuous_design     33
  initial_point         1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
                        1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
                        1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
                        1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
                        1.0
  descriptors           'k_eta' 'k_gama' 'k_sent' 'k_sdent' 'k_tmasv'
                        'k_tmasl' 'k_tmasg' 'k_tmomv' 'k_tmome'
                        'k_tmoml' 'k_xk' 'k_xkes' 'k_xkge' 'k_xkl'
                        'k_xkle' 'k_xkvls' 'k_xkwvw' 'k_xkwlw' 'k_xkwew'
                        'k_qvapl' 'k_tnrgv' 'k_tnrgl' 'k_rodqq' 'k_qradd'
                        'k_qradv' 'k_qliht' 'k_sphts' 'k_cond' 'k_xkwvx'
                        'k_xkwlx' 'k_cd' 'k_cdfb' 'k_wkr'

interface
  fork
    asynchronous
      evaluation_concurrency = 24
    analysis_driver = 'dakota-vera-analysis'
    # extract TotalPressure metric (length 1)
    analysis_components = 'TotPress'
    parameters_file = 'params.in'
    results_file    = 'results.out'
    failure_capture recover NaN
    work_directory
    directory_tag
    named 'workdir'
      file_save  directory_save

responses
  num_response_functions = 1
    descriptors
      'TotalPressure'
  no_gradients
  no_hessians
```

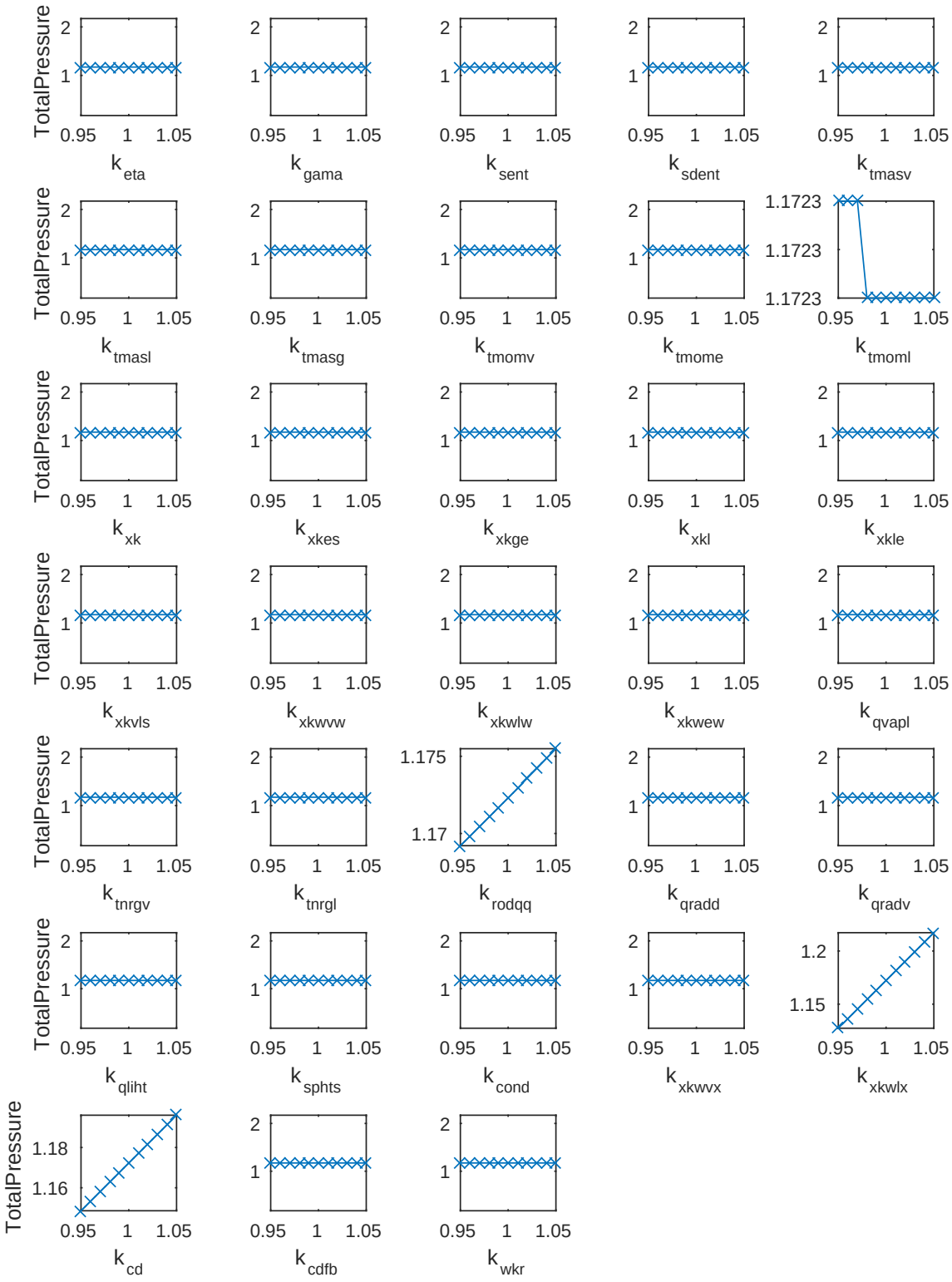


Figure 7.1: Sensitivities of `TotalPressure` to each of 33 variables varied over ±5% in a centered parameter study. Note that most parameters had identically zero variation and `k_tmoml` has only very slight variation.

7.2.2 Latin hypercube sampling studies

Latin hypercube sampling for sensitivity analysis is described in Section 3.2.3. Since all the COBRA-TF parameters affect model form in the solution, they are taken to have uniform distributions on the interval $[-10\%, 10\%]$.

Dakota Input: To change the Dakota input from a centered parameter study to a LHS study, the method and variables specifications change. The method block now prescribes a Latin hypercube sampling study. The number of samples is specified to be $N = 10 \times (M =) 33$ parameters, or $N = 330$:

```
method
  sampling
    sample_type lhs
    samples = 330
    seed = 52983
```

The variables section changes to use uncertain variables with a uniform distribution on $[0.9, 1.1]$:

```
uniform_uncertain =   33
upper_bounds          1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1
                      1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1
                      1.1 1.1 1.1 1.1 1.1 1.1 1.1
lower_bounds          0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9
                      0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9
                      0.9 0.9 0.9 0.9 0.9 0.9 0.9
```

The full Dakota input file `dakota_lhs_33.in` is shown in Listing 7.2.

Results and Discussion: Relevant outputs generated by Dakota include the correlation coefficients in the screen output `dakota_lhs_33.out`, and tabulated data in `dakota_lhs_33.dat`. The portion of the Dakota console output with the partial correlation coefficients is shown in Figure 7.2. The output is easier to comprehend when plotted with Matlab, as shown in the bar graph in Figure 7.3. The partial correlation coefficients near 1.0 indicate that there is a strong linear correlation between `k_cd` and `TotalPressure` and between `k_xkwlx` and `TotalPressure`, consistent with physical intuition that total pressure drop should depend linearly on both the axial grid spacer loss coefficient and the axial wall friction coefficient. Besides the relatively slight positive and linear correlation with `k_rodqq` no other parameters are strongly significant by this measure (greater than 0.4), although several are greater than 0.1 and could be considered for inclusion in follow-on analyses based on assessment of their interaction with other parameters or nonlinear behavior.

Figure 7.4 displays scatter plots, generated with Matlab, from the Dakota-generated tabular data file. Each plot shows the overall relationship between each parameter and `TotalPressure`, with the additional vertical variation being due to the other parameters not plotted. The red lines are a linear regression on the displayed data, indicating the strength of the linear parameter response relationship, again strongest for `k_cd` and `k_xkwlx`. There is no distinguishable input/output trend for the other variables, and no patterns in the scatter cloud to suggest concern about strong nonlinear or interaction effects.

Listing 7.2: Dakota input file for Latin hypercube sampling-based sensitivity analysis study with 330 samples and uniform input distributions.

```
environment
  tabular_graphics_data
    tabular_graphics_file 'dakota_lhs_33.dat'

method
  sampling
    sample_type lhs
    samples = 330
    seed = 52983

variables
  uniform_uncertain =   33
  upper_bounds          1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1
                        1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1
                        1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1
                        1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1
                        1.1
  lower_bounds          0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9
                        0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9
                        0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9
                        0.9 0.9 0.9 0.9 0.9 0.9 0.9 0.9
                        0.9
  descriptors           'k_eta' 'k_gama' 'k_sent' 'k_sdent' 'k_tmasv'
                        'k_tmasl' 'k_tmasg' 'k_tmomv' 'k_tmome'
                        'k_tmoml' 'k_xk' 'k_xkes' 'k_xkge' 'k_xkl'
                        'k_xkle' 'k_xkvls' 'k_xkwvw' 'k_xkwlw' 'k_xkwew'
                        'k_qvapl' 'k_tnrgv' 'k_tnrgl' 'k_rodqq' 'k_qradd'
                        'k_qradv' 'k_qliht' 'k_sphts' 'k_cond' 'k_xkwvx'
                        'k_xkwlx' 'k_cd' 'k_cdfb' 'k_wkr'

interface
  fork
    asynchronous
      evaluation_concurrency = 24
    analysis_driver = 'dakota-vera-analysis'
    # extract TotalPressure metric (length 1)
    analysis_components = 'TotPress'
    parameters_file = 'params.in'
    results_file    = 'results.out'
    failure_capture recover NaN
    work_directory
    directory_tag
    named 'workdir'
      file_save  directory_save

responses
  num_response_functions = 1
    descriptors
      'TotalPressure'
  no_gradients
  no_hessians
```

```
Partial Rank Correlation Matrix between input and output:
             TotalPressure
    k_eta     3.51224e-02
    k_gama   -4.03274e-03
    k_sent    5.59178e-02
    k_sdent   5.19745e-02
    k_tmasv  -8.44833e-02
    k_tmasl   1.00123e-01
    k_tmasg   3.30672e-02
    k_tmomv   2.74738e-02
    k_tmome   2.05980e-02
    k_tmoml  -1.67107e-02
    k_xk     -3.88345e-02
    k_xkes   -7.66940e-02
    k_xkge    6.84111e-02
    k_xkl    -1.20002e-01
    k_xkle    2.65895e-02
    k_xkvls  -5.63864e-02
    k_xkwvw   7.63692e-03
    k_xkwlw  -9.35968e-02
    k_xkwew   1.80613e-02
    k_qvapl   4.88638e-02
    k_tnrgv   5.41003e-02
    k_tnrgl   2.93560e-02
    k_rodqq   4.55252e-01
    k_qradd  -4.37857e-02
    k_qradv   4.78528e-02
    k_qliht   4.10106e-03
    k_sphts   8.48286e-02
    k_cond   -6.56069e-02
    k_xkwvx  -9.93737e-03
    k_xkwlx   9.91808e-01
    k_cd      9.67573e-01
    k_cdfb   -1.58433e-02
    k_wkr    -5.51666e-02
```

Figure 7.2: Dakota console output showing partial correlations for the COBRA-TF simulated Progression Problem 6.

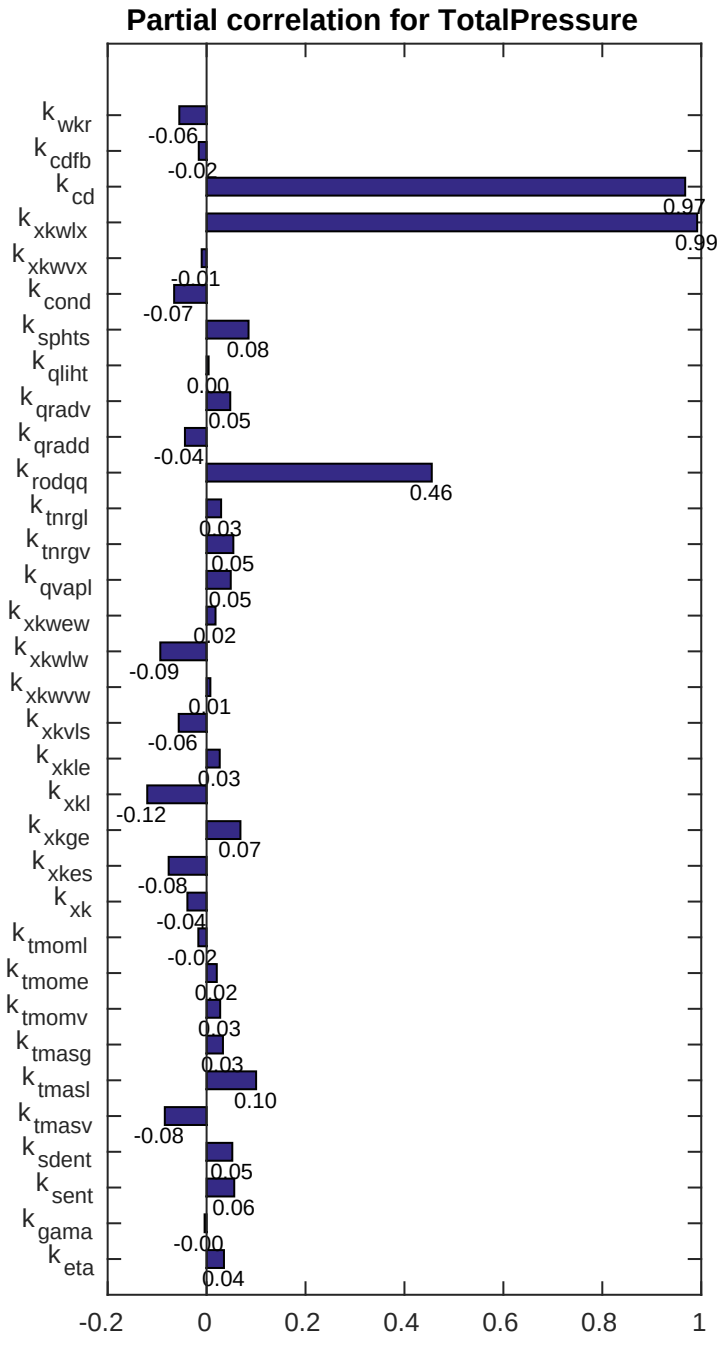


Figure 7.3: Bar graph showing partial correlation for each of 33 variables with `TotalPressure` for the COBRA-TF simulated Progression Problem 6.

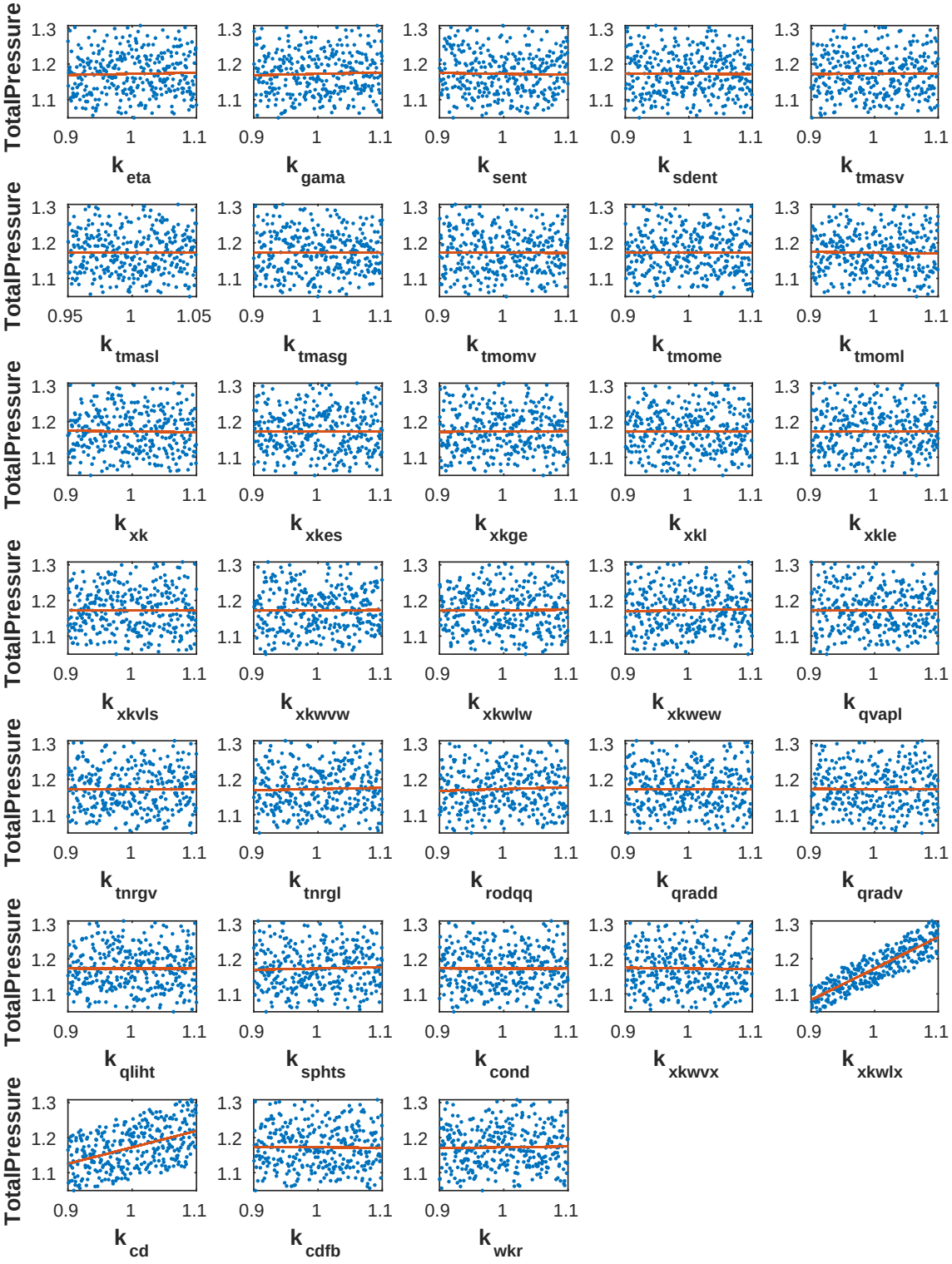


Figure 7.4: Scatter plots with regression lines for each of 33 variables with `TotalPressure` for the COBRA-TF simulated Progression Problem 6.

7.2.3 Morris Screening

Dakota Input: To change the study from LHS to the Morris screening method described in Section 3.2.4, one need only change the Dakota method specification to the following:

```
method
  psuade_moat
  partitions = 9   #to generate 10 levels
  samples = 340   # must be integer multiple of (num_vars + 1)
  seed = 20
```

The full Dakota input file is available in `examples/CobraTF/MorrisStudies/dakota_morris_33.in`.

Results and Discussion: The results of the Morris study are the modified means and standard deviations of the elementary effects in the Dakota console output, excerpted from `dakota_morris_33.out` into Figure 7.5. These indicate that inputs 30 and 31 (`k_cd` and `k_xkwlx`) have a strong main effect and some interaction effect and input 23 (`k_rodqq`) has a noticeable effect. They also indicate that no other parameters have any effect, save 10, corresponding to `k_tmoml`, which has a very small effect. This is physically consistent. The main forces impacting pressure drop are wall friction and loss coefficients with minor impacts due to possible boiling and turbulent mixing.

7.2.4 Screening to Reduce Parameters

Figure 7.6 summarizes the results of the conducted sensitivity studies showing correlations from the LHS study, effects from the Morris study, and variation seen in the centered parameter study. The various sensitivity methods are consistent with each other. Based on these results, the following studies will use only parameters inducing variation in `TotalPressure`: `k_rodqq`, `k_xkwlx`, and `k_cd`. Of these, only the last two have strong effects. Because the axial grid spacer loss coefficient and the axial wall friction coefficient appear in the axial momentum equations as multipliers of the axial velocity squared, the total pressure drop can only be related to the sum of these two coefficients. The impact of this sort of parameter nonidentifiability on model calibration will be illustrated in the analyses of Section 7.3.3.

A LHS study with 30 samples was next conducted over the three most sensitive parameters to (1) alleviate any potential confounding from the other parameters in the sensitivity metrics and (2) generate simulation data to use for calibration studies in the next section. The input and output files for this study are omitted from the text but are available in `examples/CobraTF/LHSStudies/`:

```
dakota_lhs_3.30.dat  dakota_lhs_3.30.out  dakota_lhs_3.30.in
dakota_lhs_3.20.dat  dakota_lhs_3.20.out  dakota_lhs_3.20.in
```

In particular, the tabular data files are used in the follow-on Bayesian calibration and surrogate generation examples. A companion LHS study using a different random number seed and 20 samples is also included for cross validation of the surrogate model.

```
>>>>>> PSUADE MOAT output for function 0:

*************************************************************
********************** MOAT Analysis ************************
-------------------------------------------------------------
Input   1 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input   2 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input   3 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input   4 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input   5 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input   6 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input   7 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input   8 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input   9 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  10 (mod. mean & std) =   1.0800e-05   1.5179e-05
Input  11 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  12 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  13 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  14 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  15 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  16 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  17 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  18 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  19 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  20 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  21 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  22 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  23 (mod. mean & std) =   1.2557e-02   1.0046e-03
Input  24 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  25 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  26 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  27 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  28 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  29 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  30 (mod. mean & std) =   1.7927e-01   7.4740e-03
Input  31 (mod. mean & std) =   9.5450e-02   7.9801e-03
Input  32 (mod. mean & std) =   0.0000e+00   0.0000e+00
Input  33 (mod. mean & std) =   0.0000e+00   0.0000e+00
<<<<< Function evaluation summary: 340 total (0 new, 340 duplicate)

<<<<< Iterator psuade_moat completed.
<<<<< Environment execution completed.
DAKOTA execution time in seconds:
  Total CPU        =       0.77
```

Figure 7.5: Dakota output showing modified means and standard deviations of elementary effects for the COBRA-TF simulated Progression Problem 6.

parameter	partial correlation	morris main	morris interaction	CPS variation
k_eta	0.09			
k_gama	-0.02			
k_sent	-0.04			
k_sdent	-0.06			
k_tmasv	-0.05			
k_tmasl	0.09			
k_tmasg	-0.19			
k_tmomv	-0.12			
k_tmome	0.00			
k_tmoml	0.07	1.26E-05	1.48E-05	low
k_xk	0.07			
k_xkes	-0.03			
k_xkge	-0.06			
k_xkl	0.04			
k_xkle	-0.05			
k_xkvls	0.12			
k_xkwvw	-0.10			
k_xkwlw	0.15			
k_xkwew	-0.01			
k_qvapl	-0.09			
k_tnrgv	-0.02			
k_tnrgl	-0.01	1.80E-06	5.69E-06	low
k_rodqq	0.93	1.26E-02	1.00E-03	medium
k_qradd	-0.02			
k_qradv	-0.01			
k_qliht	-0.01			
k_sphts	-0.07	1.80E-06	5.69E-06	low
k_cond	-0.04			
k_xkwvx	0.04			
k_xkwlx	1.00	1.79E-01	7.47E-03	high
k_cd	1.00	9.55E-02	7.98E-03	high
k_cdfb	-0.02			
k_wkr	0.03			

Figure 7.6: Summary of sensitivity analysis results for 33 COBRA-TF parameters. Missing values are identically zero. Highlighted rows will be used in subsequent studies.

7.3 Calibration Studies

In this section, we emulate the model calibration processes described in Sections 5.1.1 and 6.2 by generating synthetic data and then applying Dakota algorithms to determine the parameter values yielding the best match between model and data. The 10 synthetic data points, generated by adding independent and identically distributed Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ for $\sigma = 0.025$ to the nominal `TotalPressure` = 1.17231 bar, are placed in `ctf_dat.txt` for consumption by Dakota. They are:

```
1.196761
1.177256
1.160102
1.203438
```

```
1.147125
1.126521
1.190142
1.142087
1.215291
1.192622
```

7.3.1 Deterministic Calibration

Dakota Input: A local gradient-based algorithm should perform well for this model calibration problem since the previous global sensitivity analysis revealed smooth and linear trends. The Dakota input file for this problem is shown in Listing 7.3. Highlights include: use of the NL2SOL method (line 6) for local calibration; use of design variables (line 10) instead of uncertain variables, since we are calibrating; use of calibration terms in the responses section (line 35); and input of the external data file into Dakota (line 38) for computing the least-squares residuals. For illustrative purposes, we include `k_tmoml` as a fixed state variable in lines 16-18.

Results and Discussion: For our three-parameter calibration study, the end of the Dakota output (in file `examples/CobraTF/Calibration/dakota_ calibration_3.out`) indicates the best solution found, as shown in Figure 7.7. The best values of the parameters are shown, followed by the residuals between the model calculations and data. A summary of the study initial and final values/residual norms appears in Table 7.3. Comparing the residual norms shows the optimization solver made progress toward recovering the correct nominal values of `k_xkwlx`, `k_cd` and `k_rodqq`.

Table 7.3: Summary of three-parameter calibration for the COBRA-TF simulated Progression Problem 6, initial and final parameter values and residual norm.

	initial	final
`k_rodqq`	1.0100	0.9815
`k_cd`	1.0103	1.0182
`k_xkwlx`	0.9799	0.9949
$\frac{1}{2}\|\|r\|\|^2$	0.0051	0.0039

7.3.2 Surrogate Construction

The analysis of Section 7.2 resulted in a reduction of the initial 33 COBRA-TF parameters to a final set of three sensitive parameters with respect to induced variation in total pressure drop: `k_xkwlx`, `k_cd`, and `k_rodqq`.

Dakota Input: Listing 7.4 shows the Dakota input file `ctf_gp_eval.in` for fitting a kriging model with constant trend to total pressure drop as a function of these three parameters, and using this fit to predict total pressure drop on a set of 20 validation runs. The kriging fit itself was based on the results of using Dakota to run COBRA-TF on a Latin hypercube sample of size $N = 30$. The output of these COBRA-TF runs was written to the file `dakota_pstudy.dat`, which is subsequently read into this job via the `import_points` option (line 32). The validation runs were generated randomly from the input parameter domain and written to the file `ctf_val_des.dat`, and subsequently imported into Dakota via a distinct `import_points` option (line 19).

Listing 7.3: Dakota input file for deterministic local gradient-based calibration of three key parameters in the COBRA-TF simulated Progression Problem 6.

```
environment
  tabular_graphics_data
    tabular_graphics_file 'dakota_calibration_3.dat'

method
  nl2sol
    convergence_tolerance 1.0e-6

variables
  continuous_design = 3
    initial_point    1.01     1.0103   0.9799
    upper_bounds     1.05      1.1      1.1
    lower_bounds     0.95      0.9      0.9
    descriptors    'k_rodqq'  'k_cd'  'k_xkwlx'

  continuous_state = 1
    initial_state    1.0
    descriptors    'k_tmoml'

interface
  fork
    asynchronous
    analysis_driver = 'dakota-vera-analysis'
    # extract TotalPressure metric (length 1)
    analysis_components = 'TotPress'
    parameters_file = 'params.in'
    results_file    = 'results.out'
    failure_capture recover NaN
    work_directory
    directory_tag
    named 'workdir'
      file_save  directory_save

responses
  calibration_terms = 1
    descriptors
      'TotalPressure'
    calibration_data_file = 'ctf_dat.txt'
      freeform
    num_experiments = 10
  numerical_gradients
    central
    # coarse FD step as cobra might not be sensitive enough
    fd_step_size = 1.0e-2
  no_hessians
```

Listing 7.4: Dakota input file producing predictions for 20 validation runs from a GP emulator with estimated constant trend for the COBRA-TF simulated Progression Problem 6.

```
# Build and evaluate a Gaussian process emulator of COBRA-TF output
# at a user specified set of points

environment
  method_pointer = 'EvalSurrogate'
  tabular_graphics_data
    tabular_graphics_file = 'ctf_gp_evals.dat'

# Method to perform evaluations of the emulator

method
  id_method = 'EvalSurrogate'
  model_pointer = 'SurrogateModel'

  # Verbose will show the type form of the surrogate model
  output verbose

  list_parameter_study
    import_points = 'ctf_val_des.dat'

# Surrogate model specification
model
  id_model = 'SurrogateModel'
  surrogate global
    # GP model
    gaussian_process surfpack
      trend
        constant
    # compute and print diagnostics after build
    metrics 'rsquared' 'root_mean_squared'
      press
    import_points = 'dakota_pstudy.dat'

variables
  uniform_uncertain =   3
    upper_bounds          1.1 1.1 1.1
    lower_bounds          0.9 0.9 0.9
    descriptors           'k_xkwlx' 'k_cd' 'k_rodqq'

responses
  response_functions = 1
  descriptors = 'TotalPressure'
  no_gradients
  no_hessians
```

```
<<<<< Function evaluation summary: 44 total (0 new, 44 duplicate)
<<<<< Best parameters          =
                     9.8149284406e-01 k_rodqq
                     1.0181904232e+00 k_cd
                     9.9488498631e-01 k_xkwlx
                     1.0000000000e+00 k_tmoml
<<<<< Best residual norm =  8.8785103869e-02; 0.5 * norm^2 =  3.9413973345e-03
<<<<< Best residual terms =
                    -2.1621000000e-02
                    -2.1160000000e-03
                     1.5038000000e-02
                    -2.8298000000e-02
                     2.8015000000e-02
                     4.8619000000e-02
                    -1.5002000000e-02
                     3.3053000000e-02
                    -4.0151000000e-02
                    -1.7482000000e-02
<<<<< Best model response =
                     1.1751400000e+00
<<<<< Best data not found in evaluation cache

Confidence Interval for k_rodqq is [  7.2459126011e-01,  1.2383944280e+00 ]
Confidence Interval for k_cd is [  1.0181904232e+00,  1.0181904232e+00 ]
Confidence Interval for k_xkwlx is [  9.9488498631e-01,  9.9488498631e-01 ]
```

Figure 7.7: Dakota console output showing final results for calibration with three parameters.

Results and Discussion: Figure 7.8 compares the emulator predictions with the COBRA-TF calculations of total pressure drop for the 20 validation runs. The left panel plots predicted value against calculated value, with the resulting points falling very close to the desired 45° line. The right panel plots the standardized residuals (calculated minus predicted total pressure drop divided by standard error of predicted total pressure drop) against calculated value, indicating a fairly constant scatter in the standardized residuals around the zero line across the spectrum of calculated values with the exception of two potential outliers having standardized residual values of approximately -1.5. Excluding outliers, this result is desirable in that it both indicates unbiased prediction and supports the assumption of a homogeneous (constant) process variance σ^2. To investigate potential outliers, a simple screening procedure is applied to this standardized residual plot. First compute the interquartile range (IQR) of the standardized residuals, defined as the 75-th percentile minus the 25-th percentile of the standardized residuals. Second compute lower and upper bounds as the 25-th percentile minus $1.5 \times$ IQR and the 75-th percentile plus $1.5 \times$ IQR, respectively. Any standardized residuals falling outside these bounds are flagged for further investigation as potential outliers. For this kriging fit, the lower and upper bounds calculated in this way were $(-1.17, 1.16)$. The two potential outliers previously flagged do indeed fall outside this range, having standardized residual values of -1.60 and -1.49, while the remaining standardized residuals fall well within this range.

The leave-one-out cross-validation RMSPE of this kriging emulator, requested with the option `press` (line 31), evaluated to `4.0198651083e-05`. When validation samples are available, as in this

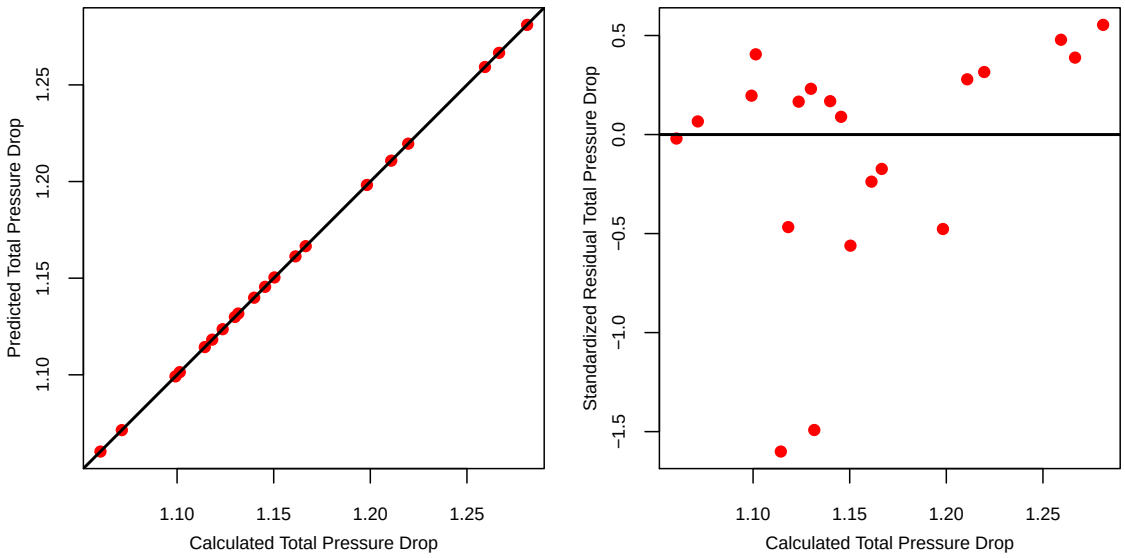


Figure 7.8: Predicted vs. calculated total pressure drop (left panel) and standardized residual vs. calculated total pressure drop (right panel) for 20 validation runs (red circles).

example, they can be used to compute a validation RMSPE. For N_v validation samples, calculate

$$RMSPE = \sqrt{\frac{1}{N_v}\sum_{i=1}^{N_v}(C_i - P_i)^2}\,,$$

where P_i and C_i denote the i-th predicted and calculated values, respectively. For our validation sample of $N_v = 20$, the RMSPE evaluates to `2.975525e-05`. These RMSPE values are 0.018% and 0.013% of the observed range in the 20 calculated total pressure drops, indicating the kriging emulator possesses high accuracy for this application.

The assumption of Gaussian errors can be checked by examining normal probability plots of the standardized residuals from both leave-one-out cross validation and out-of-sample validation. Figure 7.9 shows normal probability plots for both of these cases. Theoretical quantiles from the standard normal distribution are plotted on the x-axis, while the corresponding sample quantiles of the standardized residuals are plotted on the y-axis. Consistency with the Gaussian error assumption is indicated by the plotted points exhibiting strong linear association. The simple correlation coefficients for the two cases are 0.990 and 0.979, respectively, suggesting that the Gaussian error assumption is reasonable. In particular, the potential outliers identified previously do not have standardized errors inconsistent with the Gaussian error assumption, suggesting the surrogate assumptions are validated. The standardized residuals resulting from leave-one-out cross validation are correlated, making the normal probability plot somewhat harder to interpret as the theoretical quantiles are calculated assuming independence. On the other hand, for the validation runs, the residuals were transformed to obtain uncorrelated standardized residuals for use in the normal probability plot [6].

7.3.3 Bayesian Calibration

We illustrate here the use and comparison of the Bayesian model calibration techniques DRAM and DREAM, discussed in Section 6.2, for COBRA-TF. We focus on the three parameters `k_xkwlx`,

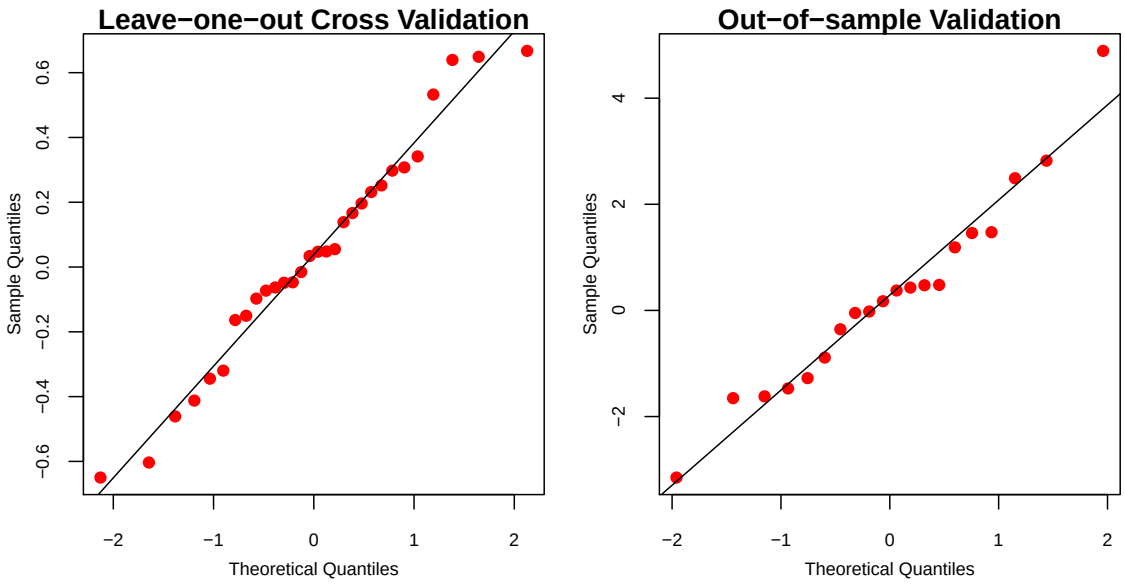


Figure 7.9: Normal probability plots based on standardized residuals from leave-one-out cross validation (left panel) and out-of-sample validation (right panel).

`k_cd`, and `k_rodqq` which, as detailed in the sensitivity analysis of Section 7.2, had non-negligible values for the Morris elementary effect statistics. The synthetic data of Section 7.3, generated as detailed in Section 6.2.5, is consistent with the likelihoods employed in DRAM and DREAM.

For DRAM and DREAM implementation, we employed the surrogate constructed using the three sensitive parameters (denoted collectively by θ), as detailed in Section 7.3.2. In both analyses, we enforced the parameter bounds summarized in Table 7.4. We employed uniform densities over these ranges as prior densities for each random variable. A statistical model of the synthetic data provided in Section 7.3 is given by

$$D_i = y(\theta) + \varepsilon_i \, , i = 1, \ldots, N \, ,$$

where $y(\theta)$ is the COBRA-TF total pressure drop for parameter value θ, $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, and $N = 10$. The inverse variance $1/\sigma^2$ was assigned a Gamma prior distribution having $a = 1$ and $b = 0.000625 = 0.025^2$.

Table 7.4: Parameter bounds used to construct prior densities.

Descriptors	`k_xkwlx`	`k_cd`	`k_rodqq`
Lower Bounds	0.9	0.9	0.9
Upper Bounds	1.1	1.1	1.1

We ran the DRAM chain for 202,000 iterations and constructed 5 DREAM chains each of length 42,000. As illustrated by the representative DRAM chains for `k_rodqq` and σ^2, which are plotted in Figure 7.10(a) and Figure 7.10(b), the chain has burned in by 2000 so we employed the last 200,000 elements thinned by taking every 20th iterate for a total of 10,000 samples for kernel density estimation (kde). For DREAM, the first 10,000 iterates of the combined chains were discarded as burn in, leaving 200,000 elements which were thinned to 10,000 samples as with

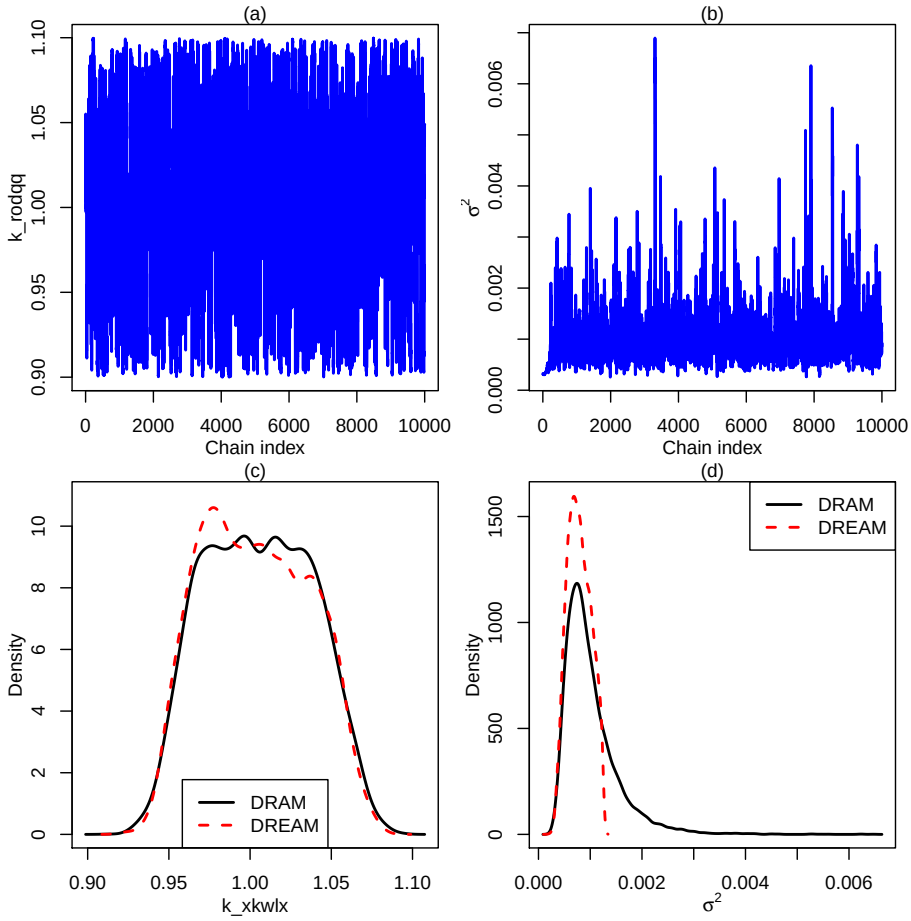


Figure 7.10: (a) and (b): Representative DRAM chains for `k_rodqq` and σ^2. (c) and (d): Comparison of the densities for `k_xkwlx` and σ^2 provided by DRAM and DREAM.

DRAM. The DRAM and DREAM densities for `k_xkwlx` are compared in Figure 7.10(c). For σ^2, the comparison is provided in Figure 7.10(d).

These results provide a verification test in the manner detailed in Section 6.2.4 (iv). Although more rigorous verification to an analytical solution is not possible, the two sampling algorithms considered can be directly compared. For example, the DRAM and DREAM marginal densities for `k_xkwlx` compare favorably. For σ^2, DREAM appears to undersample a heavy tail in its posterior density. To explore further, energy tests as described in Appendix A.3 are conducted. These tests compare the joint DRAM and DREAM samples to test the hypothesis that the underlying target distributions are equivalent. When the DRAM and DREAM samples are thinned further down to 100, 1000, and 5000, they cannot be distinguished from each other at the 1% level of significance. However, a significant difference is observed between the full sample sets of size 10,000 (p-value 0.001). These results indicate that DRAM and DREAM provide similar calibrations in this example, but that DREAM may undersample heavy tailed distributions. Further investigation of this behavior is necessary.

Figure 7.1 shows that total pressure drop exhibits a positive linear trend in inputs `k_xkwlx` and `k_cd`. The Morris sensitivity results of Figure 7.6 indicate very little interaction between these inputs. In combination, these observations suggest that `k_xkwlx` and `k_cd` may trade-off in

the process of being calibrated and, in fact, this behavior is illustrated in Figure 7.11 by calibrated negative correlation between them (recall prior independence between all three inputs was assumed). These plots reflect the parameter limits specified in Table 7.4. The fact that DRAM and DREAM yield similar joint sample plots further verifies the substantial degree of consistency in the results of both methods applied to this implementation of COBRA-TF.

The input decks for DRAM and DREAM are provided in Listings 7.5 and 7.6. These input decks both show a general approach to redirecting the likelihood calculation to an emulator. In Listing 7.5, line 3 redirects QUESO to the model block in lines 14-20, in which Dakota builds a Gaussian process surrogate with constant trend built from data contained in the file `dakota_pstudy.dat`. An analagous approach to surrogate redirection is taken in Listing 7.6. The file `ctf_dat.txt` contains the calibration data listed at the beginning of Section 7.3 along with nominal observation error variances corresponding to each observation. Each row represents an individual experiment, and the associated observations are placed in the first set of columns followed by the nominal variances in the same order.

It was noted in Section 6.2.3 that once Bayesian chains have been constructed, values from the chain can be used as inputs to surrogate models to compute calibrated predictions for additional quantities of interest. This is illustrated in Figure 7.12, where we compare predictive distributions for the total pressure drop computed from the surrogate model evaluated using inputs from the DRAM and DREAM chains. The nominal total pressure drop value of 1.172 bar, computed using COBRA-TF with nominal parameter values, lies within the central portion of both predictive distributions. Note that samples of the mean-zero Gaussian error process are added to each surrogate prediction, where the variance of the error is given by the sampled variance value corresponding to the parameter sample generating the surrogate prediction. It appears that the predictive distribution based on DREAM samples slightly underestimates the predictive uncertainty, likely due to the previously observed undersampling of the heavy tailed σ^2 posterior distribution. Significant differences between DRAM and DREAM predictive samples are detected by the energy test at the 1% level for sample sizes of 1000, 5000, and 10,000, but not for 100 (p-value 0.188).

The Dakota input deck used to generate surrogate predictions is provided in Listing 7.7, where one can note that it inputs DRAM values from the file `dram_thinned.txt` and exports surrogate predictions of total pressure drop to the file `ctf_gp_evals_dram.dat`. Using the DRAM and

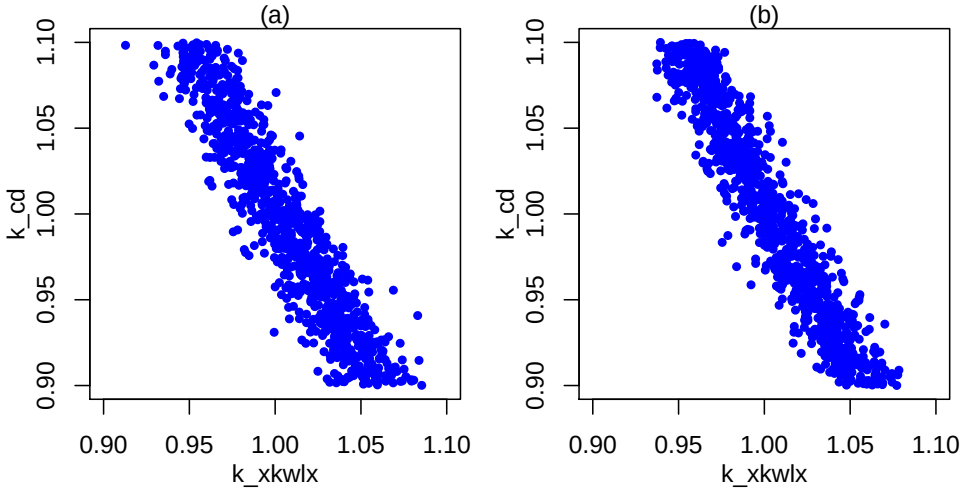


Figure 7.11: Joint sample points for `k_xkwlx` and `k_cd` provided by (a) DRAM and (b) DREAM.

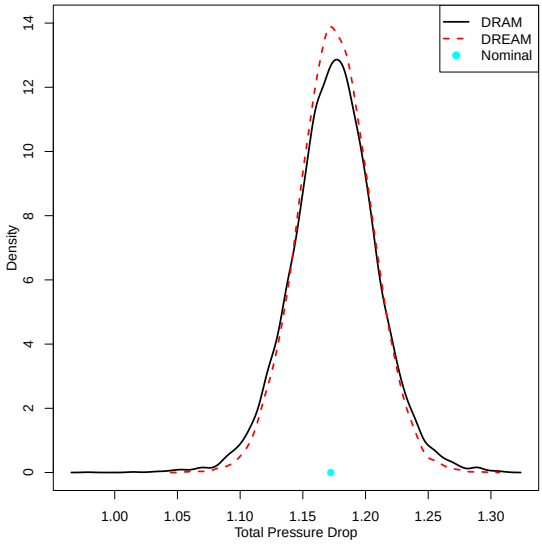


Figure 7.12: Comparison of the predictive distributions constructed using the surrogate with inputs from the DRAM and DREAM chains. The nominal total pressure drop value of 1.172 bar was computed using COBRA-TF.

DREAM predictive samples (error-adjusted surrogate predictions) and Wilks' method, a new total pressure drop measurement has 99% chance of not exceeding 1.262 and 1.246, respectively, with 99% confidence based on 78th-order Wilks applied to 10,000 samples. DREAM undersampling of the σ^2 posterior heavy tail is likely to have an impact on tail behavior of predictive distributions, potentially biasing the estimation of coverage thresholds as suggested in this example.

Listing 7.5: Input for Dakota-QUESO DRAM as applied to the COBRA-TF surrogate for Progression Problem 6.

```
method
  bayes_calibration queso
    model_pointer = 'GP'
    logit_transform
    dram
    proposal_covariance prior
    calibrate_error_multipliers one
      hyperprior_alphas 1
      hyperprior_betas 1
    chain_samples = 202000
    seed = 52983
    export_chain_points_file = 'dram_result.txt'

model
  id_model = 'GP'
  surrogate global
    gaussian_process surfpack
      trend
        constant
    import_build_points_file = 'dakota_pstudy.dat' annotated

variables
  uniform_uncertain 3
    upper_bounds      1.1 1.1 1.1
    lower_bounds      0.9 0.9 0.9
    descriptors       'k_xkwlx' 'k_cd' 'k_rodqq'

interface
  direct
    analysis_driver = 'text_book'

responses
  calibration_terms = 1
  descriptors 'TotalPressure'
  calibration_data_file 'ctf_dat.txt' freeform
  num_experiments = 10
  variance_type = 'scalar'
  no_gradients
  no_hessians
```

Listing 7.6: Input for Dakota DREAM as applied to the COBRA-TF surrogate for Progression Problem 6.

```
method
  bayes_calibration dream
    model_pointer = 'GP'
    calibrate_error_multipliers one
      hyperprior_alphas 1
      hyperprior_betas 1
    chain_samples = 210000
    chains = 5
    seed = 52983
    export_chain_points_file = 'dream_result.txt'

model
  id_model = 'GP'
  surrogate global
    gaussian_process surfpack
      trend
        constant
    import_build_points_file = 'dakota_pstudy.dat' annotated

variables
  uniform_uncertain 3
    upper_bounds      1.1 1.1 1.1
    lower_bounds      0.9 0.9 0.9
    descriptors       'k_xkwlx' 'k_cd' 'k_rodqq'

interface
  direct
    analysis_driver = 'text_book'

responses
  calibration_terms = 1
  descriptors 'TotalPressure'
  calibration_data_file 'ctf_dat.txt' freeform
  num_experiments = 10
  variance_type = 'scalar'
  no_gradients
  no_hessians
```

Listing 7.7: Dakota input used to sample predictive distributions for total pressure drop using DRAM evaluations.

```
# Build and evaluate a Gaussian process emulator of COBRA-TF output
# at a user specified set of points

environment
  method_pointer = 'EvalSurrogate'
  tabular_graphics_data
    tabular_graphics_file = 'ctf_gp_evals_dram.dat'

# Method to perform evaluations of the emulator

method
  id_method = 'EvalSurrogate'
  model_pointer = 'SurrogateModel'

  # Verbose will show the type form of the surrogate model
  output verbose

  list_parameter_study
    import_points = 'dram_thinned.txt'

# Surrogate model specification
model
  id_model = 'SurrogateModel'
  surrogate global
    # GP model
    gaussian_process surfpack
      trend
        constant
    # compute and print diagnostics after build
    metrics 'rsquared' 'root_mean_squared'
      press
    import_points = 'dakota_pstudy.dat'

variables
  uniform_uncertain =   3
    upper_bounds          1.1 1.1 1.1
    lower_bounds          0.9 0.9 0.9
    descriptors           'k_xkwlx' 'k_cd' 'k_rodqq'

responses
  response_functions = 1
  descriptors = 'TotalPressure'
  no_gradients
  no_hessians
```

Chapter 8

CIPS — Crud Induced Power Shift

This chapter extends the study of the COBRA-TF problem of the previous chapter to include coupling to both neutronics (via MPACT) and crud chemistry (via MAMBA1D) as needed to model the CIPS phenomenon. The problem geometry is essentially the same, consisting of a single 17x17 assembly. The VUQ workflow is extended to include user parameters available through the VERA Common Input, low-level closure (code) parameters for both COBRA-TF and MAMBA1D, and effects from perturbing neutronics cross-sections. For CIPS, the quantities of interest consist of maximum assembly crud thickness and total boron, both of which indicate the occurrence of power shifts arising from deposited crud.

8.1 The CIPS Phenomenon

In brief, CIPS is a coupled multi-physics phenomenon in which impurities (crud) present in the coolant deposit on the fuel pin cladding and absorb boron. The presence of boron locally reduces moderation which suppresses power and shifts the power profile accordingly. This phenomenon is modeled in CASL by treating the coupled effects of thermal hydraulics, neutronics and crud chemistry using the VERA codes COBRA-TF, MPACT and MAMBA1D, respectively. The interaction among the codes including data exchanged is shown in Figure 2.4. For the purposes of this study we chose two Quantities of Interest (QoI) consisting of maximum crud thickness and total boron, both of which indicate the onset of CIPS behavior. More details about the CIPS simulation capability in VERA can be found in [42] and [9].

8.2 Parameter Ranking and Downselection

A Phenomena Identification Ranking Table (PIRT) assessment for CIPS was conducted to identify all potentially influential physical phenomena affecting CIPS. From this, corresponding input parameters which possibly influence the two quantities of interest, i.e. maximum crud thickness and total boron, were selected and expert opinion used to determine nominal values and ranges for subsequent Quantitative PIRT (QPIRT) studies. For the purposes of this study, QPIRT amounts to using Dakota to perform centered parameter sensitivity studies around the nominal values with perturbations equal to the parameter range. The result of these studies is a quantified ranking of parameter importance based on single parameter effects which can then be used to downselect a subset of parameters to use in sampling for UQ.

The candidate set of physical phenomena deemed important to CIPS, i.e. those expected to

influence maximum crud thickness and total boron, include coolant flow, heat transfer, power characteristics and crud chemistry. Input parameters related to these take one of four forms:

- normal user input via VERA Common Input
- code-level COBRA-TF parameters via auxiliary input text files, `vuq_mult.txt` and `vuq_param.txt`
- code-level MAMBA1D parameters via an auxiliary input text file, `vuq_mult_mamba.txt`
- Cross-sections perturbed using the approach described in Appendix C.3.3 and incorporated as a collection of pre-generated files from which Dakota can select, i.e. the file number becomes a Dakota input parameter

All relevant MPACT parameters are included in those treated through VERA Common Input. The following subsections summarize the results of the four different types of parameter studies.

8.2.1 VERA Common Input Parameters

For user parameters available in VERA Common Input the parameters summarized in Table 2.7 are potentially important to CIPS and are included in a QPIRT analysis. This is performed as a Dakota centered parameter study using a single step in the positive and negative directions with step size equal to the parameter ranges shown. Results are summarized in Table 8.1.

Table 8.1: CIPS VERA Common Input Centered Parameter Sensitivity Study

		Max Crud	Total Boron
Perturb	Parameter (XML-based)	diff (%)	diff (%)
0.9950	*ASSEMBLIES/Assembly_B9B-128I/Fuels/Fuel_BLK/thden	0.00	0.00
1.0050	*ASSEMBLIES/Assembly_B9B-128I/Fuels/Fuel_BLK/thden	0.00	0.00
0.9950	*ASSEMBLIES/Assembly_B9B-128I/Fuels/Fuel_U43/thden	0.00	0.00
1.0050	*ASSEMBLIES/Assembly_B9B-128I/Fuels/Fuel_U43/thden	0.00	0.00
0.9800	*CORE/rated_flow	2.92	22.92
1.0200	*CORE/rated_flow	−3.79	−20.10
1.0400	*CORE/rated_power	10.74	54.20
−0.0500	+ASSEMBLIES/Assembly_B9B-128I/Fuels/Fuel_BLK/enrichments[1]	0.12	0.88
0.0500	+ASSEMBLIES/Assembly_B9B-128I/Fuels/Fuel_BLK/enrichments[1]	−0.24	−0.63
−0.0500	+ASSEMBLIES/Assembly_B9B-128I/Fuels/Fuel_U43/enrichments[1]	−0.14	0.04
0.0500	+ASSEMBLIES/Assembly_B9B-128I/Fuels/Fuel_U43/enrichments[1]	0.21	0.04
0.4826	+STATES/State_1/pressure	71.28	133.94
−0.4826	+STATES/State_1/pressure	−49.88	−99.98
−2.7778	+STATES/State_1/tinlet	−8.23	−41.93
2.7778	+STATES/State_1/tinlet	8.39	52.19

8.2.2 COBRA-TF Parameters

A QPIRT for all COBRA-TF code-level (e.g. closure relation) parameters exposed through the auxiliary input files is summarized in Table 8.2. Missing entries in the table correspond to runs which failed to finish. These failures are initially thought to be due to hardware issues on Titan based on the termination message, but additional effort was not made to rigorously confirm this.

Table 8.2: CIPS COBRA-TF Input Centered Parameter Sensitivity Study

		Max Crud	Total Boron		Max Crud	Total Boron
Parameter	Value	diff (%)	diff (%)	Value	diff (%)	diff (%)
k_cd	0.9	−0.11	−0.05	1.1	−0.08	0.68
k_cdfb	0.9	0.00	0.00	1.1	0.00	0.00
k_clad_avg_tmp	0.9	0.00	0.00	1.1	0.00	0.00
k_cond	0.9	0.00	0.00	1.1	0.00	0.00
k_cool_avg_den	0.9	−1.70	−4.75	1.1	1.09	7.65
k_cool_avg_tmp	0.9	0.21	0.25	1.1	−0.41	−0.58
k_eta	0.9	0.00	0.00	1.1	0.00	0.00
k_fuel_avg_tmp	0.9	0.00	0.00	1.1	0.00	0.00
k_gama	0.9	0.01	−0.11	1.1	−0.02	0.03
k_hgap	0.9	0.00	0.00			
k_htcl	0.9	0.00	0.00	1.1	0.00	0.00
k_htcv	0.9	0.00	0.00	1.1	0.00	0.00
k_qliht	0.9	0.00	0.00	1.1	0.00	0.00
k_qradd	0.9	0.00	0.00	1.1	0.00	0.00
k_qradv	0.9	0.00	0.00	1.1	0.00	0.00
k_qvapl	0.9	0.00	0.00	1.1	0.00	0.00
k_rodqq	0.9	−20.40	−89.07	1.1	23.85	143.28
k_sdent	0.9	0.00	0.00	1.1	0.00	0.00
k_sent	0.9	0.00	0.00	1.1	0.00	0.00
k_sphts	0.9	0.00	0.00	1.1	0.00	0.00
k_tmasg	0.9	0.00	0.00	1.1	0.00	0.00
k_tmasl	9.5	0.03	−0.04	1.0	0.02	0.27
k_tmasv	0.9	0.03	−0.12	1.1	0.04	−0.05
k_tmome	0.9	0.00	0.00	1.1	0.00	0.00
k_tmoml	0.9	−0.26	0.22	1.1	−0.21	0.00
k_tmomv	0.9	0.00	0.00	1.1	0.00	0.00
k_tnrgl	0.9	0.07	0.14	1.1	0.02	−0.07
k_tnrgv				1.1	0.03	−0.13
k_wkr				1.1	0.15	−0.08
k_xk	0.9	0.00	0.00	1.1	0.00	0.00
k_xkes	0.9	0.00	0.00	1.1	0.00	0.00
k_xkge	0.9	0.00	0.00	1.1	0.00	0.00
k_xkl	0.9	0.00	0.00	1.1	0.00	0.00
k_xkle	0.9	0.00	0.00	1.1	0.00	0.00
k_xkvls	0.9	0.00	0.00	1.1	0.00	0.00
k_xkwew	0.9	0.00	0.00	1.1	0.00	0.00
k_xkwlw	0.9	0.06	0.10	1.1	0.15	−0.08
k_xkwlx	0.9	−0.03	−0.81	1.1	0.16	0.99
k_xkwvw	0.9	0.00	0.00	1.1	0.00	0.00
k_xkwvx	0.9	0.02	−0.14	1.1	0.01	0.00

8.2.3 MAMBA1D Parameters

The MAMBA1D coolant chemistry parameters are treated in the same manner as the COBRA-TF closure parameters, i.e. an auxiliary file is used to impose perturbations via (2.5). The current set of parameters is summarized in Table 2.8, and centered parameter sensitivity study (QPIRT) results are summarized in Table 8.3. More detailed information about the parameters can be obtained from [24].

For the QPIRT, all MAMBA1D parameters listed in Table 2.8 were perturbed by ±10%. This default reflects the fact that we are using MAMBA1D which is a lower fidelity version of MAMBA2D and MAMBA3D. The parameters for MAMBA1D often lump finer physical features into more empirical values, and this makes such parameters less amenable to expert opinion for both their ranges as well as their uncertainty distributions. Missing entries in the table correspond to simulations for which a converged solution was not achieved.

8.2.4 Neutronics Cross Section Sensitivity

The sensitivity of maximum crud thickness and total boron to neutronics cross sections is addressed using a collection of pre-generated cross section libraries (text files) that perturb the cross section data in such a way that correlations among the different reactions are preserved. Details of how these perturbed cross section files are created are described in Appendix C.3.3. From the perspective of a Dakota sensitivity study, the files are parameterized by file number and a sensitivity study is performed using the entire collection of files. Results are summarized in Table 8.4.

8.2.5 Parameter Downselect

The preceding parameter sensitivity (QPIRT) studies are used to downselect parameters for subsequent UQ sampling studies. Parameters whose perturbations produced a change in either quantity of interest of 10% or more are retained. In addition, the MAMBA1D parameters `Bfract` and `Bthresh` are retained based on their use in previous CIPS calibration activity [9]. The one relevant COBRA-TF parameter `k_rodqq` represents a code coupling parameter governing the fraction of power transferred from neutronics (MPACT) into the coolant (COBRA-TF) and is not a physical parameter. Because the sensitivity of CIPS to power is accounted for in a physically consistent manner by virtue of the code coupling, the `k_rodqq` parameter is not included in the remainder of this study. Removal of this parameter together with the observation that the largest effect of cross section perturbations is less than 2% indicates that no COBRA-TF and cross section perturbations are sufficiently important to include in the UQ sampling studies.

The parameters chosen for inclusion in UQ sampling studies to generate samples used for Wilks analyses are summarized in Table 8.5. The first four entries correspond to VERA Common Input type parameters, while the rest represent low-level MAMBA1D code parameters exposed via auxiliary input. The sizes of the perturbations have been reduced based on our (ongoing) experience performing sampling with joint parameter variations in which we find it more difficult to achieve converged solutions.

8.3 CIPS UQ Simulations

As of version 6.4, Dakota supports Wilks-based random sampling for any probability quantile, confidence level, order statistic and both one-sided and two-sided bounds. This capability is limited to a single response. With this capability, we could use Dakota to perform various Wilks analyses.

Table 8.3: CIPS MAMBA1D Input Centered Parameter Sensitivity Study

	Max Crud	Total Boron	Max Crud	Total Boron
Parameter	diff (%)	diff (%)	diff (%)	diff (%)
k_Bfract	0.83	−0.78		
k_Bthresh	−0.88	11.68	−0.42	−12.73
k_Cpor	−18.76	−52.54		
k_crud_solid	11.40	28.73	−5.98	−23.83
k_Dc	0.30	−1.71		
k_delta_r	−3.51	−21.43		
k_fac	7.02	18.60	−8.03	−19.88
k_Hc	7.02	18.60	−8.03	−19.88
k_Hfg	−0.25	1.79	0.30	−1.71
k_kp2	−7.35	−18.00	7.22	17.11
k_M	−0.16	7.71	−0.88	−6.82
k_MB	−0.01	−6.55	−0.47	6.19
k_MB10	0.00	0.00	0.00	0.00
k_MB11	0.00	0.00		
k_MFe	−1.19	−2.66	0.47	2.87
k_mit0	−52.92	−99.99	111.24	194.02
k_mitMax	0.42	−4.18	0.00	0.00
k_MLi	−0.24	−0.69	0.00	0.70
k_MNi	0.28	1.55	−0.53	−1.34
k_Nc	7.02	18.60	−8.03	−19.88
k_rc	−8.03	−19.88		
k_RtcB	0.00	0.00		
k_RtcB2	0.06	−1.61	−0.31	2.43
k_RtcB3	1.59	1.93	−1.46	−1.42
k_RtcB4	0.53	0.19	−0.04	0.05
k_RtcB5	0.00	0.01	0.00	0.00
k_RtcB6	0.00	0.00	0.00	0.00
k_RtcNB	0.06	−0.74	−0.16	0.48
k_Tsat	−55.40	−99.83		

However, the significant computational cost of each single assembly CIPS simulation led us to instead generate a large collection of random samples (799) with which we can then perform several variants of Wilks UQ.

We generated a collection of Dakota random sampling studies on the ORNL Titan supercomputer for the set of parameters in Table 8.5. Each Dakota study involved 40 concurrent runs (Titan imposes an upper limit on the number of simultaneous processes), each of which required 60 cores. Run times varied but were typically about 15 hours. A total of 20 such Dakota studies were run with each one differing only by the random number seed used to sample from the parameter distributions. Of the 800 runs we attempted, one run failed to converge within the 24 hour wall-time limit. An example Dakota input file is shown in Listing 8.1.

Listing 8.1: Dakota input file for random (Monte Carlo) sampling using the parameter distributions in Table 8.5.

```
# DAKOTA input file for UQ sampling study

environment
  tabular_graphics_data
    tabular_graphics_file 'dakota_sampling.dat'

method
  sampling
    samples = 40
    seed = 01716 rng rnum2
    sample_type random
    distribution cumulative

variables
  normal_uncertain  4
  means                 291.333333333   15.5132039025   1.0     1.0
  std_deviations        1.3889          0.241315        0.02    0.01
  descriptors     'STATES/State_1/tinlet'
                  'STATES/State_1/pressure'
                  '*CORE/rated_power'
                  '*CORE/rated_flow'

  uniform_uncertain  11
  lower_bounds    0.99  0.99  0.99  0.99  0.99  0.99  0.99  0.99  0.99  0.99  0.99
  upper_bounds    1.01  1.01  1.01  1.01  1.01  1.01  1.01  1.01  1.01  1.01  1.01
  descriptors     'k_Bthresh' 'k_Cpor' 'k_crud_solid' 'k_delta_r' 'k_fac' 'k_Hc'
                  'k_kp2' 'k_mit0' 'k_Nc' 'k_rc' 'k_Tsat'

interface
  fork
    asynchronous
      evaluation_concurrency = 40
    analysis_driver = 'dakota-vera-analysis'
    analysis_components = 'k_eff' 'max_fuel_T' 'max_fuel_P'
    parameters_file = 'params.in'
    results_file    = 'results.out'
    failure_capture recover NaN NaN
    work_directory
    directory_tag
    named 'workdir'
      file_save  directory_save
    link_files "output_adapt*" "COBRATF.ini" "MPACT.ini" "materials.inp" "
       bison_table_SGS_rev5.txt"

responses
  num_response_functions = 2
    descriptors
       'max_crud_thickenss' 'total_boron'
  no_gradients
  no_hessians
```

Table 8.4: CIPS Cross-Section Sensitivity Study

	Max Crud	Total Boron		Max Crud	Total Boron		Max Crud	Total Boron
File #	diff (%)	diff (%)	File #	diff (%)	diff (%)	File #	diff (%)	diff (%)
1	−0.03	0.24	34	−0.09	0.60	67	0.16	0.35
2	0.15	0.11	35	−0.20	−0.03	69	−0.04	−0.06
3	−0.05	−0.10	36	0.20	0.33	70	0.15	0.46
4	−0.20	−0.71	37	0.16	0.28	71	0.07	0.40
5	0.10	0.42	38	−0.31	−0.58	72	0.00	−0.01
6	−0.04	−0.56	39	0.08	0.45	73	−0.19	−0.38
7	−0.05	0.34	40	−0.04	0.88	74	0.23	0.28
8	0.07	0.21	41	−0.45	0.33	75	−0.12	−0.38
9	−0.11	0.36	42	−0.44	−0.18	76	−0.04	0.74
10	−0.08	−0.27	43	0.34	1.12	78	0.19	0.30
11	−0.01	0.48	44	−0.27	−0.09	79	−0.39	−0.49
12	0.09	0.40	45	0.19	0.58	80	−0.02	1.08
13	−0.14	0.46	46	0.00	0.21	81	−0.06	0.63
14	−0.19	0.34	47	−0.14	−0.67	82	−0.10	−0.76
15	0.18	0.81	48	−0.37	0.16	83	−0.15	−0.31
16	−0.44	−0.76	49	−0.08	1.15	84	0.11	0.39
17	−0.31	−0.29	50	−0.24	−0.37	85	0.11	−0.04
18	−0.12	0.25	51	0.13	0.14	86	−0.15	0.20
19	0.08	−0.69	52	−0.23	−0.10	87	−0.42	−0.42
20	−0.29	−0.89	53	0.14	0.95	88	−0.25	−1.51
21	−0.25	0.12	54	−0.29	−0.21	89	0.03	0.26
22	−0.49	−0.14	55	0.28	0.69	90	−0.18	0.22
23	0.22	0.79	56	−0.20	−0.27	91	−0.38	0.09
24	−0.05	−0.37	57	−0.36	0.56	92	−0.11	0.39
25	−0.02	0.21	58	0.19	0.37	93	0.10	0.36
26	0.06	0.18	59	0.41	0.78	94	0.01	−0.35
27	0.24	0.37	60	0.15	0.76	95	0.08	1.08
28	−0.44	−0.16	61	0.03	0.05	96	−0.28	−0.36
29	0.22	0.04	62	−0.53	−0.36	97	−0.43	−0.23
30	−0.70	−0.78	63	0.01	−0.28	98	−0.10	1.04
31	−0.54	−0.39	64	−0.03	0.32	99	−0.01	0.05
32	0.00	0.27	65	0.00	0.13	100	−0.52	−0.37
33	−0.06	−0.05	66	0.10	−0.01			

In addition to the random sampling studies, we also generated 159 Latin hypercube samples to give space-filling coverage of the uncertain parameter distributions for the purpose of constructing surrogate models to compare with direct computation in the ensuing uncertainty quantification studies.

Table 8.5: CIPS UQ parameters

Parameter	Value	Distribution
+STATES/State_1/tinlet	291.33 ± 1.39	Normal
+STATES/State_1/pressure	15.513 ± 0.241	Normal
*CORE/rated_power	1.0 ± 2%	Normal
*CORE/rated_flow	88.79 ± 2%	Normal
k_Bthresh	1.0 ± 1%	Uniform
k_Cpor	1.0 ± 1%	Uniform
k_crud_solid	1.0 ± 1%	Uniform
k_delta_r	1.0 ± 1%	Uniform
k_fac	1.0 ± 1%	Uniform
k_Hc	1.0 ± 1%	Uniform
k_kp2	1.0 ± 1%	Uniform
k_mit0	1.0 ± 1%	Uniform
k_Nc	1.0 ± 1%	Uniform
k_rc	1.0 ± 1%	Uniform
k_Tsat	1.0 ± 1%	Uniform

8.4 Wilks Uncertainty Quantification

Uncertainty quantification based on Wilks sampling essentially involves determining the number of independent samples needed to make specific guarantees regarding one-sided or two-sided bound coverage of a statistical population of interest with an accompanying level of confidence. For example, the classic 95/95 Wilks analysis requires 59 samples to guarantee with 95% confidence that the 95% quantile is bounded by the largest response value in the 59 samples. This represents a one-sided Wilks analysis which can be extended to a two-sided analysis pertaining to a 95% probability interval being bounded both below and above by the smallest and largest responses in the samples, respectively. This comes with the added cost of more required samples, e.g. 93 samples for two-sided 95/95 Wilks. Finally, the order of the analysis can be increased. This expands the statement pertaining to the largest sample value (or smallest and largest for two-sided) to the largest r values where r is the order of analysis.

For purposes of demonstrating and validating Wilks-based sampling, we use Dakota and an ad hoc approach to import the collection of 799 random samples and generate the probability levels shown in Table 8.6 for a one-sided first-order Wilks baseline. All of the probability levels in the table for each quantity of interest have confidence levels of 99.9999% or higher and provide estimates of true quantile values for comparisons with Wilks sampling based on more typical sample sizes.

To demonstrate Wilks UQ for the single assembly CIPS problem, we perform a series of studies to challenge the validity of the guarantees associated with Wilks sampling. Specifically, we use Dakota to determine the number of random samples needed to represent the probability levels in Table 8.6 with 95% confidence. We then randomly sample this number of responses from the collection of 799 samples without replacement. We repeat this process a total of 1000 times and compute the fraction of samples in which the Wilks bound represents a maximum value for the quantity of interest exceeding the value reported in Table 8.6 at the same probability level. The resulting percentages are summarized in Table 8.7. In each case considered, the claimed theoretical confidence is achieved empirically.

Table 8.6: Wilks First-Order, One-Sided Probability Levels with > 99.9999% Confidence Level

Probability Level	Max Crud Thickness	Total Boron
0.70	112.22	7.4348e+04
0.75	127.33	9.1038e+04
0.80	138.47	1.0113e+05
0.85	149.74	1.1896e+05
0.90	164.77	1.3816e+05
0.95	179.82	1.6346e+05

Table 8.7: Numerical Validation of Wilks Sampling

Probability Level	Theoretical Confidence	Required Samples	Computed Confidence Max Crud	Computed Confidence Total Boron
0.70	0.95	9	0.962	0.956
0.75	0.95	11	0.959	0.956
0.80	0.95	14	0.974	0.973
0.85	0.95	19	0.952	0.959
0.90	0.95	29	0.960	0.959
0.95	0.95	59	0.953	0.957

Another useful aspect of Wilks UQ lies in being able to determine the possible order given a specified number of samples. Table 8.8 shows the orders possible near our current budget of 799 samples. This table indicates that for 799 samples, a 95/95 one-sided bound can be achieved using a 30th order Wilks analysis, while for a 99/99 one-sided bound this is reduced to second order Wilks. With this information, Wilks upper bounds for these cases derived from direct calculation and from constant trend Gaussian process (GP) surrogates are also provided in Table 8.8.

Table 8.8: Wilks One-Sided Orders for Given Sample Sizes

Probability Level	Confidence Level	Order	Required Samples	Direct Max Crud	GP Max Crud	Direct Total Boron	GP Total Boron
0.95	0.95	29	763				
0.95	0.95	30	786	183.79	182.73	1.7357e+05	1.7037e+05
0.95	0.95	31	809				
0.99	0.99	1	459				
0.99	0.99	2	662	214.13	204.92	2.4434e+05	2.0840e+05
0.99	0.99	3	838				

For the 95/95 Wilks analysis, the surrogate results are within 0.6% and 1.8% of the direct calculation results for maximum assembly crud thickness and total boron, respectively. Both the surrogate and direct results are conservative for both QoIs, as seen by comparing them with the 0.95 probability level results in Table 8.6. The discrepancies between surrogate and direct results jump to 4.3% and 14.7% in the 99/99 Wilks analysis. This likely occurs because for QoIs monotonic

with respect to parameter variations, extreme values will be found towards boundaries of the input domain. However, GP surrogates tend to perform worse in such regions often leading to larger biases and uncertainties. Unfortunately, more stringent coverage and confidence requirements are precisely the scenarios in which use of surrogates is desirable due to the increased sample size requirements of conducting such Wilks analyses. One possible solution, if the code allows, is to construct surrogates on a larger input domain than required by the input distributions. This will allow for improved surrogate quality in the region of input space of greatest relevance as defined by the input distributions. This approach is more easily implemented when Wilks analysis is applied after Bayesian calibration, as the calibration distributions will generally be restricted to a region of input space much smaller in volume than the domain in which the code is robust to input perturbations.

Listing 8.2 shows the Dakota input file used to construct the GP surrogates for use in the Wilks analyses reported above. The file `dakota_pstudy.dat` (line 32) contains the results of running VERA on the 159-run Latin hypercube sample mentioned in the previous section, used as the training sample for building the surrogates. The file `cips_val_des.dat` (line 19) contains the 799-run Monte Carlo sample used in the Wilks analyses reported above. Surrogate predictions of maximum assembly crud thickness and total boron are written to the file `cips_gp_evals.dat` (line 7). The cross-validation RMSE values for the GP surrogate are approximately 2.5% of the observed ranges in both QoIs, while the validation RMSE values (computed on the 799-run Monte Carlo sample) are significantly higher at approximately 11.5%. This validation RMSE is greater than generally desired, and its effects are seen in the 99/99 Wilks results comparisons between surrogate and direct computations as discussed above. This stresses the importance of collecting a sufficient number of validation samples to test surrogate quality prior to using a surrogate in follow-on analyses.

Listing 8.2: Dakota input file for construction of Gaussian process surrogates used in Wilks analyses.

```
# Build and evaluate a Gaussian process emulator of VERA output
# at a user specified set of points

environment
  method_pointer = 'EvalSurrogate'
  tabular_graphics_data
    tabular_graphics_file = 'cips_gp_evals.dat'

# Method to perform evaluations of the emulator

method
  id_method = 'EvalSurrogate'
  model_pointer = 'SurrogateModel'

  # Verbose will show the type form of the surrogate model
  output verbose

  list_parameter_study
    import_points = 'cips_val_des.dat'

# Surrogate model specification
model
  id_model = 'SurrogateModel'
  surrogate global
    # GP model
    gaussian_process surfpack
      trend
        constant
    # compute and print diagnostics after build
    metrics 'rsquared' 'root_mean_squared'
      press
    import_points = 'dakota_pstudy.dat'

variables
  uniform_uncertain 15
  lower_bounds      287.0690  14.58013  0.9291875  0.9672338  0.99
                    0.99  0.99  0.99  0.99  0.99  0.99  0.99  0.99
                    0.99  0.99
  upper_bounds      295.2817  16.25264  1.0694308  1.0284815  1.01
                    1.01  1.01  1.01  1.01  1.01  1.01  1.01  1.01
                    1.01  1.01
  descriptors       'STATES/State_1/tinlet' 'STATES/State_1/pressure'
                    '*CORE/rated_power' '*CORE/rated_flow'
                    'k_Bthresh' 'k_Cpor' 'k_crud_solid' 'k_delta_r'
                    'k_fac' 'k_Hc' 'k_kp2' 'k_mit0' 'k_Nc' 'k_rc'
                    'k_Tsat'

responses
  response_functions = 2
  descriptors = 'max_crud_thickness' 'total_boron'
  no_gradients
  no_hessians
```

Appendix A

General Linear Model Verification Test Suite

This appendix provides additional technical details for the linear model introduced in Section 2.2. Here we describe in detail a particular class of problems for which results of the QUESO DRAM or Dakota DREAM sampling algorithm can be verified against analytical solutions. Specifically, we present standard results for Bayesian analysis of the linear regression model ([12], pp. 233-265). In linear regression, a N-vector of outputs y is related linearly to functions of inputs x,

$$y = G\beta + \epsilon\,,$$

where the i-th row of G contains the evaluation of these regression functions at input x_i corresponding to the i-th datum y_i, β denotes the regression coefficients, and ϵ denotes the vector of observational errors. In the following, observational errors will be assumed mean-zero Gaussian, having variance $(1/\lambda)$ and possibly correlated with parametric dependencies goverened by a parameter ϕ.

Marginal posterior distributions for β, λ, and ϕ are derived analytically for three increasingly challenging verification scenarios: (i) β unknown, (λ, ϕ) fixed; (ii) (β, λ) unknown, ϕ fixed; and (iii) (β, λ, ϕ) unknown. In the third scenario, the marginal posterior distributions of β, λ, and ϕ do not belong to a standard class of probability distributions (such as Gaussian or Gamma, for example), thus requiring the use of numerical methods such as quadrature to accurately estimate their normalizing constants.

For each verification scenario, QUESO samples from the applicable marginal posterior distributions for β, λ, and ϕ will be compared with the corresponding analytical results. Convergence of posterior means and other summary statistics to corresponding parameter values assumed for data generation can also be monitored.

Section A.1 describes how data are simulated and presents the three verification scenarios of interest with corresponding analytical results for the desired posterior distributions in each scenario. Section A.2 defines the correlation functions used in the examples of Section 6.2.6. Section A.3 summarizes the hypothesis testing framework based on an energy distance statistic ([47]) used for verifying that DRAM and DREAM samples are distributed correctly.

A.1 Verification Scenarios

Let ε_t denote a mean-zero Gaussian stochastic process having covariance function $c(t_1, t_2|\phi) = (1/\lambda)\, r(t_1, t_2|\phi)$, where $\lambda > 0$ and $r(\cdot, \cdot|\phi)$ is a correlation function for $\phi \in \Phi$. We assume the

following quantities are specified:

1. Nominal parameter settings $(\beta_0, \lambda_0, \phi_0)$ with $\beta_0 \in \Re^{N_\beta}$, $\lambda_0 > 0$, and $\phi_0 \in \text{int}(\Phi)$
2. Indices $\{t_1, t_2, \ldots, t_N\}$.

We denote the multivariate Gaussian distribution having location vector μ and covariance matrix Σ by $\mathcal{N}(\mu, \Sigma)$. For specified N, generate a N-vector of errors

$$(\varepsilon_{t_1}, \ldots, \varepsilon_{t_N}) \sim \mathcal{N}\left(0_N, (1/\lambda_0) R(\phi_0)\right) ,$$

where 0_N is the N-vector of zeroes and the (i, j) element of $R(\phi_0)$ is given by $r(t_i, t_j|\phi_0)$. To complete the data generation process, we sample M-dimensional covariates $\{X_1, \ldots X_N\}$ independently from the distribution $\mathcal{N}(0_M, C)$, where C is a fixed covariance matrix. The i-th datum is calculated as $y_{t_i} = g^T(x_i)\beta_0 + \epsilon_{t_i}$, where $g(\cdot)$ is a N_β-dimensional regression function. In vector-matrix form,

$$y = G\beta_0 + \epsilon \,,$$

where

$$\begin{aligned} y &= (y_{t_1}, y_{t_2}, \ldots, y_{t_N})^T \,, \\ G &= [\, g(x_1)\ g(x_2)\ \cdots\ g(x_N)\,]^T \text{, and} \\ \epsilon &= (\epsilon_{t_1}, \epsilon_{t_2}, \ldots, \epsilon_{t_N})^T \,. \end{aligned}$$

To generate $\tilde{N}$ additional responses, we sample:

1. $\{X_{N+1}, \ldots, X_{N+\tilde{N}}\}$ independently from the distribution $\mathcal{N}(0_M, C)$
2. $(\varepsilon_{t_{N+1}}, \ldots, \varepsilon_{t_{N+\tilde{N}}})^T \sim \mathcal{N}\left(0_{\tilde{N}}, (1/\lambda_0)\left(\tilde{R}(\phi_0) - \bar{R}(\phi_0) R^{-1}(\phi_0) \bar{R}^T(\phi_0)\right)\right)$,

where the (i, j) element of $\tilde{R}(\phi_0)$ is given by $r(t_{N+i}, t_{N+j}|\phi_0)$, and the (i, j) element of $\bar{R}(\phi_0)$ is given by $r(t_{N+i}, t_j|\phi_0)$. Then $y_{t_{N+i}} = g^T(x_{N+i})\beta_0 + \epsilon_{t_{N+i}}$ and the matrix-vector form of the augmented data set follows. Note that this process of conditionally sampling errors preserves the correct joint disribution of the augmented data vector.

In the following calculations, we assume the regression matrix G is fixed, so this will not be explicitly denoted in the notation. Although (β, λ, ϕ) are fixed at $(\beta_0, \lambda_0, \phi_0)$ to generate data as above, our statistical analyses will assume some or all of $(\beta_0, \lambda_0, \phi_0)$ are unknown. The sampling distribution $f(Y|\beta, \lambda, \phi)$ of a random data set Y of size N is $\mathcal{N}\left(G\beta, (1/\lambda) R(\phi)\right)$.

(**Case 1**.) The first verification scenario assumes that λ and ϕ are fixed at λ_0 and ϕ_0, respectively. The Bayesian analysis places a prior distribution on β, which is given by the following:

1. $\pi(\beta)$ is $\mathcal{N}(\mu_0, \lambda_0^{-1}\Sigma^{-1})$.

Let $\hat{\beta}(\phi_0) = \left(G^T R^{-1}(\phi_0)\, G\right)^{-1} G^T R^{-1}(\phi_0)\, y$ be the generalized least squares estimate of β. The posterior distribution of β is $\mathcal{N}\left(\mu_1(\phi_0), \lambda_0^{-1}\Sigma_1(\phi_0)\right)$, where

$$\begin{aligned} \Sigma_1^{-1}(\phi_0) &= \Sigma + G^T R^{-1}(\phi_0)\, G \text{ and} \\ \mu_1(\phi_0) &= \Sigma_1(\phi_0)\left[\left(G^T R^{-1}(\phi_0)\, G\right)\hat{\beta}(\phi_0) + \Sigma\mu_0\right] . \end{aligned}$$

A noninformative prior for β, $\pi(\beta) \propto 1$, results in a posterior distribution of β as above, with

$$\mu_1(\phi_0) = \hat{\beta}(\phi_0) \text{ and } \Sigma_1(\phi_0) = \left(G^T R^{-1}(\phi_0)\, G\right)^{-1} . \tag{A.1}$$

The predictive distribution of Q future responses $\tilde{Y} = (Y_{\tilde{t}_1}, Y_{\tilde{t}_2}, \ldots, Y_{\tilde{t}_Q})^T$ associated with regression matrix $\tilde{G} = [\, g(\tilde{x}_1)\ g(\tilde{x}_2)\ \cdots\ g(\tilde{x}_Q)]^T$ and the Q-variate error vector $\tilde{\varepsilon}_Q = (\varepsilon_{\tilde{t}_1}, \varepsilon_{\tilde{t}_2}, \ldots, \varepsilon_{\tilde{t}_Q})^T$ is $\mathcal{N}\left(\tilde{\mu}(\phi_0), \lambda_0^{-1}\tilde{\Sigma}(\phi_0)\right)$, where

$$\begin{aligned}
\tilde{\mu}(\phi_0) &= \tilde{G}\mu_1(\phi_0) + \bar{R}(\phi_0)R^{-1}(\phi_0)\,(y - G\mu_1(\phi_0)) \text{ and} \\
\tilde{\Sigma}(\phi_0) &= \tilde{R}(\phi_0) - \bar{R}(\phi_0)R^{-1}(\phi_0)\bar{R}^T(\phi_0) + \tilde{H}(\phi_0)\Sigma_1(\phi_0)\tilde{H}(\phi_0)^T
\end{aligned}$$

for $\tilde{H}(\phi_0) = \tilde{G} - \bar{R}(\phi_0)R^{-1}(\phi_0)\, G$. Here, the (i,j) element of $\tilde{R}(\phi_0)$ is given by $r(\tilde{t}_i, \tilde{t}_j|\phi_0)$, and the (i,j) element of $\bar{R}(\phi_0)$ is given by $r(\tilde{t}_i, t_j|\phi_0)$.

(**Case 2**.) The second verification scenario assumes that ϕ is fixed at ϕ_0. The Bayesian analysis places a prior distribution on (β, λ), which is given by the following:

1. $\pi(\beta|\lambda)$ is $\mathcal{N}(\mu_0, \lambda^{-1}\Sigma^{-1})$, and
2. $\pi(\lambda)$ is Gamma(a, b).

The posterior distribution of λ is Gamma$(a_1, b_1(\phi_0))$, where

$$\begin{aligned}
a_1 &= (2a + N)/2 \\
b_1(\phi_0) &= \left(2b + \left(y - G\hat{\beta}(\phi_0)\right)^T R^{-1}(\phi_0)\left(y - G\hat{\beta}(\phi_0)\right)\right. \\
&\qquad \left. + \left(\hat{\beta}(\phi_0) - \mu_0\right)^T \Sigma_2^{-1}(\phi_0)\left(\hat{\beta}(\phi_0) - \mu_0\right)\right)/2
\end{aligned}$$

for $\Sigma_2(\phi_0) = \Sigma^{-1} + \left(G^T R^{-1}(\phi_0)\, G\right)^{-1}$.

A noninformative prior for β, $\pi(\beta) \propto 1$, results in the posterior distribution of λ given above, with

$$a_1 = (2a + N - N_\beta)/2 \text{ and } b_1(\phi_0) = \left(2b + \left(y - G\hat{\beta}(\phi_0)\right)^T R^{-1}(\phi_0)\left(y - G\hat{\beta}(\phi_0)\right)\right)/2\,. \tag{A.2}$$

A noninformative prior for λ, $\pi(\lambda) \propto (1/\lambda)$, results from taking $a = b = 0$. Note that $\pi(\beta, \lambda) \propto (1/\lambda)$ is the Jeffreys noninformative prior.

We denote the d-variate t distribution having ν degrees of freedom, location vector μ, and scale matrix Σ by $\mathcal{T}_d(\nu, \mu, \Sigma)$. The mean of this distribution is μ if $\nu > 1$ and the covariance matrix of this distribution is $\nu\Sigma/(\nu - 2)$ if $\nu > 2$. The posterior distribution of β is given by

$$\pi(\beta|y) \text{ is } \mathcal{T}_{N_\beta}\left(2a_1, \mu_1(\phi_0), b_1(\phi_0)\Sigma_1(\phi_0)/a_1\right) . \tag{A.3}$$

For the noninformative prior $\pi(\beta) \propto 1$, the quantities $\mu_1(\phi_0)$, $\Sigma_1(\phi_0)$ from (A.1) and a_1, $b_1(\phi_0)$ from (A.2) are utilized (A.3) for the posterior distribution of β, where again $a = b = 0$ for $\pi(\lambda) \propto (1/\lambda)$.

The predictive distribution of $\tilde{Y}$ is given by

$$\pi(\tilde{Y}|y) \text{ is } \mathcal{T}_Q\left(2a_1, \tilde{\mu}(\phi_0), b_1(\phi_0)\tilde{\Sigma}(\phi_0)/a_1\right) .$$

(**Case 3**.) The third verification scenario allows ϕ to be random. The Bayesian analysis places a prior distribution on (β, λ, ϕ), which is given by $\pi(\beta, \lambda, \phi) = \pi(\beta, \lambda)\pi(\phi)$, where $\pi(\beta, \lambda)$ is specified as in the previous scenario. The form of $\pi(\phi)$ used in verification testing will be provided in the following section. Our goal in this scenario is to numerically approximate the marginal posterior distributions of λ and β:

$$\pi(\lambda|y) = \int_\Phi \pi(\lambda|y,\phi)\,\pi(\phi|y)\,d\phi$$

$$\pi(\beta|y) = \int_\Phi \pi(\beta|y,\phi)\,\pi(\phi|y)\,d\phi\,.$$

The distributions $\pi(\lambda|y,\phi)$ and $\pi(\beta|y,\phi)$ are given analytically in the previous scenario. That leaves $\pi(\phi|y)$, which is given as

$$\pi(\phi|y) \propto \frac{\pi(\phi)}{b_1(\phi)^{a_1}\det\left(R(\phi)\right)^{1/2}\,\det\left(G^T R^{-1}(\phi)\,G\right)^{1/2}\,\det\left(\Sigma_2(\phi)\right)^{1/2}}\,.$$

Quadrature is used to compute the normalizing constant

$$c(y) = \int_\Phi \frac{\pi(\phi)\,d\phi}{b_1(\phi)^{a_1}\det\left(R(\phi)\right)^{1/2}\,\det\left(G^T R^{-1}(\phi)\,G\right)^{1/2}\,\det\left(\Sigma_2(\phi)\right)^{1/2}}$$

so that

$$\pi(\phi|y) = \frac{c^{-1}(y)\,\pi(\phi)}{b_1(\phi)^{a_1}\det\left(R(\phi)\right)^{1/2}\,\det\left(G^T R^{-1}(\phi)\,G\right)^{1/2}\,\det\left(\Sigma_2(\phi)\right)^{1/2}}\,.$$

For the noninformative prior $\pi(\beta) \propto 1$,

$$\pi(\phi|y) = \frac{c^{-1}(y)\,\pi(\phi)}{b_1(\phi)^{a_1}\det\left(R(\phi)\right)^{1/2}\,\det\left(G^T R^{-1}(\phi)\,G\right)^{1/2}}$$

for

$$c(y) = \int_\Phi \frac{\pi(\phi)\,d\phi}{b_1(\phi)^{a_1}\det\left(R(\phi)\right)^{1/2}\,\det\left(G^T R^{-1}(\phi)\,G\right)^{1/2}}\,,$$

where a_1, $b_1(\phi)$ are taken from (A.2) and $a = b = 0$ for $\pi(\lambda) \propto (1/\lambda)$.

The predictive distribution

$$\pi(\tilde{Y}|y) = \int_\Phi \pi(\tilde{Y}|y,\phi)\,\pi(\phi|y)\,d\phi$$

of $\tilde{Y}$ can also be numerically approximated.

A.2 Correlation Functions

Our verification examples consider two specifications of the correlation function $r(t_1, t_2|\phi)$:

1. $r(t_i, t_j) = \delta_{t_i,t_j}$
2. $r(t_i, t_j|\phi) = \phi^{|i-j|}$, $-1 < \phi < 1$.

The first case describes the standard regression setting in which errors are uncorrelated. The second case describes an AR(1) correlation structure for errors associated with observations indexed by time. The pairwise correlation between errors decays as a function of separation in time.

Both correlation functions admit explicit expressions for the inverse $R^{-1}(\phi)$ and determinant $\det(R(\phi))$,

1. $\mathcal{I}_N$ and 1
2. $\frac{1}{1-\phi^2}$
$$\begin{pmatrix} 1 & -\phi & 0 & 0 & \cdots & 0 & 0 \\ -\phi & 1+\phi^2 & -\phi & 0 & \cdots & 0 & 0 \\ 0 & -\phi & 1+\phi^2 & -\phi & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1+\phi^2 & -\phi \\ 0 & 0 & 0 & 0 & \cdots & -\phi & 1 \end{pmatrix} \text{ and } (1-\phi^2)^{N-1} \ ,$$

where $\mathcal{I}_N$ is the $N \times N$ identity matrix.

A.3 Energy Test of Equal Distributions

We utilize a nonparametric multisample test for equality of multivariate distributions based on an energy statistic [47] as a quantitative technique for stochastic verification of posterior sampling methods. Although the test can be applied to an arbitrary number of sample sets, we restrict our attention to the case of two sample sets, S_1 and S_2. In the context of stochastic verification, we take S_1 to be a sample derived from the target distribution of the sampling method we are testing, and S_2 to be a sample taken from the reference analytical posterior distribution. The null hypothesis of this test is that the two distributions are equal, and it is based on the *e-distance* statistic $e(S_1, S_2)$ defined by

$$e(S_1, S_2) = \frac{n_1 n_2}{n_1 + n_2}[2M_{12} - M_{11} - M_{22}] \,,$$

where n_1 and n_2 are the sizes of sample sets S_1 and S_2 respectively,

$$M_{ij} = \frac{1}{n_i n_j} \sum_{p=1}^{n_i} \sum_{q=1}^{n_j} ||Y_{ip} - Y_{jq}|| \,,$$

$||\cdot||$ denotes Euclidean norm, and Y_{ip} denotes the p-th observation in the i-th sample.

The null distribution of the e-distance statistic will not generally be available in closed form, and so the energy test is implemented as a permutation test. We take extremely small values of the resulting estimated p-value, say below 0.01, as strong evidence that the two distributions are *not* equal.

We require samples from the reference analytical posterior distribution for comparison with MCMC samples obtained from Dakota. To this end, we note that

$$\pi(\beta, \lambda, \phi|y) = \pi(\beta|y, \lambda, \phi)\, \pi(\lambda|y, \phi)\, \pi(\phi|y) \,.$$

Given ϕ (sampled from $\pi(\phi|y)$ in Case 3 and fixed at ϕ_0 in Cases 1 and 2), λ is sampled from its (conditional) posterior Gamma distribution having parameters specified in the discussion of Case 2 in Appendix A.1. Finally, β is sampled from $\pi(\beta|y, \lambda, \phi)$, which is a $\mathcal{N}(\mu_1(\phi), (1/\lambda)\Sigma_1(\phi))$ distribution. The resulting sample (β, λ, ϕ) is therefore generated from the joint distribution $\pi(\beta, \lambda, \phi|y)$ as desired.

Appendix B

Procedure for Running COBRA-TF Studies

This section provides a brief description of how to perform the COBRA-TF studies described in Chapter 7 with the model detailed in Section 2.3 on one of the CASL compute machines using the VERA software environment. This assumes that the steps necessary to log on to one of the various CASL machines, e.g. u233, u235, boris, natasha, anasova, etc., have been successfully completed.

The first step to create a VERA project is to clone (e.g. checkout) the top-level VERA source code,

```
git clone git@casl-dev:/VERA
```

You next clone all the CASL repositories you have access to using a VERA utility `clone_vera_repos.py` which keys off of your particular group memberships and access permissions, e.g.

```
cd VERA
./clone_vera_repos.py
```

Much more information on setting up a VERA project can be found here: `https://vminfo.casl.gov/trac/casl_phi_kanban/wiki/CASLDevLinuxDevEnv#vera_dev_quickstart`.

For the Cobra-TF studies of Chapter 7, you need the following project directory structure at a minimum:

```
VERA
|-- COBRA-TF
|-- DakotaExt
|     |-- Dakota
|-- DataTransferKit
|-- PSSDriversExt
|-- TeuchosWrappersExt
|-- TriBITS
|-- Trilinos
|-- VERAInExt
|-- VUQDemos
```

After the project has been created, it can be built in a variety of ways. Perhaps the easiest approach and the one guaranteed to work is to make use of the VERA checkin-test utility script. This should be done within a build directory separate from the project source directory created above. For example, to build the default (MPI-based) version of the project the steps shown in Listing B.1 would be followed.

Listing B.1: Steps to build a CASL VUQ project on one of the CASL machines.

```
# Create a build directory
mkdir -p CASL_BUILDS/VUQ_BUILD
cd CASL_BUILDS/VUQ_BUILD

# Create a link to the checkin-test utility script in the source project directory
ln -s ~/VERA/cmake/ctest/drivers/fissile4/checkin-test-vera.sh

# Invoke the script to build an optimized serial version of the project
./checkin-test-vera.sh --extra-repos-type=Nightly --test-categories=HEAVY \
  --enable-packages=VUQDemos --local-do-all
```

This will perform a sequence of steps which include configuring (i.e. creating Makefiles), building (i.e. invoking `make`) and testing the project (i.e. invoking `ctest`). The option `test-categories=HEAVY` enables the most comprehensive testing and can include some tests that require on the order of an hour to complete. A list of tests can be obtained by going into the subdirectory corresponding to the particular build, e.g. MPI_RELEASE_DEBUG_SHARED, and invoking `ctest -N`, and a particular test can be run by specifying its name or regular expression encompassing its name, e.g. `ctest -R single_assembly`. If the project source directory is created in a location other than the top-level user home directory, i.e. `~/VERA`, then its location must be specified in the invocation of the checkin-test script by adding the option, `-src-dir=$PATH_TO_TOP_LEVEL_VERA_DIR`, and the path must be absolute, e.g. `/home/$USER_ID/Projects/CASL/VERA`. Finally, it should be noted that the checkin-test utility script currently requires that it be invoked from the second level of a build directory tree. Any build directory structure of the following form would work, `~/SOME_LEVEL_1_DIR/LEVEL_2_DIR` with the script residing and invoked from the LEVEL_2_DIR.

Appendix C

ROMUSE2.0 User Manual

C.1 Introduction

ROMUSE (Reduced Order Modeling Based Uncertainty/Sensitivity Estimator) is a C++ based analysis code that is designed mainly to be used in conjunction with reactor analysis codes (e.g. reactor core simulators) to perform different forms of mathematical analysis on the simulator of interest (e.g. uncertainty quantification, surrogate model construction, and subspace analysis). ROMUSE interfaces with the I/O of the simulator of interest such that the I/O data are wrapped, modified and then used in ROMUSE modules to make conclusions about the problem of interest (refer to Figure C.1).

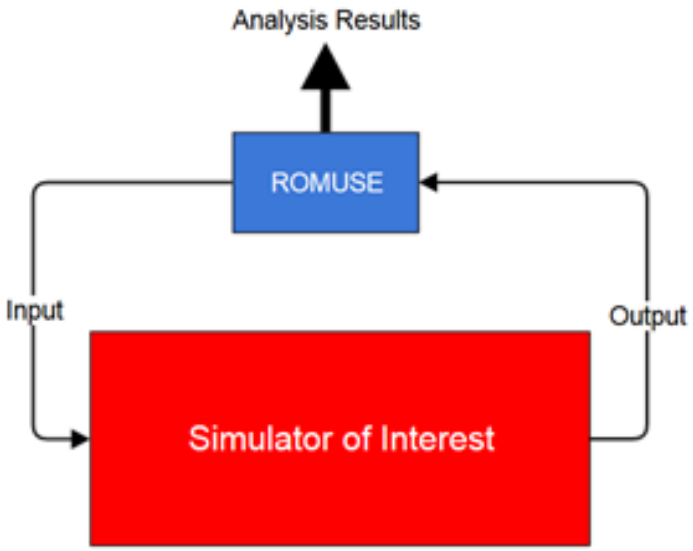


Figure C.1: ROMUSE General Scheme

Currently, ROMUSE is compatible with SCALE6.1, VERA-CS and any code with Hierarchical Data Format 5 (HDF5) I/O format. Depending on the simulator of interest, the ROMUSE input card requires various types of information. Table C.1 summarizes the main input card parameters. ROMUSE can read, manipulate and write input parameters. Moreover, response level analysis is

possible by executing the simulator of interest and then interpreting and analyzing the responses of interest.

In order to provide flexibility in using ROMUSE, it can be used at different levels (refer to Figure C.2); for example, it can be used to generate perturbed cases only, without running the simulator of interest (refer to the next section). Next, the user can command ROMUSE to run the simulator and collect the corresponding responses of interest. Finally, ROMUSE can be interfaced with Dakota 6.4 (6.1+) [1] and have access to the algorithms therein. In the latter case ROMUSE will compute the quantities needed for the problem of interest and provide them to Dakota.

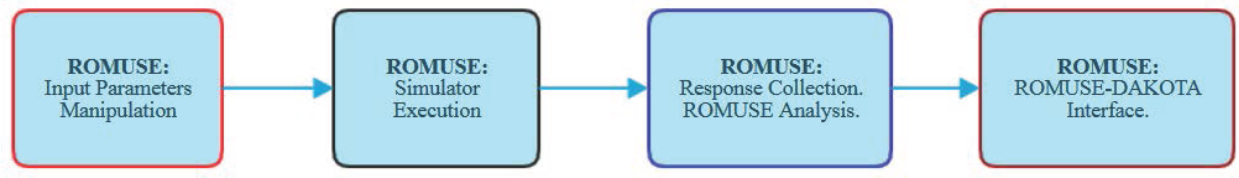


Figure C.2: ROMUSE Execution Levels

ROMUSE provides various sequences, each of which has different parameters and performs different algorithms or execution flows. Table C.1 summarizes the main input card parameters with a brief description. These parameters will be discussed further in subsequent sections. Comprehensive Uncertainty Quantification (UQ) studies can be performed via ROMUSE. Several UQ algorithms are available:

1. Brute force Monte Carlo UQ.
2. Karhunen-Lo'eve (KL) expansion based UQ (might appear as the Multi-Physics Efficient Range Finding Algorithm (MP-EUQ) in the case of coupled multi-physics codes).
3. Surrogate Based UQ (SBUQ).
4. Dakota based UQ and DA studies.

For more information about the first 3 options, refer to [26]. Options 2-4 require performing dimension reduction, which can be computed via ROMUSE, utilizing the efficient algorithms for single physics and multi-physics dimension reduction detailed in [26]. These algorithms are then used to perform dimension reduction on the uncertainty source space (i.e. revealing the active or important degrees of freedom (DoFs)). Once the active DoFs are determined they can be used to perform linear KL-based UQ, surrogate construction, or communicated to Dakota for additional analyses. Note that the material (nuclei) and reaction IDs used throughout ROMUSE follow the notation used by the SCALE covariance and cross-section library [54].

Table C.1: ROMUSE Input Card Main Parameters

Parameter	Description
SEQUENCE=	The sequence name: X-ROMUSE-UQ; X={GMODEL,SCALE,VERA} X-ROMUSE-RFA; X={GMODEL,SCALE,VERA} ROMUSE-EXE; ROMUSE-RESPONSES; ROMUSE-DAKOTA;

Ref:	Directory of reference case run.
Command:	The command used to execute the simulator of interest.
Queue Names:	Queues to be used for executing the commands and jobs submission.
Max Model Runs:	The number of model snapshots used for the analysis.
Perturbation Distribution:	Type of sampling distribution: Uniform or Normal.
Perturbation Range:	The range of perturbation (relative standard deviation in the case of normal distribution).
Number of Parameters:	The number of parameters to be perturbed (in the case of GMODEL).
Number of Responses:	The number of responses to be analyzed (always required if ROMUSE is used for more than input perturbation).
END	A flag to end the sequence information.
PoI:	Parameters of Interest: Two lines come after this, the first is the parameters file name and the second is the database path to the parameters (required in the case of GMODEL).
SCALE-PoI:	Parameters of Interest: If all parameters to be perturbed, `ALL` must be placed in the next line; otherwise, `Materials:` and `Reactions:` parameters must be used. Used with SCALE.
VERA-PoI :	Parameters of Interest: If all parameters are to be perturbed, `ALL` must be placed in the next line; otherwise, `Materials:` and `Reactions:` parameters must be used. Used with VERA.
RoI:	Response of Interest: Two lines come after this, the first is the response file name and the second is the database path to the response (required in the case of GMODEL).
Given Perts File:	The name of the file containing the given perturbations.
Library Name:	The name of the formatted library (required in the case of SCALE, VERA).
Covariance Directory:	The directory containing the unformatted covariance data.
S-SPACE=	Output the basis (default = ON) {ON,OFF}
EUQ=	Perform KL-Based UQ or MP-EUQ (default=OFF) {ON,OFF}
SMCUQ=	Perform Surrogate Based MC UQ (default=OFF) {ON,OFF}
MCUQ=	Perform brute force MC UQ (default=ON) {ON,OFF}

C.2 System Requirements and Specifications

ROMUSE is a C++ code compiled and tested via gcc version 4.8.3 on Linux Red Hat 4.8.3-9 (64 bit). All the tests are performed on CentOS Linux release 7.1.1503 available on NCSU HPC. The ROMUSE package is distributed with the required Boost, Eigen, and HDF5 libraries. Moreover, ROMUSE comes with the required covariance library, which is based on the SCALE covariance library `44groupcov` formatted in a binary file and a text formatted library [54].

Once in the ROMUSE root directory, the user can compile the code by first accessing the Makefile and modifying the path and compiler parameters according to the system's structure and then typing

`make -B ROMUSE`. The structure of the ROMUSE problem must be such that all input files, tagged input files, reference libraries, and/or reference cases are in the same directory. The ROMUSE input file must be named `romuse.in` and the simulator input file must be named `input.txt`. Other files can be customized in the ROMUSE input file as will be shown later in this manual.

In the examples of subsequent sections, links are provided to directories distributed with ROMUSE that contain all the files necessary for users to run them.

C.3 Perturbing Input Parameters

C.3.1 Introduction

Perturbing input parameters is a vital feature for any analysis code, as many mathematical and statistical analyses such as uncertainty quantification and data assimilation require manipulating the parameter set. However, in order to build an analysis code that can work with a wide variety of simulators, the ROMUSE perturbation module is designed such that it allows the user to manipulate the input parameters of any given model as long as the parameters are HDF5 formatted, or in SCALE cross-section library binary format [54], or in VERA cross-section library format [49, 27]. The ability to perturb any HDF5 formatted parameters allows ROMUSE to work with any simulator supporting this I/O format (refer to Table C.2). ROMUSE has been built to serve nuclear reactor simulators, therefore special attention has been given to nuclear reactor physics problems, mainly simulated via SCALE [54, 30, 41] and VERA-CS [49]. The ROMUSE user can determine the type of perturbations required depending on the subsequent application. ROMUSE offers three types of possible perturbations:

1. Covariance Based Perturbations: This type of perturbation requires a covariance library to be provided so that perturbations consistent with the given covariance library can be generated. This perturbation type can be used to generate Monte Carlo samples for uncertainty quantification studies.

2. Given Perturbations: The user can provide his/her own perturbations to ROMUSE. This type of perturbation is very useful for verification and validation studies where users are studying the behavior of the model of interest over an interval in the parameter's domain.

3. Random Perturbations: ROMUSE can introduce random perturbations into the nuclear data cross-sections. The user must provide a range for the perturbations and choose a distribution to generate perturbations from. This type of perturbation is important for applications such as the subspace analysis, surrogate modeling and other important analysis types.

The three perturbation types mentioned above can be applied to any parameter (e.g. cross-sections). The only requirement is that the parameters must be in SCALE, VERA, or HDF5 input format.

Table C.2: Parameters Perturbation Input Card Entries

Parameter	Description
SEQUENCE=	The sequence name: Y-ROMUSE-X; {Y=SCALE,VERA,GMODEL} | {X=UQ,RFA,GP}
Ref:	Directory of reference case run.
Max Model Runs:	The number of model snapshots used for the analysis.
Library Name:	The name of the formatted library (required in the case of SCALE, VERA).

PoI:	Parameters of Interest: Two lines comes after this, the first is the parameters file name and the second is the database path to the parameters (required in the case of GMODEL).
SCALE-PoI:	Parameters of Interest: If all parameters are to be perturbed, `ALL` must be placed in the next line; otherwise, `Materials:` and `Reactions:` parameters must be used. Used with SCALE.
VERA-PoI :	Parameters of Interest: If all parameters are to be perturbed, `ALL` must be placed in the next line; otherwise, `Materials:` and `Reactions:` parameters must be used. Used with VERA.
Perturbation Distribution:	Type of sampling distribution: Uniform or Normal.
Perturbation Range:	The range of perturbation (relative standard deviation in the case of normal distribution).
Number of Parameters:	The number of parameters to be perturbed (in the case of GMODEL).

Materials:	The materials or nuclei to be perturbed. Example:
	`Materials:`
	`N Number of materials or nuclei`
	`For i=1,N`
	`Nucleus ID or Material ID`

Reactions:	The reactions to be perturbed. Example:
	`Reactions:`
	`M Number of Reactions`
	`For i=1,M`
	`Reaction ID Reaction ID`

Given Perts File:	The name of the file containing the given perturbations.
Covariance Directory:	The directory containing the unformatted covariance data.
END	A flag to end the sequence information.

C.3.2 SCALE Cross-Section Library Perturbation

The SCALE cross-section library `44groupcov` is formatted in a binary file which can be read via ROMUSE. In order to generate perturbed libraries for a SCALE case, ROMUSE requires the following information (refer to Table C.2):

1. Reference library path.
2. Type of perturbation: Depending on the perturbation type further requirements might be needed.
   (a) Covariance Based Perturbation: Covariance library path and the number of Monte Carlo samples.
   (b) Given Perturbation: The given perturbations file path.
   (c) Random Perturbation: The relative range of perturbations (e.g. $\pm 10\%$) and the number of samples.

Example Input Card (1)

In this example case (refer to Figure C.3), ROMUSE will generate 100 different cross-section libraries based on a covariance library stored in `/Path/To/Covariance/Files/`. Note that ROMUSE assumes that the user has already run a reference case which is located in `/Path/To/Reference/Case/`. ROMUSE will perturb all cross-sections with covariance data in the provided path. After running the card below with the correct reference case and library, a new directory called `SNAPS` with subfolders labeled `snap0`, ..., `snap99` will be created. Each of these subfolders will have a full case of a perturbed library imitating the format of the reference case.

Corresponding Example: `Manual/Examples/SCALE/UQ`

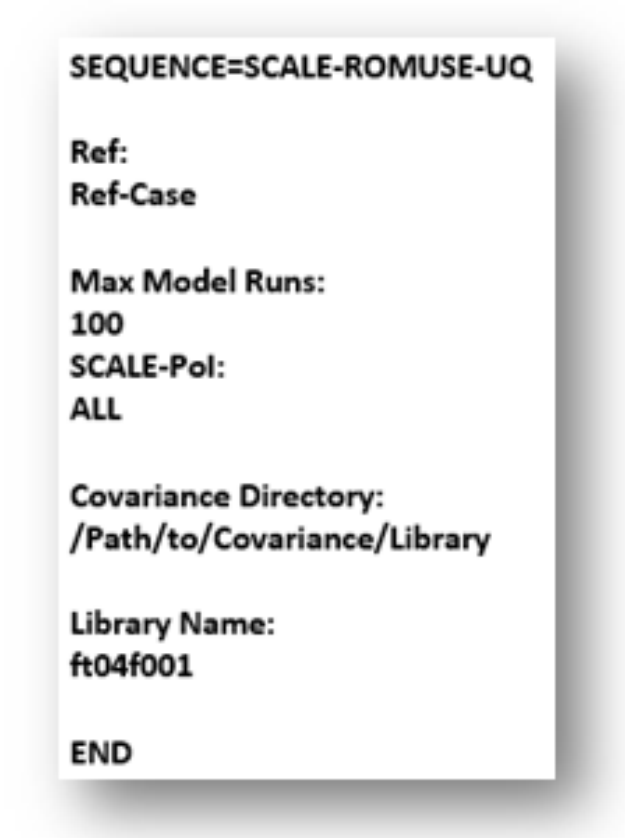


Figure C.3: SCALE-UQ Perturbation Input Card

Example Input Card (2)

In this example case (refer to Figure C.4), ROMUSE will generate 100 different perturbed cases. The perturbations are generated via a uniform distribution with ±10% of the reference cases. Note that ROMUSE assumes that the user has already run a reference case which is located in `/Path/To/Reference/Case/`. ROMUSE will perturb the specified cross-sections corresponding to two nuclei (U235 and U238) and three reactions (total cross-section, fission cross-section, and elastic scattering [ID: 1,18,2 respectively]). After running the card below with the right reference case and library, a new directory called `SNAPS` with subfolders labeled `snap0`, ..., `snap99` will be created. Each of these subfolders will have a full case of a perturbed library imitating the format of the reference case.

Corresponding Example: `Manual/Examples/SCALE/RFA`

```
SEQUENCE=SCALE-ROMUSE-RFA
Ref:
Ref-Case
Max Model Runs:
100
Perturbation Range:
0.1
Perturbation Distribution:
Uniform
SCALE-Pol:
Materials:
2
92235
92238
Reactions:
3
1
18
2
Library Name:
ft04f001
END
```

Figure C.4: SCALE-RFA Perturbation Input Card

C.3.3 VERA Cross-Section Library Perturbation

The VERA cross-section library is actually the MPACT covariance library [27]. ROMUSE is able to read this library into a vector and then manipulate it in a variety of ways. As mentioned in the previous section, the information required depends on the type of perturbation. Currently ROMUSE is compatible with the MPACT 47 group VERA library (refer to Table C.3). This library includes 295 different isotopes with different reaction cross-section data. But currently ROMUSE only has access to the 44 group covariance library [54]; as a consequence, ROMUSE has a mapping capability that is able to map perturbations from one group structure to another. Mapping the perturbations is performed via linear interpolation based on the assumption of constant lethargy intervals.

The perturbed libraries are generated via ROMUSE. ROMUSE reads the `44groupcov` covariance library available within the SCALE6.1 package and perturbs the 47 group MPACT cross-section library version 4. ROMUSE decomposes the 44-group matrix, creates 44-group perturbation factors, projects the 44-group perturbation factors to 47-group perturbation factors, then perturbs the 47-group cross-sections.

Based on the concept of a constant lethargy flux as the weight function, if the group structures do not align, ROMUSE determines the fractional lethargy width of the unaligned boundaries and uses these terms to map the cross-section perturbations from one group structure to another [22]. The following equation exemplifies how to map a perturbation from the A group structure to the

B group structure,

$$\Delta\sigma_i^B = \Delta\sigma_{j-1}^A + \frac{\Delta\sigma_j^A - \Delta\sigma_{j-1}^A}{\left|L\left(E_j^A\right) - L\left(E_{j-1}^A\right)\right|} \times \left|L\left(E_i^B\right) - L\left(E_{j-1}^A\right)\right|$$

where $\Delta\sigma_i^B$ is the perturbation associated with the i^{th} group in the B group structure and $\Delta\sigma_j^A$ is the computed perturbation associated with the j^{th} group in the A group structure. In this case $i \in (j-1, j)$ or $E_i^B \in \left(E_j^A, E_{j-1}^A\right)$. $L\left(E_j^A\right)$ is the lethargy term for energy group E_j^A. Note that when groups align the perturbations are not altered.

Table C.3: 47 Energy Group Structure

Group	Energy Boundary
1	20 MeV
2	6.0653 MeV
3	3.6788 MeV
4	2.2313 MeV
5	1.3534 MeV
6	0.8208 MeV
7	4.9787 MeV
8	0.1832 MeV
9	67.38 KeV
10	9.119 KeV
11	2.0347 KeV
12	0.13 KeV
13	78.9 eV
14	47.8512 eV
15	29.023 eV
16	13.71 eV
17	12.099 eV
18	8.3153 eV
19	7.33822 eV
20	6.47602 eV
21	5.715 eV
22	5.04348 eV
23	4.4509 eV
24	3.9279 eV
25	2.3824 eV
26	1.8554 eV
27	1.4574 eV
28	1.2351 eV
29	1.1664 eV

30	1.1254 eV
31	1.0722 eV
32	1.0137 eV
33	0.97100 eV
34	0.9099 eV
35	0.7821 eV
36	0.62506 eV
37	0.5032 eV
38	0.35767 eV
39	0.2705 eV
40	0.18443 eV
41	0.14572 eV
42	0.11157 eV
43	0.08197 eV
44	0.0569 eV
45	0.0428 eV
46	0.0306 eV
47	0.0124 eV

Example Input Card (1)

In this example case, ROMUSE will generate 100 different cross-section libraries based on the covariance library. Note that ROMUSE assumes that the user has already run a reference case which is located in `/Path/To/Reference/Case/`. ROMUSE will perturb all cross-sections available in the library using the covariance data in the covariance library. ROMUSE will generate a new directory called `SNAPS` with subfolders labeled `snap0`, ..., `snap99`. Each of these subfolders will have a full case of a perturbed library imitating the format of the reference case (refer to Figure C.5).

Corresponding Example: `Manual/Examples/VERA/UQ`

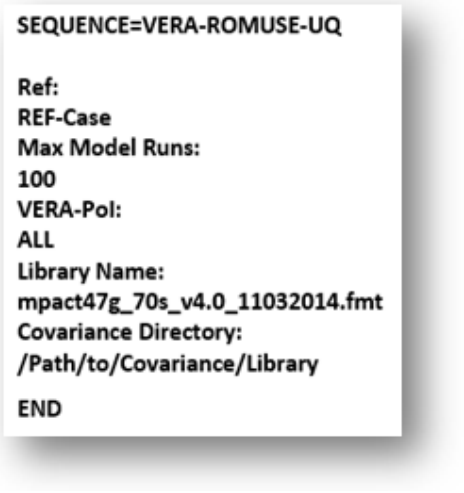


Figure C.5: VERA-UQ Input Card

C.3.4 General Parameter Perturbation

In addition to the perturbation modes introduced above, ROMUSE can manipulate the parameters of any simulator with HDF5 [28] input format. In this case the user should provide ROMUSE with the database address of the I/O data.

Example Input Card (1)

Assuming that we have a Python or Matlab based simulator, or any other simulator with HDF5 formatted I/O, then the following ROMUSE input card will allow manipulation of the input parameters. Running the card below (refer to Figure C.6) will generate 100 different perturbed input cases based on normal distributions having standard deviations 10% of the reference cases. Note that the user can use SEQUENCE=GMODEL-ROMUSE-RFA and then provide a covariance library. The samples will be Monte Carlo samples that can be used for uncertainty quantification. Moreover, note that ROMUSE assumes that the user has already run a reference case which is located at `/Path/To/Reference/Case/`. The HDF5 file name containing the parameters is `Params.h5` and the address of the parameters of interest is `/Params`. ROMUSE will generate a new directory called `SNAPS` with subfolders labeled `snap0`, ..., `snap99`. Each of these subfolders will have a full case of a perturbed library imitating the format of the reference case (in HDF5 format).

Corresponding Example: `Manual/Examples/GMODEL/RFA`

```
SEQUENCE=GMODEL-ROMUSE-RFA
Ref:
/Path/To/Reference/Case
Max Model Runs:
100
Perturbation Distribution:
Normal
Perturbation Range:
0.1
PoI:
Params.h5
/Params
END
```

Figure C.6: GMODEL-RFA Input Card

C.4 Complex Sequences

Obviously, perturbing the input parameters must be followed by various steps depending on the goal of the parameter perturbation performed in the first place. Therefore, complex sequences are required. In the following subsection, new types of ROMUSE sequences are introduced. They are referred to as "Complex Sequences" due to their nature. These sequences consist of a combination of different types of sequences to perform certain tasks. The following subsections present a summary of two important types of complex sequences.

C.4.1 Execution and Response: ROMUSE-EXE and ROMUSE-RESPONSES

The ROMUSE-EXE sequence follows any parameter perturbation cases. Providing this sequence in the ROMUSE input file will execute the perturbation cases. Two key pieces of information must be provided: The command to be used for the execution, and the job submission details (refer to Table C.4). Note that for this sequence to work as expected, a prerequisite perturbation sequence must exist such that the `SNAPS` folder is already present.

The ROMUSE-RESPONSES sequence provides information about the responses of interest to be collected and analyzed. It is generally preferred to have the responses output in a database format (e.g. HDF5 VERA-CS format). However, in many cases such as SCALE and Dakota, the output is dumped into text files.

Table C.4: ROMUSE-EXE and ROMUSE-RESPONSES Input Card Parameters

Parameter	Description
SEQUENCE=	ROMUSE-EXE or ROMUSE-RESPONSES
Command:	`/path/to/command`

Queue Names:	Job submission information. Example:
	`N number of queues`
	`M number of cores per job`
	`For i=1:N`
	`Queue Name C: capacity of this queue`

Flags:	Single Physics={YES/NO} (Default NO) Multi Physics={YES/NO} (Default NO) (required for ROMUSE-RESPONSES)
Simulator Out:	Output-File-Name Response Tag (the tag to search for in a text formatted output file)
END	A flag to end the sequence information.

Example Input Card (1)

Running the card below (Figure C.7) will generate three different perturbed input cases based on the gradient free Range Finding Algorithm (RFA) using a uniform distribution having bounds ±20% of the reference values of U235 and U238 total for fission and elastic scattering cross-sections. Note that the user can select SEQUENCE=SCALE-ROMUSE-UQ and then provide a covariance library (refer to Figure C.8). In this case, the samples will be Monte Carlo samples that can be used for uncertainty quantification. Moreover, note that ROMUSE assumes that the user has already run a reference case which is located in Ref-CASE=`/Path/To/Reference/Case`. ROMUSE will generate a new directory called `SNAPS` with subfolders labeled `snap0`, ..., `snap2`. Each of these subfolders will have a full case of a perturbed library imitating the format of the reference case (in HDF5 format).

Corresponding Example: `Manual/Examples/SCALE/RFA-EXE`

```
SEQUENCE=SCALE-ROMUSE-RFA
Ref:
Ref-Case
Max Model Runs:
3
Perturbation Range:
0.2
SCALE-Pol:
Materials:
2
92235
92238
Reactions:
3
1
18
2
Library Name:
ft04f001
END

SEQUENCE=ROMUSE-EXE
Command:
/Path/to/SCALE/ batch6.1
Queue Names:
1
1
zeus 64
END

SEQUENCE=ROMUSE-RESPONSES
Simulator Out:
input.out
k-eff =
Flags:
Single Physics=YES
END
```

Figure C.7: ROMUSE-EXE and ROMUSE-RESPONSES Input Card (RFA)

Corresponding Example: `Manual/Examples/SCALE/UQ-EXE`

```
SEQUENCE=SCALE-ROMUSE-UQ
Ref:
Ref-Case
Max Model Runs:
50
SCALE-Pol:
Materials:
2
92235
92238
Reactions:
3
1
18
2
Covariance Directory:
/Path/to/Covariance
Library Name:
ft04f001
END

SEQUENCE=ROMUSE-EXE
Command:
/Path/To/SCALE/batch6.1
Queue Names:
1
1
gnep 92
END

SEQUENCE=ROMUSE-RESPONSES
Simulator Out:
input.out
k-eff =
Flags:
Single Physics=YES
END
```

Figure C.8: ROMUSE-EXE and ROMUSE-RESPONSES Input Card (UQ)

C.4.2 ROMUSE-DAKOTA

This sequence requires the presence of the two previous sequences. This sequence offers an interface between ROMUSE and any simulator (SCALE, VERA-CS and General Model). In the following subsection a full SCALE6.1 based example will be introduced; future versions of this manual will present additional models demonstrating use of ROMUSE with VERA-CS (refer to Figure C.9).

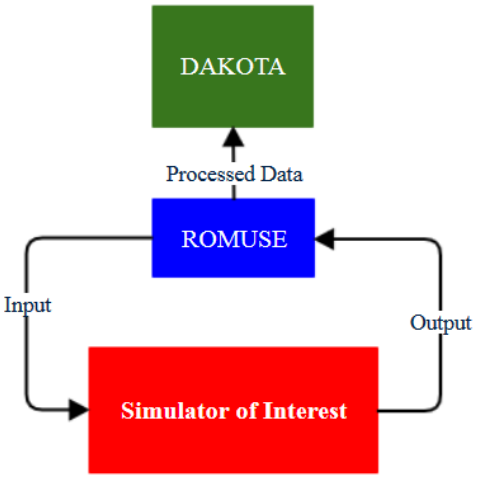


Figure C.9: ROMUSE-DAKOTA Interface Scheme

This interface is tricky, due to the fact that the required parameters depend on the Dakota method of interest. For more information about Dakota please refer to [1]. In order to solve this problem ROMUSE will adapt a flexible technique that is almost independent of the Dakota method used.

ROMUSE can compute many useful quantities related to the simulator of interest (SCALE, VERA-CS and General Model), as summarized in Table C.5. These quantities include parameter standard deviations, parameter means, response means, response standard deviations, parameter and response perturbations, sensitivity coefficients, and many others. The premise of this interface is to make any quantity read, computed or generated via ROMUSE available to Dakota such that the construction of any Dakota case will be easier through the ROMUSE input card parameters summarized in Table C.6. Therefore, development of a general interface between ROMUSE and Dakota depends on a Dakota input file provided by the user and placed in the same directory as the `romuse.in` file. However, this Dakota input must have special tags informing ROMUSE of the structure of the Dakota input file.

Table C.5: ROMUSE-DAKOTA Tags

Tag	Associated Quantity
`$np`	Number of Parameters
`$m`	Parameter Mean
`$std`	Parameter Standard Deviation (Absolute)
`$p`	Parameter Descriptors

`$tnr`	Number of Response Functions
`$ip`	Parameter Initial Point
`$lp`	Parameter Lower Bound
`$up`	Parameter Upper Bound

Table C.6: ROMUSE-DAKOTA Input Card Parameters

Tag	Associated Quantity
SEQUENCE=	ROMUSE-DAKOTA
Surrogate Type:	e.g. P-01, P-02, P-03, Gaussian-surfpack, Gaussian-dakota
Number of Validation Points:	Number of points for validation.
Max Number of Active Subspace Construction Points:	Maximum number of snapshots used for subspace construction.
Subspace Approximation Error Tolerance:	Error tolerance for the algorithm of interest.
Recycle Samples=Yes	Use same samples for building the subspace and for surrogate construction (not recommended).
Covariance Directory:	`/path/to/covariance/files`
Input File:	Name of Dakota input file with the tags.
Surrogate Data File:	Name of the file to store the surrogate construction data (this will be generated via ROMUSE).
Surrogate Challenge Data File:	Name of the file to store the surrogate challenge data (this will be generated via ROMUSE).
Observations File:	Name of the file to store the measurements and their uncertainties (in case the user wants to generate these measurements via ROMUSE).
Dakota Command:	The Dakota command.
END	End tag for the sequence.

Example Input Card (1)

This example builds a surrogate to parameterize the SCALE6.1 neutronics model (NEWT) [41] in terms of the fission cross-section of U235. The surrogate is then used to perform uncertainty quantification via Monte Carlo sampling and compared to the performance of the linearly estimated uncertainty (via SCALE6.1 SAMS module) [41].

In this example ROMUSE will be used to prepare a full surrogate-based uncertainty quantification study for a nuclear fuel pin cell problem simulated via SCALE6.1. First ROMUSE will generate 100 samples of the input parameters to be used in later sequences. Each perturbed case is then run via the SEQUENCE=ROMUSE-EXE sequence. Once the cases are run, the responses are col-

lected via the SEQUENCE=ROMUSE-RESPONSES sequence. Finally, SEQUENCE=ROMUSE-DAKOTA will construct the full Dakota case. A polynomial surrogate is used in the form represented by (C.1). Dakota estimates the coefficients (c_i) based on the given surrogate construction data,

$$\tilde{f}(\mathbf{x}) = c_0 + \sum_{i=1}^{M} c_i x_i + \sum_{i=1}^{M}\sum_{j\geq i}^{M} c_{ij} x_i x_j + \sum_{i=1}^{M}\sum_{j\geq i}^{M}\sum_{k\geq j}^{M} c_{ij} x_i x_j x_k \,, \tag{C.1}$$

where M is the number of input parameters. The challenge data are used to validate the fitted surrogate model. Different orders of polynomial surrogates require different minimum numbers of construction data points as discussed in Section 4.1. In particular, for linear, quadratic, and cubic orders respectively,

$$\begin{aligned} N_\beta &= M + 1 \\ N_\beta &= \frac{(M+1)(M+2)}{2} \\ N_\beta &= \frac{\left(M^3 + 6M^2 + 11M + 6\right)}{6} \,, \end{aligned}$$

where N_β is the number of required points. N_β can grow dramatically as M and the order of the polynomial increase. For more information on the available surrogate types refer to [1]. Since ROMUSE has the capability to perform dimension reduction, this example utilizes the gradient free RFA to find the basis of the active subspace in the uncertainty sources space so as to satisfy a provided error criterion.

This example uses a nuclear fuel pin cell model simulated via SCALE and the goal is to determine the uncertainty in the multiplication factor (k_{eff}) due to the fission cross-section of the U^{235} isotope. Figure C.10 shows a representation of the 2-dimensional geometry of the fuel pin cell. At the center (or the red region) is the fuel bullet which contains the uncertainty source in this example surrounded by a Helium gap (He) which is the green region. The blue region is the zirc4 alloy cladding, and finally the yellow region represents the borated water (H_2O and B) that serves as a coolant to extract the heat and as a moderator that slows down the neutrons such that they have a high probability of fissioning the U^{235} nuclei. Boron is added to the coolant to absorb excess neutrons that might lead to super-criticality ($k_{eff} > 1.0$) in the core system. In this example, a single fuel pin cell is extracted from the reactor core model, therefore the reference k_{eff} is 1.172437. Since this example uses the 44 group library and considers the uncertainty due to one cross-section type of one isotope, the total number of parameters is $1 \times 1 \times 44 = 44$ energy dependent parameters.

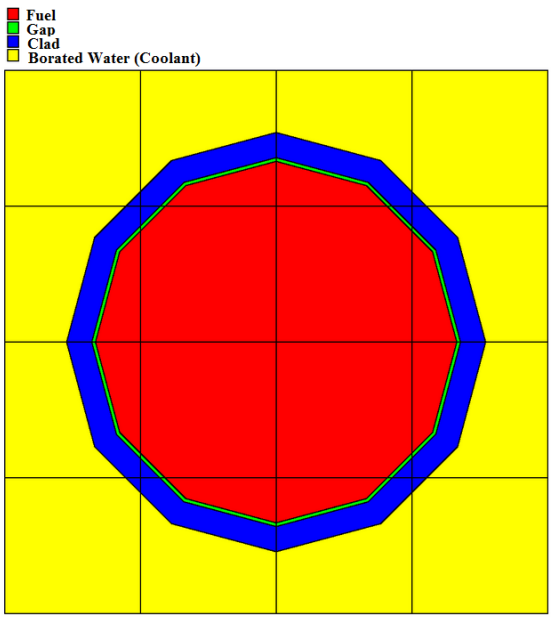


Figure C.10: SCALE Nuclear Fuel Pin Cell Model

First, 100 samples are generated, the ROMUSE-EXE sequence will run the corresponding cases, and ROMUSE-RESPONSES will collect the multiplication factor estimated via the perturbed cases. Finally, ROMUSE-DAKOTA will send the proper commands to build the subspace and pick the proper subspace for the provided error upper bound tolerance (in this case 10^{-2} or 1% of the error upper bound). Figure C.11 shows the error upper bound for this example. An expression for the error upper bound and how ROMUSE calculates it can be found in [26]. Figure C.12 shows the ROMUSE input (`romuse.in`). The tagged Dakota input file must be provided in the same directory as that of `romuse.in`. This file will instruct ROMUSE on the Dakota method and how to construct the Dakota case (refer to Figure C.13). After running this case ROMUSE will generate a folder named DAKOTA-CASE, which will have a full Dakota case ready to run by the user or the user can add another ROMUSE-EXE sequence to run the Dakota case directly from inside ROMUSE; however, it is recommended to check the case before running it.

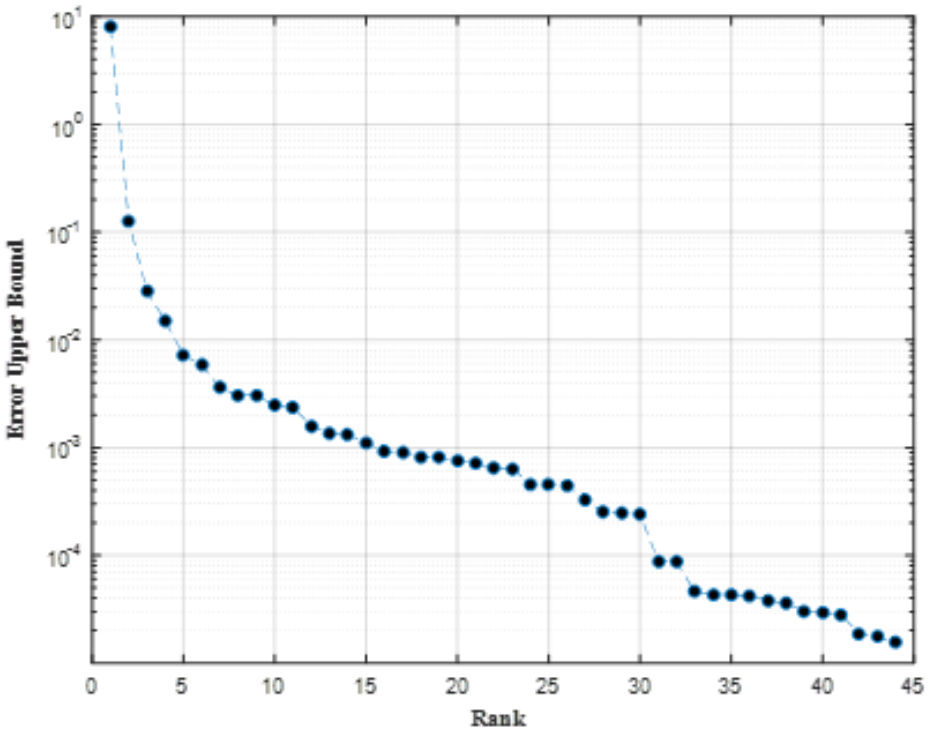


Figure C.11: Error Upper Bound Vs. Rank

Corresponding Example: `Manual/Examples/SCALE/DAKOTA/SURR-UQ`

```
SEQUENCE=SCALE-ROMUSE-UQ
Ref:
Ref-Case
Max Model Runs:
100
SCALE-PoI:
Materials:
1
92235
Reactions:
1
18
Library Name:
ft04f001
Covariance Directory:
/Path/to/Covariance
END
SEQUENCE=ROMUSE-EXE
Command:
/Path/to/SCALE/Command/batch6.1
Queue Names:
1
1
gnep 92
END
SEQUENCE=ROMUSE-RESPONSES
Simulator Out:
input.out
k-eff =
Flags:
Single Physics=YES
END
SEQUENCE=ROMUSE-DAKOTA
Input File:
dakota_romuse_scale.in
Surrogate Data File:
surrogate_build_pts.dat
Surrogate Challenge Data File:
surrogate_build_pts_ff.dat
Surrogate Type:
P-01
Number of Validation Points:
10
Max Number of Active Subspace Construction Points:
44
Subspace Approximation Error Tolerance:
0.01
Recycle Samples=Yes
Covariance Directory:
/Path/to/Covariance
END
```

Figure C.12: ROMUSE-DAKOTA Input Card

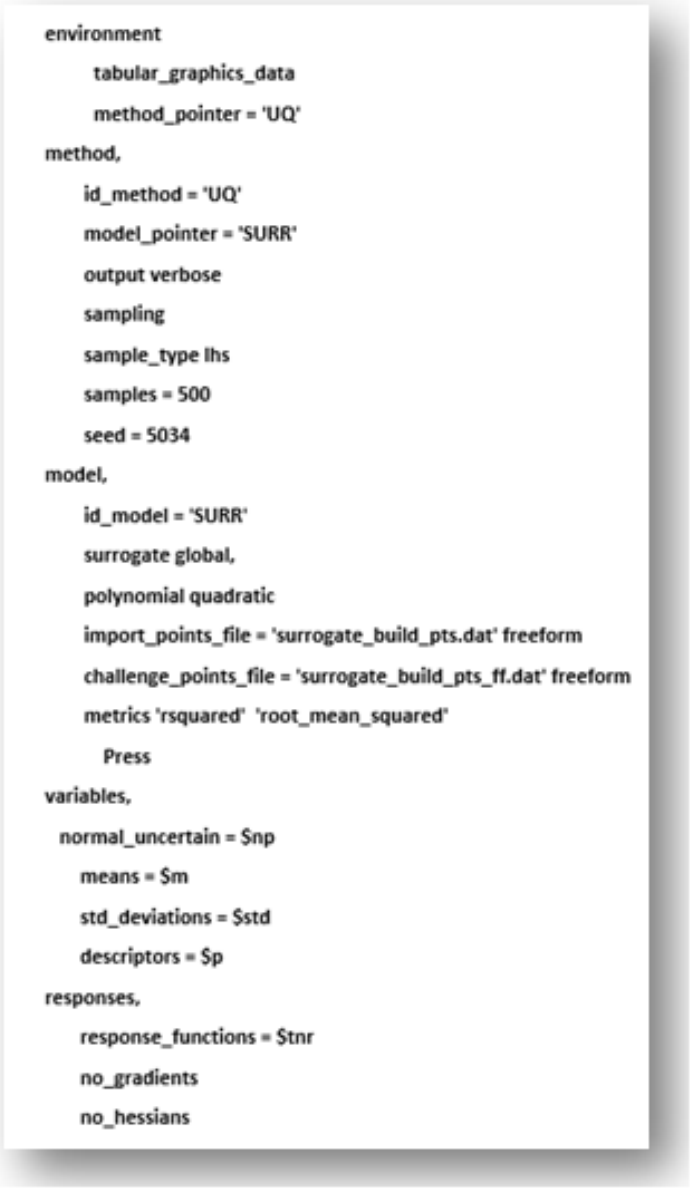


Figure C.13: Tagged Dakota Input File

As shown in Figure C.11, rank 5 achieves the error upper bound required. Therefore, instead of dealing with 44 parameters, ROMUSE identified only 5 important parameters in a different coordinate system represented by the basis. Once the basis is calculated it can be used to construct different surrogate types: linear (P-01), quadratic (P-02), cubic (P-03), and Gaussian process surfpack. Table C.7 compares the performance of the different surrogates with the linearly estimated uncertainties (sandwich equation) [26]. Table C.8 summarizes the error analysis results as reported by the Dakota output file for the second order polynomial and the Gaussian Process surrogate [1]. The leave-one-out cross-validation RMSPE and validation RMSPE (calculated on the challenge samples) are discussed in Section 7.3.2. The RMSPE values can be compared to the observed range in calculated k_{eff} values, which is approximately 1240 pcm.

Table C.7: Comparison of the uncertainty Estimated Via Different Surrogate Types

Method	Standard Deviation (pcm)
ROMUSE – linear	149
ROMUSE – non-linear	155
ROMUSE-Gaussian Process	166
Sandwich Equation	133

Table C.8: Sample Surrogate Error Analysis as Reported By Dakota

Surrogate Type	Cross-Validation RMSPE (pcm)	R-Squared Goodness of Fit	Validation RMSPE (pcm)
P-02	141	92%	17
Gaussian Process surfpack	290	88%	56

Bibliography

[1] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, J. P. Eddy, M. S. Ebeida, M. S. Eldred, R. W. Hooper, P. D. Hough, K. T. Hu, J. D. Jakeman, K. A. Maupin, J. A. Monschke, E. M. Ridgway, Ahmad Rushdi, L. P. Swiler, J. A. Stephens, D. M. Vigil, and T. M. Wildey. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.4 user's manual. Technical Report SAND2014-4633, Sandia National Laboratories, Albuquerque, NM, Updated May 2016. Available online from `http://dakota.sandia.gov/documentation.html`.

[2] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, J. P. Eddy, M. S. Ebeida, M. S. Eldred, R. W. Hooper, P. D. Hough, K. T. Hu, J. D. Jakeman, K. A. Maupin, J. A. Monschke, E. M. Ridgway, Ahmad Rushdi, L. P. Swiler, J. A. Stephens, D. M. Vigil, and T. M. Wildey. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.4 reference manual. Technical Report SAND2014-5015, Sandia National Laboratories, Albuquerque, NM, Updated May 2016. Available online from `http://dakota.sandia.gov/documentation.html`.

[3] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, J. P. Eddy, M. S. Ebeida, M. S. Eldred, R. W. Hooper, P. D. Hough, K. T. Hu, J. D. Jakeman, K. A. Maupin, J. A. Monschke, E. M. Ridgway, Ahmad Rushdi, L. P. Swiler, J. A. Stephens, D. M. Vigil, and T. M. Wildey. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.4 theory manual. Technical Report SAND2014-4253, Sandia National Laboratories, Albuquerque, NM, Updated May 2016. Available online from `http://dakota.sandia.gov/documentation.html`.

[4] J. S. Arora. *Introduction to Optimum Design*. McGraw-Hill, New York, 1989.

[5] M. N. Avramova. CTF: A thermal hydraulic sub-channel code for LWR transient analyses, userâĂŹs manual. Technical report, Pennsylvania State University, February 2009.

[6] L. S. Bastos and A. O'Hagan. Diagnostics for Gaussian process emulators. *Technometrics*, 51:425–438, 2009.

[7] G. E. P. Box and D. W. Behnken. Some new three level designs for the study of quantitative variables. *Technometrics*, 2:455–475, 1958.

[8] G. E. P. Box and K. B. Wilson. On the experimental attainment of optimum conditions (with discussion). *Journal of the Royal Statistical Society Series B*, 13:1–45, 1951.

[9] Benjamin Collins, Robert Salko, and Shane Stimson. VERA-CS with CIPS modeling capability. Technical Report CASL-I-2015-0285-000, Oak Ridge National Laboratory, August 2015.

[10] N. Cressie. *Statistics of Spatial Data*. John Wiley and Sons, New York, 1991.

[11] K. R. Dalbey, A. A. Giunta, M. D. Richards, E. C. Cyr, L. P. Swiler, S. L. Brown, M. S. Eldred, and B. M. Adams. Surfpack user's manual: Version 1.1. Technical report, Sandia National Laboratories, Albuquerque, NM, 2006. Available online from `http://dakota.sandia.gov/packages/Surfpack`.

[12] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, 1997.

[13] R. Ghanem and J. R. Red-Horse. Propagation of probabilistic uncertainty in complex physical systems using a stochastic finite element technique. *Physica D*, 133:137–144, 1999.

[14] R. G. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer-Verlag, New York, 1991.

[15] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, San Diego, CA, 1981.

[16] H. Haario, M. Laine, A. Mira, and E. Saksman. DRAM: Efficient adaptive MCMC. *Statistics and Computing*, 16:339–354, 2006.

[17] R. T. Haftka and Z. Gurdal. *Elements of Structural Optimization*. Kluwer, Boston, 1992.

[18] W. E. Hart. The Coliny project. Web site, 2007.

[19] A. S. Hedayat, N. J. A. Sloane, and J. Stufken. *Orthogonal Arrays: Theory and Applications*. Springer, New York, 1999.

[20] D. Higdon, J. Gattiker, B. Williams, and M. Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.

[21] E. T. Jaynes and G. Larry Bretthorst. *Probability theory : the logic of science*. Cambridge University Press, Cambridge, UK; New York, NY, 2003.

[22] M.A. Jessee, P.J. Turinsky, and H.S. Abdel-Khalik. Many-group cross-section adjustment techniques for boiling water reactor adaptive simulation. *Nuclear Science and Engineering*, 169(1):40–55, 2011.

[23] C. T. Kelley. *Implicit Filtering*. SIAM, 2011.

[24] B.K. Kendrick. MAMBA theory manual. Private communication, 2015.

[25] M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society*, 63:425–464, 2001.

[26] B.A.A. Khuwaileh. *Scalable Methods for Uncertainty Quantification, Data Assimilation and Target Accuracy Assessment for Multi-Physics Advanced Simulation of Light Water Reactors*. PhD thesis, North Carolina State University, 2015.

[27] B. Kochunas, B. Collins, D. Jabaay, T.J. Downar, and W.R. Martin. Overview of develoment and design of MPACT: Michigan parallel characteristics transport code. Technical report, American Nuclear Society, 555 North Kensington Avenue, La Grange Park, IL 60526, 2013.

[28] S. Koranne. Hierarchical data format 5: HDF5. In *Handbook of Open Source Tools*, chapter 10, pages 191–200. Springer, New York, 2011.

[29] J. L. Loeppky, J. Sacks, and W. J. Welch. Choosing the sample size of a computer experiment: A practial guide. *Technometrics*, 51:366–376, 2009.

[30] W.J. Marshall and B.T. Rearden. Criticality safety validation of Scale 6.1. Technical Report ORNL/TM-2011/450, Oak Ridge National Laboratory, 2011.

[31] G. Matheron. *The theory of regionalized variables and its applications.* Les Cahiers du Centre de morphologie mathématique de Fontainebleau. École national supérieure des mines, 1971.

[32] M. D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

[33] V. A. Mousseau et al. VUQ strategy. Technical Report CASL-U-2014-XXXX-YYY, Oak Ridge National Laboratory, March 2014. In preparation.

[34] R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments.* John Wiley & Sons, Inc., New York, 1995.

[35] J. Nocedal and Wright S. J. *Numerical Optimization.* Springer Series in Operations Research. Springer, New York, 1999.

[36] W.T. Nutt and G.B. Wallis. Evaluation of nuclear safety from the outputs of computer codes in the presence of uncertainties. *Reliability Engineering and System Safety*, 83:57–77, 2004.

[37] W. L. Oberkampf, M. M. Pilch, and T. G. Trucano. Predictive capability maturity model for computational modeling and simulation. Technical Report SAND2007-5948, Sandia National Laboratories, October 2007.

[38] Scott Palmtag. Coupled single assembly solution with VERA (problem 6). Technical Report CASL-U-2013-0150-000, Oak Ridge National Laboratory, 2013.

[39] Robert Salko. CTF list of global variables. Github code repository, 2013.

[40] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models.* John Wiley & Sons, 2004.

[41] SCALE: A comprehensive modeling and simulation suite for nuclear safety analysis and design. Technical Report ORNL/TM-2005/39, Version 6.1, Oak Ridge National Laboratory, 2011. Radiation Safety Information Computational Center at Oak Ridge National Laboratory as CCC-785.

[42] Jeff Secker, Ben Collins, Bob Salko, and Brian Kendrick. L1.CASL.P11.03: Qualify a core-wide PWR CIPS capability that includes an initial corrosion product treatment. Technical Report CASL-I-2015-0318-000, Oak Ridge National Laboratory, September 2015.

[43] R. C. Smith. *Uncertainty Quantification: Theory, Implementation and Applications.* SIAM, Philadelphia, PA, 2014.

[44] A. Solonen, P. Ollinaho, M. Laine, H. Haario, J. Tamminen, and H. Järvinen. Efficient MCMC for climate model parameter estimation: parallel adaptive chains and early rejection. *Bayesian Analysis*, 7(3):715–736, 2012.

[45] R. Sues, M. Aminpour, and Y. Shin. Reliability-based multidisciplinary optimization for aerospace systems. In *Proc. 42rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, number AIAA-2001-1521, Seattle, WA, April 16-19 2001.

[46] L.P. Swiler and V.J. Romero. JANNAF V&V guide: A survey of advanced probabilistic uncertainty propagation and sensitivity analysis methods. Technical Report SAND2013-5456P, Sandia National Laboratories, Albuquerque, NM, 2013.

[47] G. J. Székely and M. L. Rizzo. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143:1249–1272, 2013.

[48] G. L. Székely and M. L. Rizzo. Testing for equal distributions in high dimensions. *InterStat*, 5:1–6, 2004.

[49] J.A. Turner. Virtual environment for reactor applications (VERA): Snapshot 3.1. Technical Report CASLU-2013-0164-000, Oak Ridge National Laboratory, 2013.

[50] G. N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design: With Applications*. McGraw-Hill, New York, 1984.

[51] J.A. Vrugt, C.J.F. Ter Braak, M. P. Clark, J. M. Hyman, and B. A. Robinson. Treatment of input uncertainty in hydrological modeling: Doing hydrology backward with markov chain monte carlo, simulation. *Water Research Resources*, 44(12):W00B09, doi:10.1029/2007WR006720, 2008.

[52] J.A. Vrugt, C.J.F. Ter Braak, C. G. H. Diks, B. A. Robinson, J. M. Hyman, and D. Higdon. Accelerating markov chain monte carlo simulation by differential evolution with self-adaptive randomized subspace sampling. *International Journal of Nonlinear Sciences and Numerical Simulation*, 10(3):273–290, 2009.

[53] S.S Wilks. Determination of sample sizes for setting tolerance limits. *The Annals of Mathematical Statistics*, 12(1):91–96, 1941.

[54] M.L. Williams, D. Wiarda, G. Arbanas, and B.L. Broadhead. SCALE nuclear data covariance library. Technical Report ORNL/TM-2005/39, Version 6, Vol. III, Sect. M19, Oak Ridge National Laboratory, 2009.

[55] Y.-T. Wu, Y. Shin, R. Sues, and M. Cesare. Safety-factor based approach for probability-based design optimization. In *Proc. 42nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, number AIAA-2001-1522, Seattle, WA, April 16–19 2001.