SAND2020-12496
Unlimited Release
November 2020

Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.13 Theory Manual

Keith R. Dalbey, Michael S. Eldred, Gianluca Geraci, John D. Jakeman, Kathryn A. Maupin, Jason A. Monschke, D. Thomas Seidl, Laura P. Swiler, and Anh Tran; with Friedrich Menhorn (Technische Universitat Munchen) and Xiaoshu Zeng (University of Southern California)

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185

Abstract

The Dakota toolkit provides a flexible and extensible interface between simulation codes and iterative analysis methods. Dakota contains algorithms for optimization with gradient and nongradient-based methods; uncertainty quantification with sampling, reliability, and stochastic expansion methods; parameter estimation with nonlinear least squares methods; and sensitivity/variance analysis with design of experiments and parameter study methods. These capabilities may be used on their own or as components within advanced strategies such as surrogate-based optimization, mixed integer nonlinear programming, or optimization under uncertainty. By employing object-oriented design to implement abstractions of the key components required for iterative systems analyses, the Dakota toolkit provides a flexible and extensible problem-solving environment for design and performance analysis of computational models on high performance computers.

This report serves as a theoretical manual for selected algorithms implemented within the Dakota software. It is not intended as a comprehensive theoretical treatment, since a number of existing texts cover general optimization theory, statistical analysis, and other introductory topics. Rather, this manual is intended to summarize a set of Dakota-related research publications in the areas of surrogate-based optimization, uncertainty quantification, and optimization under uncertainty that provide the foundation for many of Dakota's iterative analysis capabilities.

Contents

Chapter 1

Sampling Methods

This chapter introduces several fundamental concepts related to sampling methods. In particular, the statistical properties of the Monte Carlo estimator are discussed (§1.1) and strategies for multilevel and multifidelity sampling are introduced within this context. Hereafter, multilevel refers to the possibility of exploiting distinct discretization levels (i.e. space/time resolution) within a single model form, whereas multifidelity involves the use of more than one model form. In §1.2, we describe the control variate Monte Carlo algorithm that we align with multifidelity sampling, and in §1.3, we describe the multilevel Monte Carlo algorithm that we align with multilevel sampling. In §1.4, we show that these two approaches can be combined to create multilevel-multifidelity sampling approaches.

1.1 Monte Carlo (MC)

Monte Carlo is a popular algorithm for stochastic simulations due to its simplicity, flexibility, and the provably convergent behavior that is independent of the number of input uncertainties. A quantity of interest $Q : \Xi \to \mathbb{R}$, represented as a random variable (RV), can be introduced as a function of a random vector $\boldsymbol{\xi} \in \Xi \subset \mathbb{R}^d$. The goal of any MC simulation is computing statistics for Q, e.g. the expected value $\mathbb{E}[Q]$. The MC estimator $\hat{Q}_N^{MC}$ for $\mathbb{E}[Q]$ is defined as follows

$$\hat{Q}_N^{MC} = \frac{1}{N} \sum_{i=1}^{N} Q^{(i)}, \tag{1.1}$$

where $Q^{(i)} = Q(\boldsymbol{\xi}^{(i)})$ and N is used to indicate the number of realizations of the model.

For large N it is trivial to demonstrate that the MC estimator is unbiased, i.e., its bias is zero and its convergence to the true statistics is $\mathcal{O}(N^{-1/2})$. Moreover, since each sequence of realizations is different, another crucial property of any estimator is its own variance:

$$\mathbb{V}ar\left(\hat{Q}_N^{MC}\right) = \frac{1}{N^2} \sum_{i=1}^{N} \mathbb{V}ar\left(Q\right) = \frac{\mathbb{V}ar\left(Q\right)}{N}. \tag{1.2}$$

Furthermore, it is possible to show, in the limit $N \to \infty$, that the error $\left(\mathbb{E}[Q] - \hat{Q}_N^{MC}\right) \sim \sqrt{\frac{\mathbb{V}ar\left(Q\right)}{N}} \mathcal{N}(0,1)$, where $\mathcal{N}(0,1)$ represents a standard normal RV. As a consequence, it is possible to define a confidence interval for the MC estimator which has an amplitude proportional to the standard deviation of the estimator. Indeed, the

variance of the estimator plays a fundamental role in the quality of the numerical results: the reduction of the variance (for a fixed computational cost) is a very effective way of improving the quality of the MC prediction.

1.2 Control variate Monte Carlo

A closer inspection of Eq. (1.2) indicates that only an increase in the number of simulations N might reduce the overall variance, since $\mathbb{V}ar\left(Q\right)$ is an intrinsic property of the model under analysis. However, more sophisticated techniques have been proposed to accelerate the convergence of a MC simulation. For instance, an incomplete list of these techniques can include stratified sampling, importance sampling, Latin hypercube, deterministic Sobol' sequences and control variates. In particular, the control variate approach, is based on the idea of replacing the RV Q with a different one which has the same expected value, but with a smaller variance. The goal is to reduce the numerator in Eq. (1.2), and hence the value of the estimator variance without requiring a larger number of simulations. In a practical setting, the control variate makes use of an auxiliary functional $G = G(\boldsymbol{\xi})$ for which the expected value $\mathbb{E}\left[G\right]$ is known. Indeed, the alternative estimator can be defined as

$$\hat{Q}_N^{MCCV} = \hat{Q}_N^{MC} - \beta\left(\hat{G}_N^{MC} - \mathbb{E}\left[G\right]\right). \tag{1.3}$$

The MC control variate estimator $\hat{Q}_N^{MCCV}$ is unbiased (irrespective of the value of the parameter $\beta \in \mathbb{R}$), but its variance now has a more complex dependence not only on the $\mathbb{V}ar\left(Q\right)$, but also on $\mathbb{V}ar\left(G\right)$ and the covariance between Q and G since

$$\mathbb{V}ar\left(\hat{Q}_N^{MCCV}\right) = \frac{1}{N}\left(\mathbb{V}ar\left(\hat{Q}_N^{MC}\right) + \beta^2\mathbb{V}ar\left(\hat{G}_N^{MC}\right) - 2\beta\mathrm{Cov}\left(Q,G\right)\right). \tag{1.4}$$

The parameter β can be used to minimize the overall variance leading to

$$\beta = \frac{\mathrm{Cov}\left(Q,G\right)}{\mathbb{V}ar\left(G\right)}, \tag{1.5}$$

for which the estimator variance follows as

$$\mathbb{V}ar\left(\hat{Q}_N^{MCCV}\right) = \mathbb{V}ar\left(\hat{Q}_N^{MC}\right)\left(1-\rho^2\right). \tag{1.6}$$

Therefore, the overall variance of the estimator $\hat{Q}_N^{MCCV}$ is proportional to the variance of the standard MC estimator $\hat{Q}_N^{MC}$ through a factor $1-\rho^2$ where $\rho = \dfrac{\mathrm{Cov}\left(Q,G\right)}{\sqrt{\mathbb{V}ar\left(Q\right)\mathbb{V}ar\left(G\right)}}$ is the Pearson correlation coefficient between Q and G. Since $0 < \rho^2 < 1$, the variance $\mathbb{V}ar\left(\hat{Q}_N^{MCCV}\right)$ is always less than the corresponding $\mathbb{V}ar\left(\hat{Q}_N^{MC}\right)$. The control variate technique can be seen as a very general approach to accelerate a MC simulation. The main step is to define a convenient control variate function which is cheap to evaluate and well correlated to the target function. For instance, function evaluations obtained through a different (coarse) resolution may be employed or even coming from a more crude physical/engineering approximation of the problem. A viable way of building a well correlated control variate is to rely on a low-fidelity model (i.e. a crude approximation of the model of interest) to estimate the control variate using estimated control means (see [104]). This technique has been introduced in the context of optimization under uncertainty in [98].

1.3 Multilevel Monte Carlo

In general engineering applications, the quantity of interest Q is obtained as the result of the numerical solution of a partial partial differential equation (possibly a system of them). Therefore, the dependence on the physical

$\mathbf{x} \in \Omega \subset \mathbb{R}^n$ and/or temporal $t \in T \subset \mathbb{R}^+$ coordinates should be included, hence $Q = Q(\mathbf{x}, \boldsymbol{\xi}, t)$. A finite spatial/temporal resolution is always employed to numerically solve a PDE, implying the presence of a discretization error in addition to the stochastic error. The term discretization is applied generically with reference to either the spatial tessellation, the temporal resolution, or both (commonly, they are linked). For a generic tessellation with M degrees-of-freedom (DOFs), the PDE solution of Q is referred to as Q_M. Since $Q_M \to Q$ for $M \to \infty$, then $\mathbb{E}[Q_M] \to \mathbb{E}[Q]$ for $M \to \infty$ with a prescribed order of convergence. A MC estimator in presence of a finite spatial resolution and finite sampling is

$$\hat{Q}^{MC}_{M,N} = \frac{1}{N} \sum_{i=1}^{N} Q^{(i)}_M \tag{1.7}$$

for which the mean square error (MSE) is

$$\mathbb{E}\left[\left(\hat{Q}^{MC}_{M,N} - \mathbb{E}[Q] \right)^2 \right] = N^{-1} \mathbb{V}ar\left[Q_M \right] + \left(\mathbb{E}[Q_M - Q] \right)^2, \tag{1.8}$$

where the first term represents the variance of the estimator, and the second term $\left(\mathbb{E}[Q_M - Q] \right)^2$ reflects the bias introduced by the (finite) spatial discretization. The two contributions appear to be independent of each other; accurate MC estimates can only be obtained by drawing the required N number of simulations of $Q_M(\boldsymbol{\xi})$ at a sufficiently fine resolution M. Since the numerical cost of a PDE is related to the number of DOFs of the tessellation, the total cost of a MC simulation for a PDE can easily become intractable for complex multi-physics applications that are computationally intensive.

1.3.1 Multilevel Monte Carlo for the estimator mean

The multilevel Monte Carlo (MLMC) algorithm has been introduced, starting from the control variate idea, for situation in which additional discretization levels can be defined. The basic idea, borrowed from the multigrid approach, is to replace the evaluation of the statistics of Q_M with a sequence of evaluations at coarser levels. If it is possible to define a sequence of discretization levels $\{M_\ell : \ell = 0, \dots, L\}$ with $M_0 < M_1 < \cdots < M_L \stackrel{\text{def}}{=} M$, the expected value $\mathbb{E}[Q_M]$ can be decomposed, exploiting the linearity of the expected value operator as

$$\mathbb{E}[Q_M] = \mathbb{E}[Q_{M_0}] + \sum_{\ell=1}^{L} \mathbb{E}\left[Q_{M_\ell} - Q_{M_{\ell-1}} \right]. \tag{1.9}$$

If the difference function Y_ℓ is defined according to

$$Y_\ell = \begin{cases} Q_{M_0} & \text{if} \quad \ell = 0 \\ Q_{M_\ell} - Q_{M_{\ell-1}} & \text{if} \quad 0 < \ell \leq L, \end{cases} \tag{1.10}$$

the expected value $\mathbb{E}[Q_M] = \sum_{\ell=0}^{L} \mathbb{E}[Y_\ell]$. A multilevel MC estimator is obtained when a MC estimator is adopted independently for the evaluation of the expected value of Y_ℓ on each level. The resulting multilevel estimator $\hat{Q}^{\text{ML}}_M$ is

$$\hat{Q}^{\text{ML}}_M = \sum_{\ell=0}^{L} \hat{Y}^{\text{MC}}_{\ell,N_\ell} = \sum_{\ell=0}^{L} \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} Y^{(i)}_\ell. \tag{1.11}$$

Since the multilevel estimator is unbiased, the advantage of using this formulation is in its reduced estimator variance $\sum_{\ell=0}^{L} N_\ell^{-1} \mathbb{V}ar[Y_\ell]$: since $Q_M \to Q$, the difference function $Y_\ell \to 0$ as the level ℓ increases. Indeed, the corresponding number of samples N_ℓ required to resolve the variance associated with the ℓth level is expected to decrease with ℓ.

The MLMC algorithm can be interpreted as a strategy to optimally allocate resources. If the total cost of the MLMC algorithm is written as

$$\mathcal{C}(\hat{Q}_M^{ML}) = \sum_{\ell=0}^{L} N_\ell \, \mathcal{C}_\ell, \tag{1.12}$$

with $\mathcal{C}_\ell$ being the cost of the evaluation of Y_ℓ (involving either one or two discretization evaluations), then the following constrained minimization problem can be formulated where an equality constraint enforces a stochastic error (from MLMC estimator variance) equal to the residual bias error ($\varepsilon^2/2$)

$$f(N_\ell, \lambda) = \sum_{\ell=0}^{L} N_\ell \, \mathcal{C}_\ell + \lambda \left(\sum_{\ell=0}^{L} N_\ell^{-1} \mathbb{V}ar\left(Y_\ell\right) - \varepsilon^2/2 \right). \tag{1.13}$$

using a Lagrange multiplier λ. This equality constraint reflects a balance between the two contributions to MSE, reflecting the goal to not over-resolve one or the other. The result of the minimization is

$$N_\ell = \frac{2}{\varepsilon^2} \left[\sum_{k=0}^{L} \left(\mathbb{V}ar\left(Y_k\right) \mathcal{C}_k \right)^{1/2} \right] \sqrt{\frac{\mathbb{V}ar\left(Y_\ell\right)}{\mathcal{C}_\ell}}, \tag{1.14}$$

defining an optimal sample allocation per discretization level.

1.3.2 Extension to the estimator variance

Despite the original introduction of the MLMC approach for the computation of the mean estimator in [63, 64], it is possible to estimate higher-order moments with a MLMC sampling strategy, as for instance the variance.

A single level unbiased estimator for the variance of a generic QoI at the highest level M_L of the hierarchy can be written as

$$\mathbb{V}ar\left[Q_{M_L}\right] \approx \frac{1}{N_{M_L} - 1} \sum_{i=1}^{N_{M_L}} \left(Q_{M_L}^{(i)} - \mathbb{E}\left[Q_L\right] \right)^2. \tag{1.15}$$

The multilevel version of Eq. (1.15) can be obtained via a telescopic expansion in term of difference of estimators over subsequent levels. To simplify the notation and for simplicity of exposure from now on we only indicate the level, *i.e.* $M_\ell = \ell$.

The expansion is obtained by re-writing Eq. (1.15) as

$$\begin{aligned} \mathbb{V}ar\left[Q_L\right] &\approx \frac{1}{N_L - 1} \sum_{i=1}^{N_L} \left(Q_L^{(i)} - \mathbb{E}\left[Q_L\right] \right)^2 \\ &\approx \sum_{\ell=0}^{L} \frac{1}{N_\ell - 1} \left(\left(Q_\ell^{(i)} - \mathbb{E}\left[Q_\ell\right] \right)^2 - \left(Q_{\ell-1}^{(i)} - \mathbb{E}\left[Q_{\ell-1}\right] \right)^2 \right). \end{aligned} \tag{1.16}$$

It is important here to note that since the estimators at the levels ℓ and $\ell - 1$ are computed with the same number of samples both estimators use the factor $1/(N_\ell - 1)$ to obtain their unbiased version. Moreover, each estimator is indeed written with respect to its own mean value, *i.e.* the mean value on its level, either ℓ or $\ell - 1$. This last requirement leads to the computation of a local expected value estimator with respect to the same samples employed for the difference estimator. If we now denote with $\hat{Q}_{\ell,2}$ the sampling estimator for the second order moment of the QoI Q_ℓ we can write

$$\mathbb{V}ar\left[Q_L\right] \approx \hat{Q}_{L,2}^{\mathrm{ML}} = \sum_{\ell=0}^{L} \hat{Q}_{\ell,2} - \hat{Q}_{\ell-1,2}, \tag{1.17}$$

where

$$\hat{Q}_{\ell,2} = \frac{1}{N_\ell - 1} \sum_{i=1}^{N_\ell} \left(Q_\ell^{(i)} - \hat{Q}_\ell \right)^2 \quad \text{and} \quad \hat{Q}_{\ell-1,2} = \frac{1}{N_\ell - 1} \sum_{i=1}^{N_\ell} \left(Q_{\ell-1}^{(i)} - \hat{Q}_{\ell-1} \right)^2 . \tag{1.18}$$

Note that $\hat{Q}_{\ell,2}$ and $\hat{Q}_{\ell-1,2}$ are explicitly sharing the same samples N_ℓ.

For this estimator we are interested in minimizing its cost while also prescribing its variance as done for the expected value. This is accomplished by evaluating the variance of the multilevel variance estimator $\hat{Q}_{L,2}^{ML}$

$$\mathbb{V}ar\left[\hat{Q}_{L,2}^{\mathrm{ML}}\right] = \sum_{\ell=0}^{L} \mathbb{V}ar\left[\hat{Q}_{\ell,2} - \hat{Q}_{\ell-1,2}\right] = \sum_{\ell=0}^{L} \mathbb{V}ar\left[\hat{Q}_{\ell,2}\right] + \mathbb{V}ar\left[\hat{Q}_{\ell-1,2}\right] - 2\mathbb{C}ov\left(\hat{Q}_{\ell,2}, \hat{Q}_{\ell-1,2}\right), \tag{1.19}$$

where the covariance term is a result of the dependence described in (1.18).

The previous expression can be evaluated once the variance for the sample estimator of the second order order moment $\mathbb{V}ar\left[\hat{Q}_{\ell,2}\right]$ and the covariance term $\mathbb{C}ov\left(\hat{Q}_{\ell,2}, \hat{Q}_{\ell-1,2}\right)$ are known. These terms can be evaluated as:

$$\mathbb{V}ar\left[\hat{Q}_{\ell,2}\right] \approx \frac{1}{N_\ell}\left(\hat{Q}_{\ell,4} - \frac{N_\ell - 3}{N_\ell - 1}\left(\hat{Q}_{\ell,2}\right)^2\right), \tag{1.20}$$

where $\hat{Q}_{\ell,4}$ denotes the sampling estimator for the fourth order central moment.

The expression for the covariance term is more involved and can be written as

$$\begin{aligned}\mathbb{C}ov\left(\hat{Q}_{\ell,2}, \hat{Q}_{\ell-1,2}\right) \approx & \frac{1}{N_\ell}\mathbb{E}\left[\hat{Q}_{\ell,2}, \hat{Q}_{\ell-1,2}\right] \\ & + \frac{1}{N_\ell N_{\ell-1}}\left(\mathbb{E}\left[Q_\ell Q_{\ell-1}\right]^2 - 2\mathbb{E}\left[Q_\ell Q_{\ell-1}\right]\mathbb{E}\left[Q_\ell\right]\mathbb{E}\left[Q_{\ell-1}\right] + \left(\mathbb{E}\left[Q_\ell\right]\mathbb{E}\left[Q_{\ell-1}\right]\right)^2\right).\end{aligned} \tag{1.21}$$

The first term of the previous expression is evaluated by estimating and combining several sampling moments as

$$\begin{aligned}\mathbb{E}\left[\hat{Q}_{\ell,2}, \hat{Q}_{\ell-1,2}\right] = & \frac{1}{N_\ell}\left(\mathbb{E}\left[Q_\ell^2 Q_{\ell-1}^2\right]\right) - \mathbb{E}\left[Q_\ell^2\right]\mathbb{E}\left[Q_{\ell-1}^2\right] - 2\mathbb{E}\left[Q_{\ell-1}\right]\mathbb{E}\left[Q_\ell^2 Q_{\ell-1}\right] \\ & + 2\mathbb{E}\left[Q_{\ell-1}^2\right]\mathbb{E}\left[Q_\ell^2\right] - 2\mathbb{E}\left[Q_\ell\right]\mathbb{E}\left[Q_\ell Q_{\ell-1}^2\right] + 2\mathbb{E}\left[Q_\ell\right]^2\mathbb{E}\left[Q_{\ell-1}^2\right] \\ & + 4\mathbb{E}\left[Q_\ell\right]\mathbb{E}\left[Q_{\ell-1}\right]\mathbb{E}\left[Q_\ell Q_{\ell-1}\right] - 4\mathbb{E}\left[Q_\ell\right]^2\mathbb{E}\left[Q_{\ell-1}\right]^2 .\end{aligned} \tag{1.22}$$

It is important to note here that the previous expression can be computed only if several sampling estimators for product of the QoIs at levels ℓ and $\ell - 1$ are available. These quantities are not required in the standard MLMC implementation for the mean and therefore for the estimation of the variance more data need to be stored to assemble the quantities on each level.

An optimization problem, similar to the one formulated for the mean in the previous section, can be written in the case of variance

$$\min_{N_\ell} \sum_{\ell=0}^{L} \mathcal{C}_\ell N_\ell \quad \text{s.t.} \quad \mathbb{V}ar\left[\hat{Q}_{L,2}^{\mathrm{ML}}\right] = \varepsilon^2/2. \tag{1.23}$$

This optimization problem can be solved in two different ways, namely an analytical approximation and by solving a non-linear optimization problem. The analytical approximation follows the approach described in [109] and introduces a helper variable

$$\hat{V}_{2,\ell} := \mathbb{V}ar\left[\hat{Q}_{\ell,2}\right] \cdot N_\ell. \tag{1.24}$$

Next, the following constrained minimization problem is formulated

$$f(N_\ell, \lambda) = \sum_{\ell=0}^{L} N_\ell \mathcal{C}_\ell + \lambda \left(\sum_{\ell=0}^{L} N_\ell^{-1} \hat{V}_{2,\ell} - \varepsilon^2/2 \right), \tag{1.25}$$

and a closed form solution is obtained

$$N_\ell = \frac{2}{\varepsilon^2} \left[\sum_{k=0}^{L} \left(\hat{V}_{2,k} \mathcal{C}_k \right)^{1/2} \right] \sqrt{\frac{\hat{V}_{2,\ell}}{\mathcal{C}_\ell}}, \tag{1.26}$$

similarly as for the expected value in (1.13).

The second approach uses numerical optimization directly on the non-linear optimization problem (1.23) to find an optimal sample allocation. Dakota uses OPTPP as the default optimizer and switches to NPSOL if it is available.

Both approaches for finding the optimal sample allocation when allocating for the variance are currently implemented in Dakota. The analytical solution is employed by default while the optimization is enabled using a keyword. We refer to the reference manual for a discussion of the keywords to select these different options.

1.4 A multilevel-multifidelity approach

The MLMC approach described in §1.3 can be considered to be a recursive control variate technique in that it seeks to reduce the variance of the target function in order to limit the sampling at high resolution. In addition, the difference function Y_ℓ for each level can itself be the target of an additional control variate (refer to §1.2). A practical scenario is when not only different resolution levels are available (multilevel part), but also a cheaper computational model can be used (multifidelity part). The combined approach is a multilevel-multifidelity algorithm, and in particular, a multilevel-control variate Monte Carlo sampling approach.

1.4.1 Y_l correlations

If the target QoI can be generated from both a high-fidelity (HF) model and a cheaper, possibly biased low-fidelity (LF) model, it is possible to write the following estimator

$$\mathbb{E}\left[Q_M^{\mathrm{HF}}\right] = \sum_{l=0}^{L_{\mathrm{HF}}} \mathbb{E}\left[Y_\ell^{\mathrm{HF}}\right] \approx \sum_{l=0}^{L_{\mathrm{HF}}} \hat{Y}_\ell^{\mathrm{HF}} = \sum_{l=0}^{L_{\mathrm{HF}}} Y_\ell^{\mathrm{HF},\star}, \tag{1.27}$$

where

$$Y_\ell^{\mathrm{HF},\star} = Y_\ell^{\mathrm{HF}} + \alpha_\ell \left(\hat{Y}_\ell^{\mathrm{LF}} - \mathbb{E}\left[Y_\ell^{\mathrm{LF}}\right] \right). \tag{1.28}$$

The estimator $Y_\ell^{\mathrm{HF},\star}$ is unbiased with respect to $\hat{Y}_\ell^{\mathrm{HF}}$, hence with respect to the true value $\mathbb{E}\left[Y_\ell^{\mathrm{HF}}\right]$. The control variate is obtained by means of the LF model realizations for which the expected value can be computed in two different ways: $\hat{Y}_\ell^{\mathrm{LF}}$ and $\mathbb{E}\left[Y_\ell^{\mathrm{LF}}\right]$. A MC estimator is employed for each term but the estimation of $\mathbb{E}\left[Y_\ell^{\mathrm{LF}}\right]$ is more resolved than $\hat{Y}_\ell^{\mathrm{LF}}$. For $\hat{Y}_\ell^{\mathrm{LF}}$, we choose the number of LF realizations to be equal to the number of HF realizations, N_ℓ^{HF}. For the more resolved $\mathbb{E}\left[Y_\ell^{\mathrm{LF}}\right]$, we augment with an additional and independent set of realizations $\Delta_\ell^{\mathrm{LF}}$, hence $N_\ell^{\mathrm{LF}} = N_\ell^{\mathrm{HF}} + \Delta_\ell^{\mathrm{LF}}$. The set $\Delta_\ell^{\mathrm{LF}}$ is written, for convenience, as proportional to N_ℓ^{HF} by means of a parameter $r_\ell \in \mathbb{R}_0^+$

$$N_\ell^{\mathrm{LF}} = N_\ell^{\mathrm{HF}} + \Delta_\ell^{\mathrm{LF}} = N_\ell^{\mathrm{HF}} + r_\ell N_\ell^{\mathrm{HF}} = N_\ell^{\mathrm{HF}}(1 + r_\ell). \tag{1.29}$$

The set of samples $\Delta_\ell^{\mathrm{LF}}$ is independent of N_ℓ^{HF}, therefore the variance of the estimator can be written as (for further details see [60])

$$\begin{aligned}\mathbb{V}ar\left(\hat{Q}_M^{MLMF}\right) = \sum_{l=0}^{L_{\mathrm{HF}}} \Bigg(& \frac{1}{N_\ell^{\mathrm{HF}}}\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right) + \frac{\alpha_\ell^2 r_\ell}{(1+r_\ell)N_\ell^{\mathrm{HF}}}\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right) \\ & + 2\frac{\alpha_\ell r_\ell^2}{(1+r_\ell)N_\ell^{\mathrm{HF}}}\rho_\ell\sqrt{\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right)\mathbb{V}ar\left(Y_\ell^{\mathrm{LF}}\right)}\Bigg),\end{aligned} \tag{1.30}$$

The Pearson's correlation coefficient between the HF and LF models is indicated by ρ_ℓ in the previous equations. Assuming the vector r_ℓ as a parameter, the variance is minimized per level, mimicking the standard control variate approach, and thus obtaining the optimal coefficient as $\alpha_\ell = -\rho_\ell\sqrt{\frac{\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right)}{\mathbb{V}ar\left(Y_\ell^{\mathrm{LF}}\right)}}$. By making use of the optimal coefficient α_ℓ, it is possible to show that the variance $\mathbb{V}ar\left(Y_\ell^{\mathrm{HF},\star}\right)$ is proportional to the variance $\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right)$ through a factor $\Lambda_\ell(r_\ell)$, which is an explicit function of the ratio r_ℓ:

$$\begin{aligned}\mathbb{V}ar\left(\hat{Q}_M^{MLMF}\right) &= \sum_{l=0}^{L_{\mathrm{HF}}} \frac{1}{N_\ell^{\mathrm{HF}}}\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right)\Lambda_\ell(r_\ell) \quad \text{where} \\ \Lambda_\ell(r_\ell) &= \left(1 - \frac{r_\ell}{1+r_\ell}\rho_\ell^2\right).\end{aligned} \tag{1.31}$$

Note that $\Lambda_\ell(r_\ell)$ represents a penalty with respect to the classical control variate approach presented in §1.2, which stems from the need to evaluate the unknown function $\mathbb{E}\left[Y_\ell^{\mathrm{LF}}\right]$. However, the ratio $r_\ell/(r_\ell+1)$ is dependent on the additional number of LF evaluations $\Delta_\ell^{\mathrm{LF}}$, hence it is fair to assume that it can be made very close to unity by choosing an affordably large r_ℓ, i.e., $\Delta_\ell^{\mathrm{LF}} >> N_\ell^{\mathrm{HF}}$.

The optimal sample allocation is determined taking into account the relative cost between the HF and LF models and their correlation (per level). In particular the optimization problem introduced in Eq. (1.13) is replaced by

$$\underset{N_\ell^{\mathrm{HF}},r_\ell}{\operatorname{argmin}}(\mathcal{L}), \quad \text{where} \quad \mathcal{L} = \sum_{\ell=0}^{L_{\mathrm{HF}}} N_\ell^{\mathrm{HF}}\mathcal{C}_\ell^{\mathrm{eq}} + \lambda\left(\sum_{\ell=0}^{L_{\mathrm{HF}}}\frac{1}{N_\ell^{\mathrm{HF}}}\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right)\Lambda_\ell(r_\ell) - \varepsilon^2/2\right),$$

where the optimal allocation is obtained as well as the optimal ratio r_ℓ. The cost per level includes now the sum of the HF and LF realization cost, therefore it can be expressed as $\mathcal{C}_\ell^{\mathrm{eq}} = \mathcal{C}_\ell^{\mathrm{HF}} + \mathcal{C}_\ell^{\mathrm{LF}}(1+r_\ell)$.

If the cost ratio between the HF and LF model is $w_\ell = \mathcal{C}_\ell^{\mathrm{HF}}/\mathcal{C}_\ell^{\mathrm{LF}}$ then the optimal ratio is

$$r_\ell^\star = -1 + \sqrt{\frac{\rho_\ell^2}{1-\rho_\ell^2}w_\ell}, \tag{1.32}$$

and the optimal allocation is

$$N_\ell^{\mathrm{HF},\star} = \frac{2}{\varepsilon^2}\left[\sum_{k=0}^{L_{\mathrm{HF}}}\left(\frac{\mathbb{V}ar\left(Y_k^{\mathrm{HF}}\right)\mathcal{C}_k^{\mathrm{HF}}}{1-\rho_\ell^2}\right)^{1/2}\Lambda_k(r_k^\star)\right]\sqrt{(1-\rho_\ell^2)\frac{\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right)}{\mathcal{C}_\ell^{\mathrm{HF}}}}. \tag{1.33}$$

It is clear that the efficiency of the algorithm is related not only to the efficiency of the LF model, i.e. how fast a simulation runs with respect to the HF model, but also to the correlation between the LF and HF model.

1.4.2 Q_l correlations

A potential refinement of the previous approach consists in exploiting the QoI on each pair of levels, ℓ and $\ell-1$, to build a more correlated LF function. For instance, it is possible to use

$$\mathring{Y}_\ell^{\mathrm{LF}} = \gamma_\ell Q_\ell^{\mathrm{LF}} - Q_{\ell-1}^{\mathrm{LF}} \tag{1.34}$$

and maximize the correlation between Y_ℓ^{HF} and $\mathring{Y}_\ell^{\mathrm{LF}}$ through the coefficient γ_ℓ.

Formally the two formulations are completely equivalent if Y_ℓ^{LF} is replaced with $\mathring{Y}_\ell^{\mathrm{LF}}$ in Equation (1.27) and they can be linked through the two ratios

$$\begin{aligned}\theta_\ell &= \frac{\mathrm{Cov}\left(Y_\ell^{\mathrm{HF}}, \mathring{Y}_\ell^{\mathrm{LF}}\right)}{\mathrm{Cov}\left(Y_\ell^{\mathrm{HF}}, Y_\ell^{\mathrm{LF}}\right)} \\ \tau_\ell &= \frac{\mathbb{V}ar\left(\mathring{Y}_\ell^{\mathrm{LF}}\right)}{\mathbb{V}ar\left(Y_\ell^{\mathrm{LF}}\right)},\end{aligned} \tag{1.35}$$

obtaining the following variance for the estimator

$$\mathbb{V}ar\left(\hat{Q}_M^{MLMF}\right) = \frac{1}{N_\ell^{\mathrm{HF}}}\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right)\left(1 - \frac{r_\ell}{1+r_\ell}\rho_\ell^2\frac{\theta_\ell^2}{\tau_\ell}\right). \tag{1.36}$$

Therefore, a way to increase the variance reduction is to maximize the ratio $\frac{\theta_\ell^2}{\tau_\ell}$ with respect to the parameter γ_ℓ. It is possible to solve analytically this maximization problem obtaining

$$\gamma_\ell^\star = \frac{\mathrm{Cov}\left(Y_\ell^{\mathrm{HF}}, Q_{\ell-1}^{\mathrm{LF}}\right)\mathrm{Cov}\left(Q_\ell^{\mathrm{LF}}, Q_{\ell-1}^{\mathrm{LF}}\right) - \mathbb{V}ar\left(Q_{\ell-1}^{\mathrm{LF}}\right)\mathrm{Cov}\left(Y_\ell^{\mathrm{HF}}, Q_\ell^{\mathrm{LF}}\right)}{\mathbb{V}ar\left(Q_\ell^{\mathrm{LF}}\right)\mathrm{Cov}\left(Y_\ell^{\mathrm{HF}}, Q_{\ell-1}^{\mathrm{LF}}\right) - \mathrm{Cov}\left(Y_\ell^{\mathrm{HF}}, Q_\ell^{\mathrm{LF}}\right)\mathrm{Cov}\left(Q_\ell^{\mathrm{LF}}, Q_{\ell-1}^{\mathrm{LF}}\right)}. \tag{1.37}$$

The resulting optimal allocation of samples across levels and model forms is given by

$$\begin{aligned}r_\ell^\star &= -1 + \sqrt{\frac{\rho_l^2\frac{\theta_\ell^2}{\tau_\ell}}{1-\rho_\ell^2\frac{\theta_\ell^2}{\tau_\ell}}w_\ell}, \quad \text{where} \quad w_\ell = \mathcal{C}_\ell^{\mathrm{HF}}/\mathcal{C}_\ell^{\mathrm{LF}} \\ \Lambda_\ell &= 1 - \rho_\ell^2\frac{\theta_\ell^2}{\tau_\ell}\frac{r_\ell^\star}{1+r_\ell^\star} \\ N_\ell^{\mathrm{HF},\star} &= \frac{2}{\varepsilon^2}\left[\sum_{k=0}^{L_{\mathrm{HF}}}\left(\frac{\mathbb{V}ar\left(Y_k^{\mathrm{HF}}\right)\mathcal{C}_k^{\mathrm{HF}}}{1-\rho_\ell^2\frac{\theta_\ell^2}{\tau_\ell}}\right)^{1/2}\Lambda_k(r_k^\star)\right]\sqrt{\left(1-\rho_\ell^2\frac{\theta_\ell^2}{\tau_\ell}\right)\frac{\mathbb{V}ar\left(Y_\ell^{\mathrm{HF}}\right)}{\mathcal{C}_\ell^{\mathrm{HF}}}}\end{aligned}$$

Chapter 2

Reliability Methods

2.1 Local Reliability Methods

Local reliability methods include the Mean Value method and the family of most probable point (MPP) search methods. Each of these methods is gradient-based, employing local approximations and/or local optimization methods.

2.1.1 Mean Value

The Mean Value method (MV, also known as MVFOSM in [70]) is the simplest, least-expensive reliability method because it estimates the response means, response standard deviations, and all CDF/CCDF response-probability-reliability levels from a single evaluation of response functions and their gradients at the uncertain variable means. This approximation can have acceptable accuracy when the response functions are nearly linear and their distributions are approximately Gaussian, but can have poor accuracy in other situations.

The expressions for approximate response mean μ_g and approximate response variance σ_g^2 are

$$\mu_g = g(\mu_{\mathbf{x}}) \tag{2.1}$$

$$\sigma_g^2 = \sum_i \sum_j Cov(i,j) \frac{dg}{dx_i}(\mu_{\mathbf{x}}) \frac{dg}{dx_j}(\mu_{\mathbf{x}}) \tag{2.2}$$

where $\mathbf{x}$ are the uncertain values in the space of the original uncertain variables ("x-space"), $g(\mathbf{x})$ is the limit state function (the response function for which probability-response level pairs are needed), and the use of a linear Taylor series approximation is evident. These two moments are then used for mappings from response target to approximate reliability level ($\bar{z} \rightarrow \beta$) and from reliability target to approximate response level ($\bar{\beta} \rightarrow z$) using

$$\bar{z} \rightarrow \beta: \qquad \beta_{\text{CDF}} = \frac{\mu_g - \bar{z}}{\sigma_g}, \quad \beta_{\text{CCDF}} = \frac{\bar{z} - \mu_g}{\sigma_g} \tag{2.3}$$

$$\bar{\beta} \rightarrow z: \qquad z = \mu_g - \sigma_g \bar{\beta}_{\text{CDF}}, \quad z = \mu_g + \sigma_g \bar{\beta}_{\text{CCDF}} \tag{2.4}$$

respectively, where β_{CDF} and β_{CCDF} are the reliability indices corresponding to the cumulative and complementary cumulative distribution functions (CDF and CCDF), respectively.

With the introduction of second-order limit state information, MVSOSM calculates a second-order mean as

$$\mu_g = g(\mu_{\mathbf{x}}) + \frac{1}{2}\sum_i \sum_j Cov(i,j)\frac{d^2 g}{dx_i dx_j}(\mu_{\mathbf{x}}) \tag{2.5}$$

This is commonly combined with a first-order variance (Equation 2.2), since second-order variance involves higher order distribution moments (skewness, kurtosis) [70] which are often unavailable.

The first-order CDF probability $p(g \leq z)$, first-order CCDF probability $p(g > z)$, β_{CDF}, and β_{CCDF} are related to one another through

$$\begin{aligned} p(g \leq z) &= \Phi(-\beta_{\text{CDF}}) && (2.6)\\ p(g > z) &= \Phi(-\beta_{\text{CCDF}}) && (2.7)\\ \beta_{\text{CDF}} &= -\Phi^{-1}(p(g \leq z)) && (2.8)\\ \beta_{\text{CCDF}} &= -\Phi^{-1}(p(g > z)) && (2.9)\\ \beta_{\text{CDF}} &= -\beta_{\text{CCDF}} && (2.10)\\ p(g \leq z) &= 1 - p(g > z) && (2.11) \end{aligned}$$

where $\Phi()$ is the standard normal cumulative distribution function, indicating the introduction of a Gaussian assumption on the output distributions. A common convention in the literature is to define g in such a way that the CDF probability for a response level z of zero (i.e., $p(g \leq 0)$) is the response metric of interest. Dakota is not restricted to this convention and is designed to support CDF or CCDF mappings for general response, probability, and reliability level sequences.

With the Mean Value method, it is possible to obtain importance factors indicating the relative contribution of the input variables to the output variance. The importance factors can be viewed as an extension of linear sensitivity analysis combining deterministic gradient information with input uncertainty information, *i.e.* input variable standard deviations. The accuracy of the importance factors is contingent of the validity of the linear Taylor series approximation used to approximate the response quantities of interest. The importance factors are determined as follows for each of n random variables:

$$\text{ImportFactor}_i = \left[\frac{\sigma_{x_i}}{\sigma_g}\frac{dg}{dx_i}(\mu_{\mathbf{x}})\right]^2, \quad i = 1, \ldots, n \tag{2.12}$$

where it is evident that these importance factors correspond to the diagonal terms in Eq. 2.2 normalized by the total response variance. In the case where the input variables are correlated resulting in off-diagonal terms for the input covariance, we can also compute a two-way importance factor as

$$\text{ImportFactor}_{ij} = 2\frac{\sigma^2_{x_{ij}}}{\sigma^2_g}\frac{dg}{dx_i}(\mu_{\mathbf{x}})\frac{dg}{dx_j}(\mu_{\mathbf{x}}), \quad i = 1, \ldots, n; \quad j = 1, \ldots, i-1 \tag{2.13}$$

These two-way factors differ from the Sobol' interaction terms that are computed in variance-based decomposition (refer to Section 3.10) due to the non-orthogonality of the Taylor series basis. Due to this non-orthogonality, two-way importance factors may be negative, and due to normalization by the total response variance, the set of importance factors will always sum to one.

2.1.2 MPP Search Methods

All other local reliability methods solve an equality-constrained nonlinear optimization problem to compute a most probable point (MPP) and then integrate about this point to compute probabilities. The MPP search is

performed in uncorrelated standard normal space ("u-space") since it simplifies the probability integration: the distance of the MPP from the origin has the meaning of the number of input standard deviations separating the mean response from a particular response threshold. The transformation from correlated non-normal distributions (x-space) to uncorrelated standard normal distributions (u-space) is denoted as $\mathbf{u} = T(\mathbf{x})$ with the reverse transformation denoted as $\mathbf{x} = T^{-1}(\mathbf{u})$. These transformations are nonlinear in general, and possible approaches include the Rosenblatt [117], Nataf [36], and Box-Cox [14] transformations. The nonlinear transformations may also be linearized, and common approaches for this include the Rackwitz-Fiessler [111] two-parameter equivalent normal and the Chen-Lind [23] and Wu-Wirsching [139] three-parameter equivalent normals. Dakota employs the Nataf nonlinear transformation which is suitable for the common case when marginal distributions and a correlation matrix are provided, but full joint distributions are not known[1]. This transformation occurs in the following two steps. To transform between the original correlated x-space variables and correlated standard normals ("z-space"), a CDF matching condition is applied for each of the marginal distributions:

$$\Phi(z_i) = F(x_i) \tag{2.14}$$

where $F()$ is the cumulative distribution function of the original probability distribution. Then, to transform between correlated z-space variables and uncorrelated u-space variables, the Cholesky factor $\mathbf{L}$ of a modified correlation matrix is used:

$$\mathbf{z} = \mathbf{L}\mathbf{u} \tag{2.15}$$

where the original correlation matrix for non-normals in x-space has been modified to represent the corresponding "warped" correlation in z-space [36].

The forward reliability analysis algorithm of computing CDF/CCDF probability/reliability levels for specified response levels is called the reliability index approach (RIA), and the inverse reliability analysis algorithm of computing response levels for specified CDF/CCDF probability/reliability levels is called the performance measure approach (PMA) [130]. The differences between the RIA and PMA formulations appear in the objective function and equality constraint formulations used in the MPP searches. For RIA, the MPP search for achieving the specified response level $\bar{z}$ is formulated as computing the minimum distance in u-space from the origin to the $\bar{z}$ contour of the limit state response function:

$$\begin{aligned} \text{minimize} \quad & \mathbf{u}^T\mathbf{u} \\ \text{subject to} \quad & G(\mathbf{u}) = \bar{z} \end{aligned} \tag{2.16}$$

where $\mathbf{u}$ is a vector centered at the origin in u-space and $g(\mathbf{x}) \equiv G(\mathbf{u})$ by definition. For PMA, the MPP search for achieving the specified reliability level $\bar{\beta}$ or first-order probability level $\bar{p}$ is formulated as computing the minimum/maximum response function value corresponding to a prescribed distance from the origin in u-space:

$$\begin{aligned} \text{minimize} \quad & \pm G(\mathbf{u}) \\ \text{subject to} \quad & \mathbf{u}^T\mathbf{u} = \bar{\beta}^2 \end{aligned} \tag{2.17}$$

where $\bar{\beta}$ is computed from $\bar{p}$ using Eq. 2.8 or 2.9 in the latter case of a prescribed first-order probability level. For a specified generalized reliability level $\bar{\beta}^*$ or second-order probability level $\bar{p}$, the equality constraint is reformulated in terms of the generalized reliability index:

$$\begin{aligned} \text{minimize} \quad & \pm G(\mathbf{u}) \\ \text{subject to} \quad & \beta^*(\mathbf{u}) = \bar{\beta}^* \end{aligned} \tag{2.18}$$

where $\bar{\beta}^*$ is computed from $\bar{p}$ using Eq. 10.7 (or its CCDF complement) in the latter case of a prescribed second-order probability level.

[1] If joint distributions are known, then the Rosenblatt transformation is preferred.

In the RIA case, the optimal MPP solution $\mathbf{u}^*$ defines the reliability index from $\beta = \pm\|\mathbf{u}^*\|_2$, which in turn defines the CDF/CCDF probabilities (using Equations 2.6-2.7 in the case of first-order integration). The sign of β is defined by

$$G(\mathbf{u}^*) > G(\mathbf{0}) : \beta_{\text{CDF}} < 0, \beta_{\text{CCDF}} > 0 \tag{2.19}$$

$$G(\mathbf{u}^*) < G(\mathbf{0}) : \beta_{\text{CDF}} > 0, \beta_{\text{CCDF}} < 0 \tag{2.20}$$

where $G(\mathbf{0})$ is the median limit state response computed at the origin in u-space[2] (where $\beta_{\text{CDF}} = \beta_{\text{CCDF}} = 0$ and first-order $p(g \leq z) = p(g > z) = 0.5$). In the PMA case, the sign applied to $G(\mathbf{u})$ (equivalent to minimizing or maximizing $G(\mathbf{u})$) is similarly defined by either $\bar{\beta}$ (for a specified reliability or first-order probability level) or from a $\bar{\beta}$ estimate[3] computed from $\bar{\beta}^*$ (for a specified generalized reliability or second-order probability level)

$$\bar{\beta}_{\text{CDF}} < 0, \bar{\beta}_{\text{CCDF}} > 0 : \text{maximize } G(\mathbf{u}) \tag{2.21}$$

$$\bar{\beta}_{\text{CDF}} > 0, \bar{\beta}_{\text{CCDF}} < 0 : \text{minimize } G(\mathbf{u}) \tag{2.22}$$

where the limit state at the MPP ($G(\mathbf{u}^*)$) defines the desired response level result.

2.1.2.1 Limit state approximations

There are a variety of algorithmic variations that are available for use within RIA/PMA reliability analyses. First, one may select among several different limit state approximations that can be used to reduce computational expense during the MPP searches. Local, multipoint, and global approximations of the limit state are possible. [45] investigated local first-order limit state approximations, and [46] investigated local second-order and multipoint approximations. These techniques include:

1. a single Taylor series per response/reliability/probability level in x-space centered at the uncertain variable means. The first-order approach is commonly known as the Advanced Mean Value (AMV) method:

$$g(\mathbf{x}) \cong g(\mu_{\mathbf{x}}) + \nabla_x g(\mu_{\mathbf{x}})^T (\mathbf{x} - \mu_{\mathbf{x}}) \tag{2.23}$$

   and the second-order approach has been named AMV^2:

$$g(\mathbf{x}) \cong g(\mu_{\mathbf{x}}) + \nabla_{\mathbf{x}} g(\mu_{\mathbf{x}})^T (\mathbf{x} - \mu_{\mathbf{x}}) + \frac{1}{2}(\mathbf{x} - \mu_{\mathbf{x}})^T \nabla^2_{\mathbf{x}} g(\mu_{\mathbf{x}})(\mathbf{x} - \mu_{\mathbf{x}}) \tag{2.24}$$

2. same as AMV/AMV^2, except that the Taylor series is expanded in u-space. The first-order option has been termed the u-space AMV method:

$$G(\mathbf{u}) \cong G(\mu_{\mathbf{u}}) + \nabla_u G(\mu_{\mathbf{u}})^T (\mathbf{u} - \mu_{\mathbf{u}}) \tag{2.25}$$

   where $\mu_{\mathbf{u}} = T(\mu_{\mathbf{x}})$ and is nonzero in general, and the second-order option has been named the u-space AMV^2 method:

$$G(\mathbf{u}) \cong G(\mu_{\mathbf{u}}) + \nabla_{\mathbf{u}} G(\mu_{\mathbf{u}})^T (\mathbf{u} - \mu_{\mathbf{u}}) + \frac{1}{2}(\mathbf{u} - \mu_{\mathbf{u}})^T \nabla^2_{\mathbf{u}} G(\mu_{\mathbf{u}})(\mathbf{u} - \mu_{\mathbf{u}}) \tag{2.26}$$

3. an initial Taylor series approximation in x-space at the uncertain variable means, with iterative expansion updates at each MPP estimate ($\mathbf{x}^*$) until the MPP converges. The first-order option is commonly known as AMV+:

$$g(\mathbf{x}) \cong g(\mathbf{x}^*) + \nabla_x g(\mathbf{x}^*)^T (\mathbf{x} - \mathbf{x}^*) \tag{2.27}$$

[2] It is not necessary to explicitly compute the median response since the sign of the inner product $\langle \mathbf{u}^*, \nabla_{\mathbf{u}} G \rangle$ can be used to determine the orientation of the optimal response with respect to the median response.

[3] computed by inverting the second-order probability relationships described in Section 2.1.2.2 at the current $\mathbf{u}^*$ iterate.

and the second-order option has been named AMV^2+:

$$g(\mathbf{x}) \cong g(\mathbf{x}^*) + \nabla_{\mathbf{x}} g(\mathbf{x}^*)^T(\mathbf{x}-\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T\nabla^2_{\mathbf{x}} g(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*) \tag{2.28}$$

4. same as AMV+/AMV^2+, except that the expansions are performed in u-space. The first-order option has been termed the u-space AMV+ method.

$$G(\mathbf{u}) \cong G(\mathbf{u}^*) + \nabla_u G(\mathbf{u}^*)^T(\mathbf{u}-\mathbf{u}^*) \tag{2.29}$$

and the second-order option has been named the u-space AMV^2+ method:

$$G(\mathbf{u}) \cong G(\mathbf{u}^*) + \nabla_{\mathbf{u}} G(\mathbf{u}^*)^T(\mathbf{u}-\mathbf{u}^*) + \frac{1}{2}(\mathbf{u}-\mathbf{u}^*)^T\nabla^2_{\mathbf{u}} G(\mathbf{u}^*)(\mathbf{u}-\mathbf{u}^*) \tag{2.30}$$

5. a multipoint approximation in x-space. This approach involves a Taylor series approximation in intermediate variables where the powers used for the intermediate variables are selected to match information at the current and previous expansion points. Based on the two-point exponential approximation concept (TPEA, [53]), the two-point adaptive nonlinearity approximation (TANA-3, [144]) approximates the limit state as:

$$g(\mathbf{x}) \cong g(\mathbf{x}_2) + \sum_{i=1}^{n} \frac{\partial g}{\partial x_i}(\mathbf{x}_2)\frac{x_{i,2}^{1-p_i}}{p_i}(x_i^{p_i} - x_{i,2}^{p_i}) + \frac{1}{2}\epsilon(\mathbf{x})\sum_{i=1}^{n}(x_i^{p_i} - x_{i,2}^{p_i})^2 \tag{2.31}$$

where n is the number of uncertain variables and:

$$p_i = 1 + \ln\left[\frac{\frac{\partial g}{\partial x_i}(\mathbf{x}_1)}{\frac{\partial g}{\partial x_i}(\mathbf{x}_2)}\right] \Big/ \ln\left[\frac{x_{i,1}}{x_{i,2}}\right] \tag{2.32}$$

$$\epsilon(\mathbf{x}) = \frac{H}{\sum_{i=1}^{n}(x_i^{p_i} - x_{i,1}^{p_i})^2 + \sum_{i=1}^{n}(x_i^{p_i} - x_{i,2}^{p_i})^2} \tag{2.33}$$

$$H = 2\left[g(\mathbf{x}_1) - g(\mathbf{x}_2) - \sum_{i=1}^{n}\frac{\partial g}{\partial x_i}(\mathbf{x}_2)\frac{x_{i,2}^{1-p_i}}{p_i}(x_{i,1}^{p_i} - x_{i,2}^{p_i})\right] \tag{2.34}$$

and $\mathbf{x}_2$ and $\mathbf{x}_1$ are the current and previous MPP estimates in x-space, respectively. Prior to the availability of two MPP estimates, x-space AMV+ is used.

6. a multipoint approximation in u-space. The u-space TANA-3 approximates the limit state as:

$$G(\mathbf{u}) \cong G(\mathbf{u}_2) + \sum_{i=1}^{n} \frac{\partial G}{\partial u_i}(\mathbf{u}_2)\frac{u_{i,2}^{1-p_i}}{p_i}(u_i^{p_i} - u_{i,2}^{p_i}) + \frac{1}{2}\epsilon(\mathbf{u})\sum_{i=1}^{n}(u_i^{p_i} - u_{i,2}^{p_i})^2 \tag{2.35}$$

where:

$$p_i = 1 + \ln\left[\frac{\frac{\partial G}{\partial u_i}(\mathbf{u}_1)}{\frac{\partial G}{\partial u_i}(\mathbf{u}_2)}\right] \Big/ \ln\left[\frac{u_{i,1}}{u_{i,2}}\right] \tag{2.36}$$

$$\epsilon(\mathbf{u}) = \frac{H}{\sum_{i=1}^{n}(u_i^{p_i} - u_{i,1}^{p_i})^2 + \sum_{i=1}^{n}(u_i^{p_i} - u_{i,2}^{p_i})^2} \tag{2.37}$$

$$H = 2\left[G(\mathbf{u}_1) - G(\mathbf{u}_2) - \sum_{i=1}^{n}\frac{\partial G}{\partial u_i}(\mathbf{u}_2)\frac{u_{i,2}^{1-p_i}}{p_i}(u_{i,1}^{p_i} - u_{i,2}^{p_i})\right] \tag{2.38}$$

and $\mathbf{u}_2$ and $\mathbf{u}_1$ are the current and previous MPP estimates in u-space, respectively. Prior to the availability of two MPP estimates, u-space AMV+ is used.

7. the MPP search on the original response functions without the use of any approximations. Combining this option with first-order and second-order integration approaches (see next section) results in the traditional first-order and second-order reliability methods (FORM and SORM).

The Hessian matrices in AMV^2 and AMV^2+ may be available analytically, estimated numerically, or approximated through quasi-Newton updates. The selection between x-space or u-space for performing approximations depends on where the approximation will be more accurate, since this will result in more accurate MPP estimates (AMV, AMV^2) or faster convergence (AMV+, AMV^2+, TANA). Since this relative accuracy depends on the forms of the limit state $g(x)$ and the transformation $T(x)$ and is therefore application dependent in general, Dakota supports both options. A concern with approximation-based iterative search methods (i.e., AMV+, AMV^2+ and TANA) is the robustness of their convergence to the MPP. It is possible for the MPP iterates to oscillate or even diverge. However, to date, this occurrence has been relatively rare, and Dakota contains checks that monitor for this behavior. Another concern with TANA is numerical safeguarding (e.g., the possibility of raising negative x_i or u_i values to nonintegral p_i exponents in Equations 2.31, 2.33-2.35, and 2.37-2.38). Safeguarding involves offseting negative x_i or u_i and, for potential numerical difficulties with the logarithm ratios in Equations 2.32 and 2.36, reverting to either the linear ($p_i = 1$) or reciprocal ($p_i = -1$) approximation based on which approximation has lower error in $\frac{\partial g}{\partial x_i}(\mathbf{x}_1)$ or $\frac{\partial G}{\partial u_i}(\mathbf{u}_1)$.

2.1.2.2 Probability integrations

The second algorithmic variation involves the integration approach for computing probabilities at the MPP, which can be selected to be first-order (Equations 2.6-2.7) or second-order integration. Second-order integration involves applying a curvature correction [16, 74, 75]. Breitung applies a correction based on asymptotic analysis [16]:

$$p = \Phi(-\beta_p) \prod_{i=1}^{n-1} \frac{1}{\sqrt{1 + \beta_p \kappa_i}} \tag{2.39}$$

where κ_i are the principal curvatures of the limit state function (the eigenvalues of an orthonormal transformation of $\nabla^2_{\mathbf{u}} G$, taken positive for a convex limit state) and $\beta_p \geq 0$ (a CDF or CCDF probability correction is selected to obtain the correct sign for β_p). An alternate correction in [74] is consistent in the asymptotic regime ($\beta_p \rightarrow \infty$) but does not collapse to first-order integration for $\beta_p = 0$:

$$p = \Phi(-\beta_p) \prod_{i=1}^{n-1} \frac{1}{\sqrt{1 + \psi(-\beta_p)\kappa_i}} \tag{2.40}$$

where $\psi() = \frac{\phi()}{\Phi()}$ and $\phi()$ is the standard normal density function. [75] applies further corrections to Equation 2.40 based on point concentration methods. At this time, all three approaches are available within the code, but the Hohenbichler-Rackwitz correction is used by default (switching the correction is a compile-time option in the source code and has not been exposed in the input specification).

2.1.2.3 Hessian approximations

To use a second-order Taylor series or a second-order integration when second-order information ($\nabla^2_{\mathbf{x}} g$, $\nabla^2_{\mathbf{u}} G$, and/or κ) is not directly available, one can estimate the missing information using finite differences or approximate it through use of quasi-Newton approximations. These procedures will often be needed to make second-order approaches practical for engineering applications.

In the finite difference case, numerical Hessians are commonly computed using either first-order forward differences of gradients using

$$\nabla^2 g(\mathbf{x}) \cong \frac{\nabla g(\mathbf{x} + h\mathbf{e}_i) - \nabla g(\mathbf{x})}{h} \tag{2.41}$$

to estimate the i^{th} Hessian column when gradients are analytically available, or second-order differences of function values using

$$\nabla^2 g(\mathbf{x}) \cong \frac{g(\mathbf{x}+h\mathbf{e}_i+h\mathbf{e}_j)-g(\mathbf{x}+h\mathbf{e}_i-h\mathbf{e}_j)-g(\mathbf{x}-h\mathbf{e}_i+h\mathbf{e}_j)+g(\mathbf{x}-h\mathbf{e}_i-h\mathbf{e}_j)}{4h^2} \tag{2.42}$$

to estimate the ij^{th} Hessian term when gradients are not directly available. This approach has the advantage of locally-accurate Hessians for each point of interest (which can lead to quadratic convergence rates in discrete Newton methods), but has the disadvantage that numerically estimating each of the matrix terms can be expensive.

Quasi-Newton approximations, on the other hand, do not reevaluate all of the second-order information for every point of interest. Rather, they accumulate approximate curvature information over time using secant updates. Since they utilize the existing gradient evaluations, they do not require any additional function evaluations for evaluating the Hessian terms. The quasi-Newton approximations of interest include the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{\mathbf{B}_k \mathbf{s}_k \mathbf{s}_k^T \mathbf{B}_k}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \tag{2.43}$$

which yields a sequence of symmetric positive definite Hessian approximations, and the Symmetric Rank 1 (SR1) update

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T}{(\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)^T \mathbf{s}_k} \tag{2.44}$$

which yields a sequence of symmetric, potentially indefinite, Hessian approximations. $\mathbf{B}_k$ is the k^{th} approximation to the Hessian $\nabla^2 g$, $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ is the step and $\mathbf{y}_k = \nabla g_{k+1} - \nabla g_k$ is the corresponding yield in the gradients. The selection of BFGS versus SR1 involves the importance of retaining positive definiteness in the Hessian approximations; if the procedure does not require it, then the SR1 update can be more accurate if the true Hessian is not positive definite. Initial scalings for $\mathbf{B}_0$ and numerical safeguarding techniques (damped BFGS, update skipping) are described in [46].

2.1.2.4 Optimization algorithms

The next algorithmic variation involves the optimization algorithm selection for solving Eqs. 2.16 and 2.17. The Hasofer-Lind Rackwitz-Fissler (HL-RF) algorithm [70] is a classical approach that has been broadly applied. It is a Newton-based approach lacking line search/trust region globalization, and is generally regarded as computationally efficient but occasionally unreliable. Dakota takes the approach of employing robust, general-purpose optimization algorithms with provable convergence properties. In particular, we employ the sequential quadratic programming (SQP) and nonlinear interior-point (NIP) optimization algorithms from the NPSOL [65] and OPT++ [94] libraries, respectively.

2.1.2.5 Warm Starting of MPP Searches

The final algorithmic variation for local reliability methods involves the use of warm starting approaches for improving computational efficiency. [45] describes the acceleration of MPP searches through warm starting with approximate iteration increment, with $z/p/\beta$ level increment, and with design variable increment. Warm started data includes the expansion point and associated response values and the MPP optimizer initial guess. Projections

are used when an increment in $z/p/\beta$ level or design variables occurs. Warm starts were consistently effective in [45], with greater effectiveness for smaller parameter changes, and are used by default in Dakota.

2.2 Global Reliability Methods

Local reliability methods, while computationally efficient, have well-known failure mechanisms. When confronted with a limit state function that is nonsmooth, local gradient-based optimizers may stall due to gradient inaccuracy and fail to converge to an MPP. Moreover, if the limit state is multimodal (multiple MPPs), then a gradient-based local method can, at best, locate only one local MPP solution. Finally, a linear (Eqs. 2.6–2.7) or parabolic (Eqs. 2.39–2.40) approximation to the limit state at this MPP may fail to adequately capture the contour of a highly nonlinear limit state.

A reliability analysis method that is both efficient when applied to expensive response functions and accurate for a response function of any arbitrary shape is needed. This section develops such a method based on efficient global optimization [79] (EGO) to the search for multiple points on or near the limit state throughout the random variable space. By locating multiple points on the limit state, more complex limit states can be accurately modeled, resulting in a more accurate assessment of the reliability. It should be emphasized here that these multiple points exist on a single limit state. Because of its roots in efficient global optimization, this method of reliability analysis is called efficient global reliability analysis (EGRA) [12]. The following two subsections describe two capabilities that are incorporated into the EGRA algorithm: importance sampling and EGO.

2.2.1 Importance Sampling

An alternative to MPP search methods is to directly perform the probability integration numerically by sampling the response function. Sampling methods do not rely on a simplifying approximation to the shape of the limit state, so they can be more accurate than FORM and SORM, but they can also be prohibitively expensive because they generally require a large number of response function evaluations. Importance sampling methods reduce this expense by focusing the samples in the important regions of the uncertain space. They do this by centering the sampling density function at the MPP rather than at the mean. This ensures the samples will lie the region of interest, thus increasing the efficiency of the sampling method. Adaptive importance sampling (AIS) further improves the efficiency by adaptively updating the sampling density function. Multimodal adaptive importance sampling [38, 147] is a variation of AIS that allows for the use of multiple sampling densities making it better suited for cases where multiple sections of the limit state are highly probable.

Note that importance sampling methods require that the location of at least one MPP be known because it is used to center the initial sampling density. However, current gradient-based, local search methods used in MPP search may fail to converge or may converge to poor solutions for highly nonlinear problems, possibly making these methods inapplicable. As the next section describes, EGO is a global optimization method that does not depend on the availability of accurate gradient information, making convergence more reliable for nonsmooth response functions. Moreover, EGO has the ability to locate multiple failure points, which would provide multiple starting points and thus a good multimodal sampling density for the initial steps of multimodal AIS. The resulting Gaussian process model is accurate in the vicinity of the limit state, thereby providing an inexpensive surrogate that can be used to provide response function samples. As will be seen, using EGO to locate multiple points along the limit state, and then using the resulting Gaussian process model to provide function evaluations in multimodal AIS for the probability integration, results in an accurate and efficient reliability analysis tool.

2.2.2 Efficient Global Optimization

Chapter 8 is now rewritten to support EGO/Bayesian optimization theory.

2.2.2.1 Expected Feasibility Function

The expected improvement function provides an indication of how much the true value of the response at a point can be expected to be less than the current best solution. It therefore makes little sense to apply this to the forward reliability problem where the goal is not to minimize the response, but rather to find where it is equal to a specified threshold value. The expected feasibility function (EFF) is introduced here to provide an indication of how well the true value of the response is expected to satisfy the equality constraint $G(\mathbf{u}) = \bar{z}$. Inspired by the contour estimation work in [112], this expectation can be calculated in a similar fashion as Eq. 8.11 by integrating over a region in the immediate vicinity of the threshold value $\bar{z} \pm \epsilon$:

$$EF\big(\hat{G}(\mathbf{u})\big) = \int_{z-\epsilon}^{z+\epsilon} \big[\epsilon - |\bar{z} - G|\big]\, \hat{G}(\mathbf{u})\, dG \tag{2.45}$$

where G denotes a realization of the distribution $\hat{G}$, as before. Allowing z^+ and z^- to denote $\bar{z} \pm \epsilon$, respectively, this integral can be expressed analytically as:

$$\begin{aligned} EF\big(\hat{G}(\mathbf{u})\big) = (\mu_G - \bar{z}) &\left[2\,\Phi\left(\frac{\bar{z} - \mu_G}{\sigma_G}\right) - \Phi\left(\frac{z^- - \mu_G}{\sigma_G}\right) - \Phi\left(\frac{z^+ - \mu_G}{\sigma_G}\right)\right] \\ - \sigma_G &\left[2\,\phi\left(\frac{\bar{z} - \mu_G}{\sigma_G}\right) - \phi\left(\frac{z^- - \mu_G}{\sigma_G}\right) - \phi\left(\frac{z^+ - \mu_G}{\sigma_G}\right)\right] \\ + \quad \epsilon &\left[\Phi\left(\frac{z^+ - \mu_G}{\sigma_G}\right) - \Phi\left(\frac{z^- - \mu_G}{\sigma_G}\right)\right] \end{aligned} \tag{2.46}$$

where ϵ is proportional to the standard deviation of the GP predictor ($\epsilon \propto \sigma_G$). In this case, z^-, z^+, μ_G, σ_G, and ϵ are all functions of the location $\mathbf{u}$, while $\bar{z}$ is a constant. Note that the EFF provides the same balance between exploration and exploitation as is captured in the EIF. Points where the expected value is close to the threshold ($\mu_G \approx \bar{z}$) and points with a large uncertainty in the prediction will have large expected feasibility values.

Chapter 3

Stochastic Expansion Methods

This chapter explores two approaches to forming stochastic expansions, the polynomial chaos expansion (PCE), which employs bases of multivariate orthogonal polynomials, and stochastic collocation (SC), which employs bases of multivariate interpolation polynomials. Both approaches capture the functional relationship between a set of output response metrics and a set of input random variables.

3.1 Orthogonal polynomials

3.1.1 Askey scheme

Table 3.1 shows the set of classical orthogonal polynomials which provide an optimal basis for different continuous probability distribution types. It is derived from the family of hypergeometric orthogonal polynomials known as the Askey scheme [7], for which the Hermite polynomials originally employed by Wiener [136] are a subset. The optimality of these basis selections derives from their orthogonality with respect to weighting functions that correspond to the probability density functions (PDFs) of the continuous distributions when placed in a standard form. The density and weighting functions differ by a constant factor due to the requirement that the integral of the PDF over the support range is one.

Table 3.1: Linkage between standard forms of continuous probability distributions and Askey scheme of continuous hyper-geometric polynomials.

Distribution	Density function	Polynomial	Weight function	Support range
Normal	$\frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$	Hermite $He_n(x)$	$e^{\frac{-x^2}{2}}$	$[-\infty, \infty]$
Uniform	$\frac{1}{2}$	Legendre $P_n(x)$	1	$[-1, 1]$
Beta	$\frac{(1-x)^\alpha (1+x)^\beta}{2^{\alpha+\beta+1} B(\alpha+1, \beta+1)}$	Jacobi $P_n^{(\alpha,\beta)}(x)$	$(1-x)^\alpha (1+x)^\beta$	$[-1, 1]$
Exponential	e^{-x}	Laguerre $L_n(x)$	e^{-x}	$[0, \infty]$
Gamma	$\frac{x^\alpha e^{-x}}{\Gamma(\alpha+1)}$	Generalized Laguerre $L_n^{(\alpha)}(x)$	$x^\alpha e^{-x}$	$[0, \infty]$

Note that Legendre is a special case of Jacobi for $\alpha = \beta = 0$, Laguerre is a special case of generalized Laguerre for $\alpha = 0$, $\Gamma(a)$ is the Gamma function which extends the factorial function to continuous values, and $B(a, b)$ is the Beta function defined as $B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$. Some care is necessary when specifying the α and β parameters

for the Jacobi and generalized Laguerre polynomials since the orthogonal polynomial conventions [1] differ from the common statistical PDF conventions. The former conventions are used in Table 3.1.

3.1.2 Numerically generated orthogonal polynomials

If all random inputs can be described using independent normal, uniform, exponential, beta, and gamma distributions, then Askey polynomials can be directly applied. If correlation or other distribution types are present, then additional techniques are required. One solution is to employ nonlinear variable transformations as described in Section 3.5 such that an Askey basis can be applied in the transformed space. This can be effective as shown in [51], but convergence rates are typically degraded. In addition, correlation coefficients are warped by the nonlinear transformation [36], and simple expressions for these transformed correlation values are not always readily available. An alternative is to numerically generate the orthogonal polynomials (using Gauss-Wigert [120], discretized Stieltjes [59], Chebyshev [59], or Gramm-Schmidt [137] approaches) and then compute their Gauss points and weights (using the Golub-Welsch [69] tridiagonal eigensolution). These solutions are optimal for given random variable sets having arbitrary probability density functions and eliminate the need to induce additional nonlinearity through variable transformations, but performing this process for general joint density functions with correlation is a topic of ongoing research (refer to Section 3.5 for additional details).

3.2 Interpolation polynomials

Interpolation polynomials may be either local or global and either value-based or gradient-enhanced: Lagrange (global value-based), Hermite (global gradient-enhanced), piecewise linear spline (local value-based), and piecewise cubic spline (local gradient-enhanced). Each of these combinations can be used within nodal or hierarchical interpolation formulations. The subsections that follow describe the one-dimensional interpolation polynomials for these cases and Section 3.4 describes their use for multivariate interpolation within the stochastic collocation algorithm.

3.2.1 Nodal interpolation

For value-based interpolation of a response function R in one dimension at an interpolation level l containing m^l points, the expression

$$R(\xi) \cong I^l(R) = \sum_{j=1}^{m_l} r(\xi_j)\, L_j(\xi) \tag{3.1}$$

reproduces the response values $r(\xi_j)$ at the interpolation points and smoothly interpolates between these values at other points. As we refine the interpolation level, we increase the number of collocation points in the rule and the number of interpolated response values.

For the case of gradient-enhancement, interpolation of a one-dimensional function involves both type 1 and type 2 interpolation polynomials,

$$R(\xi) \cong I^l(R) = \sum_{j=1}^{m_l} \left[r(\xi_j) H_j^{(1)}(\xi) + \frac{dr}{d\xi}(\xi_j) H_j^{(2)}(\xi) \right] \tag{3.2}$$

where the former interpolate a particular value while producing a zero gradient (i^{th} type 1 interpolant produces a value of 1 for the i^{th} collocation point, zero values for all other points, and zero gradients for all points) and the

latter interpolate a particular gradient while producing a zero value (i^{th} type 2 interpolant produces a gradient of 1 for the i^{th} collocation point, zero gradients for all other points, and zero values for all points).

3.2.1.1 Global value-based

Lagrange polynomials interpolate a set of points in a single dimension using the functional form

$$L_j = \prod_{\substack{k=1 \\ k \neq j}}^{m} \frac{\xi - \xi_k}{\xi_j - \xi_k} \tag{3.3}$$

where it is evident that L_j is 1 at $\xi = \xi_j$, is 0 for each of the points $\xi = \xi_k$, and has order $m - 1$.

To improve numerical efficiency and stability, a barycentric Lagrange formulation [11, 72] is used. We define the barycentric weights w_j as

$$w_j = \prod_{\substack{k=1 \\ k \neq j}}^{m} \frac{1}{\xi_j - \xi_k} \tag{3.4}$$

and we precompute them for a given interpolation point set $\xi_j, j \in 1, ..., m$. Then, defining the quantity $l(\xi)$ as

$$l(\xi) = \prod_{k=1}^{m} (\xi - \xi_k) \tag{3.5}$$

which will be computed for each new interpolated point ξ, we can rewrite Eq. 3.1 as

$$R(\xi) = l(\xi) \sum_{j=1}^{m} \frac{w_j}{x - x_j} r(\xi_j) \tag{3.6}$$

where much of the computational work has been moved outside the summation. Eq. 3.6 is the first form of barycentric interpolation. Using an identity from the interpolation of unity ($R(\xi) = 1$ and each $r(\xi_j) = 1$ in Eq. 3.6) to eliminate $l(x)$, we arrive at the second form of the barycentric interpolation formula:

$$R(\xi) = \frac{\sum_{j=1}^{m} \frac{w_j}{x - x_j} r(\xi_j)}{\sum_{j=1}^{m} \frac{w_j}{x - x_j}} \tag{3.7}$$

For both formulations, we reduce the computational effort for evaluating the interpolant from $O(m^2)$ to $O(m)$ operations per interpolated point, with the penalty of requiring additional care to avoid division by zero when ξ matches one of the ξ_j. Relative to the first form, the second form has the additional advantage that common factors within the w_j can be canceled (possible for Clenshaw-Curtis and Newton-Cotes point sets, but not for general Gauss points), further reducing the computational requirements. Barycentric formulations can also be used for hierarchical interpolation (Section 3.2.2) with Lagrange interpolation polynomials, but they are not applicable to local spline or gradient-enhanced Hermite interpolants.

3.2.1.2 Global gradient-enhanced

Hermite interpolation polynomials (not to be confused with Hermite orthogonal polynomials shown in Table 3.1) interpolate both values and derivatives. In our case, we are interested in interpolating values and first derivatives, i.e, gradients. One-dimensional polynomials satisfying the interpolation constraints for general point sets are generated using divided differences as described in [18].

3.2.1.3 Local value-based

Linear spline basis polynomials define a "hat function," which produces the value of one at its collocation point and decays linearly to zero at its nearest neighbors. In the case where its collocation point corresponds to a domain boundary, then the half interval that extends beyond the boundary is truncated.

For the case of non-equidistant closed points (e.g., Clenshaw-Curtis), the linear spline polynomials are defined as

$$L_j(\xi) = \begin{cases} 1 - \frac{\xi - \xi_j}{\xi_{j-1} - \xi_j} & \text{if } \xi_{j-1} \leq \xi \leq \xi_j \text{ (left half interval)} \\ 1 - \frac{\xi - \xi_j}{\xi_{j+1} - \xi_j} & \text{if } \xi_j < \xi \leq \xi_{j+1} \text{ (right half interval)} \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

For the case of equidistant closed points (i.e., Newton-Cotes), this can be simplified to

$$L_j(\xi) = \begin{cases} 1 - \frac{|\xi - \xi_j|}{h} & \text{if } |\xi - \xi_j| \leq h \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

for h defining the half-interval $\frac{b-a}{m-1}$ of the hat function L_j over the range $\xi \in [a, b]$. For the special case of $m = 1$ point, $L_1(\xi) = 1$ for $\xi_1 = \frac{b+a}{2}$ in both cases above.

3.2.1.4 Local gradient-enhanced

Type 1 cubic spline interpolants are formulated as follows:

$$H_j^{(1)}(\xi) = \begin{cases} t^2(3-2t) \text{ for } t = \frac{\xi - \xi_{j-1}}{\xi_j - \xi_{j-1}} & \text{if } \xi_{j-1} \leq \xi \leq \xi_j \text{ (left half interval)} \\ (t-1)^2(1+2t) \text{ for } t = \frac{\xi - \xi_j}{\xi_{j+1} - \xi_j} & \text{if } \xi_j < \xi \leq \xi_{j+1} \text{ (right half interval)} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

which produce the desired zero-one-zero property for left-center-right values and zero-zero-zero property for left-center-right gradients. Type 2 cubic spline interpolants are formulated as follows:

$$H_j^{(2)}(\xi) = \begin{cases} ht^2(t-1) \text{ for } h = \xi_j - \xi_{j-1}, \ t = \frac{\xi - \xi_{j-1}}{h} & \text{if } \xi_{j-1} \leq \xi \leq \xi_j \text{ (left half interval)} \\ ht(t-1)^2 \text{ for } h = \xi_{j+1} - \xi_j, \ t = \frac{\xi - \xi_j}{h} & \text{if } \xi_j < \xi \leq \xi_{j+1} \text{ (right half interval)} \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

which produce the desired zero-zero-zero property for left-center-right values and zero-one-zero property for left-center-right gradients. For the special case of $m = 1$ point over the range $\xi \in [a, b]$, $H_1^{(1)}(\xi) = 1$ and $H_1^{(2)}(\xi) = \xi$ for $\xi_1 = \frac{b+a}{2}$.

3.2.2 Hierarchical interpolation

In a hierarchical formulation, we reformulate the interpolation in terms of differences between interpolation levels:

$$\Delta^l(R) = I^l(R) - I^{l-1}(R), \ \ l \geq 1 \tag{3.12}$$

where $I^l(R)$ is as defined in Eqs. 3.1–3.2 using the same local or global definitions for $L_j(\xi)$, $H_j^{(1)}(\xi)$, and $H_j^{(2)}(\xi)$, and $I^{l-1}(R)$ is evaluated as $I^l(I^{l-1}(R))$, indicating reinterpolation of the lower level interpolant across the higher level point set [82, 4].

Utilizing Eqs. 3.1–3.2, we can represent this difference interpolant as

$$\Delta^l(R) = \begin{cases} \sum_{j=1}^{m_l} \left[r(\xi_j) - I^{l-1}(R)(\xi_j)\right]\, L_j(\xi) & \text{value-based} \\ \sum_{j=1}^{m_l} \left(\left[r(\xi_j) - I^{l-1}(R)(\xi_j)\right]\, H_j^{(1)}(\xi) + \left[\frac{dr}{d\xi}(\xi_j) - \frac{dI^{l-1}(R)}{d\xi}(\xi_j)\right]\, H_j^{(2)}(\xi)\right) & \text{gradient-enhanced} \end{cases} \tag{3.13}$$

where $I^{l-1}(R)(\xi_j)$ and $\frac{dI^{l-1}(R)}{d\xi}(\xi_j)$ are the value and gradient, respectively, of the lower level interpolant evaluated at the higher level points. We then define hierarchical surpluses $s, s^{(1)}, s^{(2)}$ at a point ξ_j as the bracketed terms in Eq 3.13. These surpluses can be interpreted as local interpolation error estimates since they capture the difference between the true values and the values predicted by the previous interpolant.

For the case where we use nested point sets among the interpolation levels, the interpolant differences for points contained in both sets are zero, allowing us to restrict the summations above to $\sum_{j=1}^{m_{\Delta_l}}$ where we define the set $\Xi_{\Delta_l} = \Xi_l \setminus \Xi_{l-1}$ that contains $m_{\Delta_l} = m_l - m_{l-1}$ points. $\Delta^l(R)$ then becomes

$$\Delta^l(R) = \begin{cases} \sum_{j=1}^{m_{\Delta_l}} s(\xi_j)\, L_j(\xi) & \text{value-based} \\ \sum_{j=1}^{m_{\Delta_l}} \left(s^{(1)}(\xi_j)\, H_j^{(1)}(\xi) + s^{(2)}(\xi_j)\, H_j^{(2)}(\xi)\right) & \text{gradient-enhanced} \end{cases} \tag{3.14}$$

The original interpolant $I^l(R)$ can be represented as a summation of these difference interpolants

$$I^l(R) = \Delta^l(R) + I^{l-1}(R) = \sum_{i=1}^{l} \Delta^l(R) \tag{3.15}$$

We will employ these hierarchical definitions within stochastic collocation on sparse grids in Section 3.4.3.

3.3 Generalized Polynomial Chaos

The set of polynomials from 3.1.1 and 3.1.2 are used as an orthogonal basis to approximate the functional form between the stochastic response output and each of its random inputs. The chaos expansion for a response R takes the form

$$R = a_0 B_0 + \sum_{i_1=1}^{\infty} a_{i_1} B_1(\xi_{i_1}) + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} a_{i_1 i_2} B_2(\xi_{i_1}, \xi_{i_2}) + \sum_{i_1=1}^{\infty} \sum_{i_2=1}^{i_1} \sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3} B_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3}) + \ldots \tag{3.16}$$

where the random vector dimension is unbounded and each additional set of nested summations indicates an additional order of polynomials in the expansion. This expression can be simplified by replacing the order-based indexing with a term-based indexing

$$R = \sum_{j=0}^{\infty} \alpha_j \Psi_j(\boldsymbol{\xi}) \tag{3.17}$$

where there is a one-to-one correspondence between $a_{i_1 i_2 \ldots i_n}$ and α_j and between $B_n(\xi_{i_1}, \xi_{i_2}, \ldots, \xi_{i_n})$ and $\Psi_j(\boldsymbol{\xi})$. Each of the $\Psi_j(\boldsymbol{\xi})$ are multivariate polynomials which involve products of the one-dimensional polynomials. For example, a multivariate Hermite polynomial $B(\boldsymbol{\xi})$ of order n is defined from

$$B_n(\xi_{i_1}, \ldots, \xi_{i_n}) = e^{\frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi}} (-1)^n \frac{\partial^n}{\partial \xi_{i_1} \ldots \partial \xi_{i_n}} e^{-\frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi}} \tag{3.18}$$

which can be shown to be a product of one-dimensional Hermite polynomials involving an expansion term multi-index t_i^j:

$$B_n(\xi_{i_1}, \ldots, \xi_{i_n}) = \Psi_j(\boldsymbol{\xi}) = \prod_{i=1}^{n} \psi_{t_i^j}(\xi_i) \tag{3.19}$$

In the case of a mixed basis, the same multi-index definition is employed although the one-dimensional polynomials $\psi_{t_i^j}$ are heterogeneous in type.

3.3.1 Expansion truncation and tailoring

In practice, one truncates the infinite expansion at a finite number of random variables and a finite expansion order

$$R \cong \sum_{j=0}^{P} \alpha_j \Psi_j(\boldsymbol{\xi}) \tag{3.20}$$

Traditionally, the polynomial chaos expansion includes a complete basis of polynomials up to a fixed total-order specification. That is, for an expansion of total order p involving n random variables, the expansion term multi-index defining the set of Ψ_j is constrained by

$$\sum_{i=1}^{n} t_i^j \leq p \tag{3.21}$$

For example, the multidimensional basis polynomials for a second-order expansion over two random dimensions are

$$\begin{aligned}
\Psi_0(\boldsymbol{\xi}) &= \psi_0(\xi_1)\,\psi_0(\xi_2) &&= 1 \\
\Psi_1(\boldsymbol{\xi}) &= \psi_1(\xi_1)\,\psi_0(\xi_2) &&= \xi_1 \\
\Psi_2(\boldsymbol{\xi}) &= \psi_0(\xi_1)\,\psi_1(\xi_2) &&= \xi_2 \\
\Psi_3(\boldsymbol{\xi}) &= \psi_2(\xi_1)\,\psi_0(\xi_2) &&= \xi_1^2 - 1 \\
\Psi_4(\boldsymbol{\xi}) &= \psi_1(\xi_1)\,\psi_1(\xi_2) &&= \xi_1\xi_2 \\
\Psi_5(\boldsymbol{\xi}) &= \psi_0(\xi_1)\,\psi_2(\xi_2) &&= \xi_2^2 - 1
\end{aligned}$$

The total number of terms N_t in an expansion of total order p involving n random variables is given by

$$N_t \;=\; 1 + P \;=\; 1 + \sum_{s=1}^{p} \frac{1}{s!} \prod_{r=0}^{s-1} (n+r) \;=\; \frac{(n+p)!}{n!p!} \tag{3.22}$$

This traditional approach will be referred to as a "total-order expansion."

An important alternative approach is to employ a "tensor-product expansion," in which polynomial order bounds are applied on a per-dimension basis (no total-order bound is enforced) and all combinations of the one-dimensional polynomials are included. That is, the expansion term multi-index defining the set of Ψ_j is constrained by

$$t_i^j \leq p_i \tag{3.23}$$

where p_i is the polynomial order bound for the i^{th} dimension. In this case, the example basis for $p = 2, n = 2$ is

$$\begin{aligned}
\Psi_0(\boldsymbol{\xi}) &= \psi_0(\xi_1)\,\psi_0(\xi_2) &&= 1 \\
\Psi_1(\boldsymbol{\xi}) &= \psi_1(\xi_1)\,\psi_0(\xi_2) &&= \xi_1 \\
\Psi_2(\boldsymbol{\xi}) &= \psi_2(\xi_1)\,\psi_0(\xi_2) &&= \xi_1^2 - 1 \\
\Psi_3(\boldsymbol{\xi}) &= \psi_0(\xi_1)\,\psi_1(\xi_2) &&= \xi_2 \\
\Psi_4(\boldsymbol{\xi}) &= \psi_1(\xi_1)\,\psi_1(\xi_2) &&= \xi_1\xi_2 \\
\Psi_5(\boldsymbol{\xi}) &= \psi_2(\xi_1)\,\psi_1(\xi_2) &&= (\xi_1^2 - 1)\xi_2 \\
\Psi_6(\boldsymbol{\xi}) &= \psi_0(\xi_1)\,\psi_2(\xi_2) &&= \xi_2^2 - 1 \\
\Psi_7(\boldsymbol{\xi}) &= \psi_1(\xi_1)\,\psi_2(\xi_2) &&= \xi_1(\xi_2^2 - 1) \\
\Psi_8(\boldsymbol{\xi}) &= \psi_2(\xi_1)\,\psi_2(\xi_2) &&= (\xi_1^2 - 1)(\xi_2^2 - 1)
\end{aligned}$$

and the total number of terms N_t is

$$N_t = 1 + P = \prod_{i=1}^{n}(p_i + 1) \tag{3.24}$$

It is apparent from Eq. 3.24 that the tensor-product expansion readily supports anisotropy in polynomial order for each dimension, since the polynomial order bounds for each dimension can be specified independently. It is also feasible to support anisotropy with total-order expansions, through pruning polynomials that satisfy the total-order bound but violate individual per-dimension bounds (the number of these pruned polynomials would then be subtracted from Eq. 3.22). Finally, additional tailoring of the expansion form is used in the case of sparse grids (see Section 3.6.3) through the use of a summation of anisotropic tensor expansions. In all cases, the specifics of the expansion are codified in the term multi-index, and subsequent machinery for estimating response values and statistics from the expansion can be performed in a manner that is agnostic to the specific expansion form.

3.4 Stochastic Collocation

The SC expansion is formed as a sum of a set of multidimensional interpolation polynomials, one polynomial per interpolated response quantity (one response value and potentially multiple response gradient components) per unique collocation point.

3.4.1 Value-Based Nodal

For value-based interpolation in multiple dimensions, a tensor-product of the one-dimensional polynomials described in Section 3.2.1.1 or Section 3.2.1.3 is used:

$$R(\boldsymbol{\xi}) \cong \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_n=1}^{m_{i_n}} r\left(\xi_{j_1}^{i_1}, \ldots, \xi_{j_n}^{i_n}\right) \left(L_{j_1}^{i_1} \otimes \cdots \otimes L_{j_n}^{i_n}\right) \tag{3.25}$$

where $\boldsymbol{i} = (m_1, m_2, \cdots, m_n)$ are the number of nodes used in the n-dimensional interpolation and $\xi_{j_k}^{i_k}$ indicates the j^{th} point out of i possible collocation points in the k^{th} dimension. This can be simplified to

$$R(\boldsymbol{\xi}) \cong \sum_{j=1}^{N_p} r_j \boldsymbol{L}_j(\boldsymbol{\xi}) \tag{3.26}$$

where N_p is the number of unique collocation points in the multidimensional grid. The multidimensional interpolation polynomials are defined as

$$\boldsymbol{L}_j(\boldsymbol{\xi}) = \prod_{k=1}^{n} L_{c_k^j}(\xi_k) \tag{3.27}$$

where c_k^j is a collocation multi-index (similar to the expansion term multi-index in Eq. 3.19) that maps from the j^{th} unique collocation point to the corresponding multidimensional indices within the tensor grid, and we have dropped the superscript notation indicating the number of nodes in each dimension for simplicity. The tensor-product structure preserves the desired interpolation properties where the j^{th} multivariate interpolation polynomial assumes the value of 1 at the j^{th} point and assumes the value of 0 at all other points, thereby reproducing the response values at each of the collocation points and smoothly interpolating between these values at other unsampled points. When the one-dimensional interpolation polynomials are defined using a barycentric formulation as described in Section 3.2.1.1 (i.e., Eq. 3.7), additional efficiency in evaluating a tensor interpolant

is achieved using the procedure in [83], which amounts to a multi-dimensional extension to Horner's rule for tensor-product polynomial evaluation.

Multivariate interpolation on Smolyak sparse grids involves a weighted sum of the tensor products in Eq. 3.25 with varying $\boldsymbol{i}$ levels. For sparse interpolants based on nested quadrature rules (e.g., Clenshaw-Curtis, Gauss-Patterson, Genz-Keister), the interpolation property is preserved, but sparse interpolants based on non-nested rules may exhibit some interpolation error at the collocation points.

3.4.2 Gradient-Enhanced Nodal

For gradient-enhanced interpolation in multiple dimensions, we extend the formulation in Eq 3.26 to use a tensor-product of the one-dimensional type 1 and type 2 polynomials described in Section 3.2.1.2 or Section 3.2.1.4:

$$R(\boldsymbol{\xi}) \cong \sum_{j=1}^{N_p} \left[r_j \boldsymbol{H}_j^{(1)}(\boldsymbol{\xi}) + \sum_{k=1}^{n} \frac{dr_j}{d\xi_k} \boldsymbol{H}_{jk}^{(2)}(\boldsymbol{\xi}) \right] \tag{3.28}$$

The multidimensional type 1 basis polynomials are

$$\boldsymbol{H}_j^{(1)}(\boldsymbol{\xi}) = \prod_{k=1}^{n} H_{c_k^j}^{(1)}(\xi_k) \tag{3.29}$$

where c_k^j is the same collocation multi-index described for Eq. 3.27 and the superscript notation indicating the number of nodes in each dimension has again been omitted. The multidimensional type 2 basis polynomials for the k^{th} gradient component are the same as the type 1 polynomials for each dimension except k:

$$\boldsymbol{H}_{jk}^{(2)}(\boldsymbol{\xi}) = H_{c_k^j}^{(2)}(\xi_k) \prod_{\substack{l=1 \\ l \neq k}}^{n} H_{c_l^j}^{(1)}(\xi_l) \tag{3.30}$$

As for the value-based case, multivariate interpolation on Smolyak sparse grids involves a weighted sum of the tensor products in Eq. 3.28 with varying $\boldsymbol{i}$ levels.

3.4.3 Hierarchical

In the case of multivariate hierarchical interpolation on nested grids, we are interested in tensor products of the one-dimensional difference interpolants described in Section 3.2.2, with

$$\Delta^l(R) = \sum_{j_1=1}^{m_{\Delta_1}} \cdots \sum_{j_n=1}^{m_{\Delta_n}} s\left(\xi_{j_1}^{\Delta_1}, \ldots, \xi_{j_n}^{\Delta_n}\right) \left(L_{j_1}^{\Delta_1} \otimes \cdots \otimes L_{j_n}^{\Delta_n}\right) \tag{3.31}$$

for value-based, and

$$\begin{aligned} \Delta^l(R) \quad = & \sum_{j_1=1}^{m_{\Delta_1}} \cdots \sum_{j_n=1}^{m_{\Delta_n}} \\ & \left[s^{(1)}\left(\xi_{j_1}^{\Delta_1}, \ldots, \xi_{j_n}^{\Delta_n}\right) \left(H^{(1)}{}_{j_1}^{\Delta_1} \otimes \cdots \otimes H^{(1)}{}_{j_n}^{\Delta_n}\right) + \sum_{k=1}^{n} s_k^{(2)}\left(\xi_{j_1}^{\Delta_1}, \ldots, \xi_{j_n}^{\Delta_n}\right) \left(H_k^{(2)}{}_{j_1}^{\Delta_1} \otimes \cdots \otimes H_k^{(2)}{}_{j_n}^{\Delta_n}\right) \right] \end{aligned} \tag{3.32}$$

for gradient-enhanced, where k indicates the gradient component being interpolated.

These difference interpolants are particularly useful within sparse grid interpolation, for which the Δ^l can be employed directly within Eq. 3.40.

3.5 Transformations to uncorrelated standard variables

Polynomial chaos and stochastic collocation are expanded using polynomials that are functions of independent random variables $\boldsymbol{\xi}$, which are often standardized forms of common distributions. Thus, a key component of stochastic expansion approaches is performing a transformation of variables from the original random variables $\boldsymbol{x}$ to independent (standard) random variables $\boldsymbol{\xi}$ and then applying the stochastic expansion in the transformed space. This notion of independent standard space is extended over the notion of "u-space" used in reliability methods (see Section 2.1.2) in that it extends the standardized set beyond standard normals. For distributions that are already independent, three different approaches are of interest:

1. *Extended basis:* For each Askey distribution type, employ the corresponding Askey basis (Table 3.1). For non-Askey types, numerically generate an optimal polynomial basis for each independent distribution as described in Section 3.1.2. These numerically-generated basis polynomials are not coerced into any standardized form, but rather employ the actual distribution parameters of the individual random variables. Thus, not even a linear variable transformation is employed for these variables. With usage of the optimal basis corresponding to each of the random variable types, we avoid inducing additional nonlinearity that can slow convergence.

2. *Askey basis:* For non-Askey types, perform a nonlinear variable transformation from a given input distribution to the most similar Askey basis. For example, lognormal distributions might employ a Hermite basis in a transformed standard normal space and loguniform, triangular, and histogram distributions might employ a Legendre basis in a transformed standard uniform space. All distributions then employ the Askey orthogonal polynomials and their associated Gauss points/weights.

3. *Wiener basis:* For non-normal distributions, employ a nonlinear variable transformation to standard normal distributions. All distributions then employ the Hermite orthogonal polynomials and their associated Gauss points/weights.

For dependent distributions, we must first perform a nonlinear variable transformation to uncorrelated standard normal distributions, due to the independence of decorrelated standard normals. This involves the Nataf transformation, described in the following paragraph. We then have the following choices:

1. *Single transformation:* Following the Nataf transformation to independent standard normal distributions, employ the Wiener basis in the transformed space.

2. *Double transformation:* From independent standard normal space, transform back to either the original marginal distributions or the desired Askey marginal distributions and employ an extended or Askey basis, respectively, in the transformed space. Independence is maintained, but the nonlinearity of the Nataf transformation is at least partially mitigated.

Dakota does not yet implement the double transformation concept, such that each correlated variable will employ a Wiener basis approach.

The transformation from correlated non-normal distributions to uncorrelated standard normal distributions is denoted as $\boldsymbol{\xi} = T(\mathbf{x})$ with the reverse transformation denoted as $\mathbf{x} = T^{-1}(\boldsymbol{\xi})$. These transformations are nonlinear

in general, and possible approaches include the Rosenblatt [117], Nataf [36], and Box-Cox [14] transformations. Dakota employs the Nataf transformation, which is suitable for the common case when marginal distributions and a correlation matrix are provided, but full joint distributions are not known[1]. The Nataf transformation occurs in the following two steps. To transform between the original correlated x-space variables and correlated standard normals ("z-space"), a CDF matching condition is applied for each of the marginal distributions:

$$\Phi(z_i) = F(x_i) \tag{3.33}$$

where $\Phi()$ is the standard normal cumulative distribution function and $F()$ is the cumulative distribution function of the original probability distribution. Then, to transform between correlated z-space variables and uncorrelated ξ-space variables, the Cholesky factor $\mathbf{L}$ of a modified correlation matrix is used:

$$\mathbf{z} = \mathbf{L}\boldsymbol{\xi} \tag{3.34}$$

where the original correlation matrix for non-normals in x-space has been modified to represent the corresponding "warped" correlation in z-space [36].

3.6 Spectral projection

The major practical difference between PCE and SC is that, in PCE, one must estimate the coefficients for known basis functions, whereas in SC, one must form the interpolants for known coefficients. PCE estimates its coefficients using either spectral projection or linear regression, where the former approach involves numerical integration based on random sampling, tensor-product quadrature, Smolyak sparse grids, or cubature methods. In SC, the multidimensional interpolants need to be formed over structured data sets, such as point sets from quadrature or sparse grids; approaches based on random sampling may not be used.

The spectral projection approach projects the response against each basis function using inner products and employs the polynomial orthogonality properties to extract each coefficient. Similar to a Galerkin projection, the residual error from the approximation is rendered orthogonal to the selected basis. From Eq. 3.20, taking the inner product of both sides with respect to Ψ_j and enforcing orthogonality yields:

$$\alpha_j = \frac{\langle R, \Psi_j \rangle}{\langle \Psi_j^2 \rangle} = \frac{1}{\langle \Psi_j^2 \rangle} \int_\Omega R\, \Psi_j\, \varrho(\boldsymbol{\xi})\, d\boldsymbol{\xi}, \tag{3.35}$$

where each inner product involves a multidimensional integral over the support range of the weighting function. In particular, $\Omega = \Omega_1 \otimes \cdots \otimes \Omega_n$, with possibly unbounded intervals $\Omega_j \subset \mathbb{R}$ and the tensor product form $\varrho(\boldsymbol{\xi}) = \prod_{i=1}^{n} \varrho_i(\xi_i)$ of the joint probability density (weight) function. The denominator in Eq. 3.35 is the norm squared of the multivariate orthogonal polynomial, which can be computed analytically using the product of univariate norms squared

$$\langle \Psi_j^2 \rangle = \prod_{i=1}^{n} \langle \psi_{t_i^j}^2 \rangle \tag{3.36}$$

where the univariate inner products have simple closed form expressions for each polynomial in the Askey scheme [1] and are readily computed as part of the numerically-generated solution procedures described in Section 3.1.2. Thus, the primary computational effort resides in evaluating the numerator, which is evaluated numerically using sampling, quadrature, cubature, or sparse grid approaches (and this numerical approximation leads to use of the term "pseudo-spectral" by some investigators).

[1] If joint distributions are known, then the Rosenblatt transformation is preferred.

3.6.1 Sampling

In the sampling approach, the integral evaluation is equivalent to computing the expectation (mean) of the response-basis function product (the numerator in Eq. 3.35) for each term in the expansion when sampling within the density of the weighting function. This approach is only valid for PCE and since sampling does not provide any particular monomial coverage guarantee, it is common to combine this coefficient estimation approach with a total-order chaos expansion.

In computational practice, coefficient estimations based on sampling benefit from first estimating the response mean (the first PCE coefficient) and then removing the mean from the expectation evaluations for all subsequent coefficients. While this has no effect for quadrature/sparse grid methods (see following two sections) and little effect for fully-resolved sampling, it does have a small but noticeable beneficial effect for under-resolved sampling.

3.6.2 Tensor product quadrature

In quadrature-based approaches, the simplest general technique for approximating multidimensional integrals, as in Eq. 3.35, is to employ a tensor product of one-dimensional quadrature rules. Since there is little benefit to the use of nested quadrature rules in the tensor-product case[2], we choose Gaussian abscissas, i.e. the zeros of polynomials that are orthogonal with respect to a density function weighting, e.g. Gauss-Hermite, Gauss-Legendre, Gauss-Laguerre, generalized Gauss-Laguerre, Gauss-Jacobi, or numerically-generated Gauss rules.

We first introduce an index $i \in \mathbb{N}_+$, $i \geq 1$. Then, for each value of i, let $\{\xi_1^i, \ldots, \xi_{m_i}^i\} \subset \Omega_i$ be a sequence of abscissas for quadrature on Ω_i. For $f \in C^0(\Omega_i)$ and $n = 1$ we introduce a sequence of one-dimensional quadrature operators

$$\mathscr{U}^i(f)(\xi) = \sum_{j=1}^{m_i} f(\xi_j^i)\, w_j^i, \tag{3.37}$$

with $m_i \in \mathbb{N}$ given. When utilizing Gaussian quadrature, Eq. 3.37 integrates exactly all polynomials of degree less than $2m_i - 1$, for each $i = 1, \ldots, n$. Given an expansion order p, the highest order coefficient evaluations (Eq. 3.35) can be assumed to involve integrands of at least polynomial order $2p$ (Ψ of order p and R modeled to order p) in each dimension such that a minimal Gaussian quadrature order of $p+1$ will be required to obtain good accuracy in these coefficients.

Now, in the multivariate case $n > 1$, for each $f \in C^0(\Omega)$ and the multi-index $\mathbf{i} = (i_1, \ldots, i_n) \in \mathbb{N}_+^n$ we define the full tensor product quadrature formulas

$$\mathcal{Q}_{\mathbf{i}}^n f(\xi) = \left(\mathscr{U}^{i_1} \otimes \cdots \otimes \mathscr{U}^{i_n}\right)(f)(\boldsymbol{\xi}) = \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_n=1}^{m_{i_n}} f\left(\xi_{j_1}^{i_1}, \ldots, \xi_{j_n}^{i_n}\right)\left(w_{j_1}^{i_1} \otimes \cdots \otimes w_{j_n}^{i_n}\right). \tag{3.38}$$

Clearly, the above product needs $\prod_{j=1}^n m_{i_j}$ function evaluations. Therefore, when the number of input random variables is small, full tensor product quadrature is a very effective numerical tool. On the other hand, approximations based on tensor product grids suffer from the *curse of dimensionality* since the number of collocation points in a tensor grid grows exponentially fast in the number of input random variables. For example, if Eq. 3.38 employs the same order for all random dimensions, $m_{i_j} = m$, then Eq. 3.38 requires m^n function evaluations.

In [47], it is demonstrated that close synchronization of expansion form with the monomial resolution of a particular numerical integration technique can result in significant performance improvements. In particular, the traditional approach of exploying a total-order PCE (Eqs. 3.21–3.22) neglects a significant portion of the monomial coverage for a tensor-product quadrature approach, and one should rather employ a tensor-product PCE

[2] Unless a refinement procedure is in use.

(Eqs. 3.23–3.24) to provide improved synchronization and more effective usage of the Gauss point evaluations. When the quadrature points are standard Gauss rules (i.e., no Clenshaw-Curtis, Gauss-Patterson, or Genz-Keister nested rules), it has been shown that tensor-product PCE and SC result in identical polynomial forms [29], completely eliminating a performance gap that exists between total-order PCE and SC [47].

3.6.3 Smolyak sparse grids

If the number of random variables is moderately large, one should rather consider sparse tensor product spaces as first proposed by Smolyak [121] and further investigated by Refs. [61, 9, 56, 143, 99, 100] that reduce dramatically the number of collocation points, while preserving a high level of accuracy.

Here we follow the notation and extend the description in Ref. [99] to describe the Smolyak *isotropic* formulas $\mathscr{A}(\mathrm{w}, n)$, where w is a level that is independent of dimension[3]. The Smolyak formulas are just linear combinations of the product formulas in Eq. 3.38 with the following key property: only products with a relatively small number of points are used. With $\mathscr{U}^0 = 0$ and for $i \geq 1$ define

$$\Delta^i = \mathscr{U}^i - \mathscr{U}^{i-1}. \tag{3.39}$$

and we set $|\mathbf{i}| = i_1 + \cdots + i_n$. Then the isotropic Smolyak quadrature formula is given by

$$\mathscr{A}(\mathrm{w}, n) = \sum_{|\mathbf{i}| \leq \mathrm{w}+n} \left(\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_n}\right). \tag{3.40}$$

This form is preferred for use in forming hierarchical interpolants as described in Sections 3.2.2 and 3.4.3. For nodal interpolants and polynomial chaos in sparse grids, the following equivalent form [135] is often more convenient since it collapses repeated index sets

$$\mathscr{A}(\mathrm{w}, n) = \sum_{\mathrm{w}+1 \leq |\mathbf{i}| \leq \mathrm{w}+n} (-1)^{\mathrm{w}+n-|\mathbf{i}|} \binom{n-1}{\mathrm{w}+n-|\mathbf{i}|} \cdot \left(\mathscr{U}^{i_1} \otimes \cdots \otimes \mathscr{U}^{i_n}\right). \tag{3.41}$$

For each index set $\mathbf{i}$ of levels, linear or nonlinear growth rules are used to define the corresponding one-dimensional quadrature orders. The following growth rules are employed for indices $i \geq 1$, where closed and open refer to the inclusion and exclusion of the bounds within an interval, respectively:

$$\text{closed nonlinear}: \; m = \begin{cases} 1 & i = 1 \\ 2^{i-1} + 1 & i > 1 \end{cases} \tag{3.42}$$

$$\text{open nonlinear}: \; m = 2^i - 1 \tag{3.43}$$

$$\text{open linear}: \; m = 2i - 1 \tag{3.44}$$

Nonlinear growth rules are used for fully nested rules (e.g., Clenshaw-Curtis is closed fully nested and Gauss-Patterson is open fully nested), and linear growth rules are best for standard Gauss rules that take advantage of, at most, "weak" nesting (e.g., reuse of the center point).

Examples of isotropic sparse grids, constructed from the fully nested Clenshaw-Curtis abscissas and the weakly-nested Gaussian abscissas are shown in Figure 3.1, where $\Omega = [-1, 1]^2$ and both Clenshaw-Curtis and Gauss-Legendre employ nonlinear growth[4] from Eqs. 3.42 and 3.43, respectively. There, we consider a two-dimensional parameter space and a maximum level $\mathrm{w} = 5$ (sparse grid $\mathscr{A}(5, 2)$). To see the reduction in function evaluations with respect to full tensor product grids, we also include a plot of the corresponding Clenshaw-Curtis isotropic full tensor grid having the same maximum number of points in each direction, namely $2^{\mathrm{w}} + 1 = 33$.

[3] Other common formulations use a dimension-dependent level q where $q \geq n$. We use $w = q - n$, where $w \geq 0$ for all n.

[4] We prefer linear growth for Gauss-Legendre, but employ nonlinear growth here for purposes of comparison.

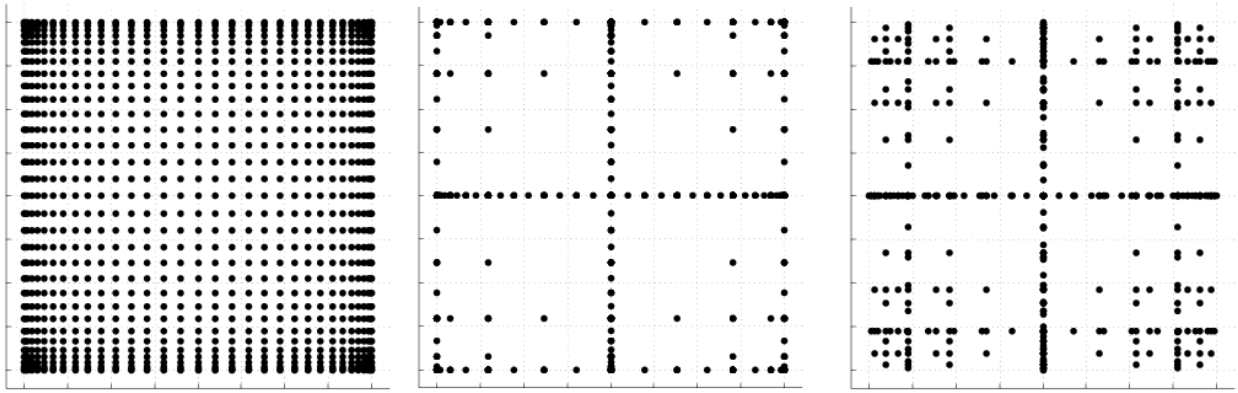


Figure 3.1: Two-dimensional grid comparison with a tensor product grid using Clenshaw-Curtis points (left) and sparse grids $\mathscr{A}(5,2)$ utilizing Clenshaw-Curtis (middle) and Gauss-Legendre (right) points with nonlinear growth.

In [47], it is demonstrated that the synchronization of total-order PCE with the monomial resolution of a sparse grid is imperfect, and that sparse grid SC consistently outperforms sparse grid PCE when employing the sparse grid to directly evaluate the integrals in Eq. 3.35. In our Dakota implementation, we depart from the use of sparse integration of total-order expansions, and instead employ a linear combination of tensor expansions [27]. That is, we compute separate tensor polynomial chaos expansions for each of the underlying tensor quadrature grids (for which there is no synchronization issue) and then sum them using the Smolyak combinatorial coefficient (from Eq. 3.41 in the isotropic case). This improves accuracy, preserves the PCE/SC consistency property described in Section 3.6.2, and also simplifies PCE for the case of anisotropic sparse grids described next.

For anisotropic Smolyak sparse grids, a dimension preference vector is used to emphasize important stochastic dimensions. Given a mechanism for defining anisotropy, we can extend the definition of the sparse grid from that of Eq. 3.41 to weight the contributions of different index set components. First, the sparse grid index set constraint becomes

$$\mathrm{w}\underline{\gamma} < \mathbf{i} \cdot \gamma \leq \mathrm{w}\underline{\gamma} + |\gamma| \tag{3.45}$$

where $\underline{\gamma}$ is the minimum of the dimension weights γ_k, k = 1 to n. The dimension weighting vector γ amplifies the contribution of a particular dimension index within the constraint, and is therefore inversely related to the dimension preference (higher weighting produces lower index set levels). For the isotropic case of all $\gamma_k = 1$, it is evident that you reproduce the isotropic index constraint $\mathrm{w}+1 \leq |\mathbf{i}| \leq \mathrm{w}+n$ (note the change from $<$ to $\leq$). Second, the combinatorial coefficient for adding the contribution from each of these index sets is modified as described in [17].

3.6.4 Cubature

Cubature rules [124, 142] are specifically optimized for multidimensional integration and are distinct from tensor-products and sparse grids in that they are not based on combinations of one-dimensional Gauss quadrature rules. They have the advantage of improved scalability to large numbers of random variables, but are restricted in integrand order and require homogeneous random variable sets (achieved via transformation). For example, optimal rules for integrands of 2, 3, and 5 and either Gaussian or uniform densities allow low-order polynomial chaos expansions ($p = 1$ or 2) that are useful for global sensitivity analysis including main effects and, for $p = 2$, all two-way interactions.

3.7 Linear regression

Regression-based PCE approaches solve the linear system:

$$\boldsymbol{\Psi}\boldsymbol{\alpha} = \boldsymbol{R} \tag{3.46}$$

for a set of PCE coefficients $\boldsymbol{\alpha}$ that best reproduce a set of response values $\boldsymbol{R}$. The set of response values can be defined on an unstructured grid obtained from sampling within the density function of $\boldsymbol{\xi}$ (point collocation [134, 76]) or on a structured grid defined from uniform random sampling on the multi-index[5] of a tensor-product quadrature grid (probabilistic collocation [126]), where the quadrature is of sufficient order to avoid sampling at roots of the basis polynomials[6]. In either case, each row of the matrix $\boldsymbol{\Psi}$ contains the N_t multivariate polynomial terms Ψ_j evaluated at a particular $\boldsymbol{\xi}$ sample. It is common to combine this coefficient estimation approach with a total-order chaos expansion in order to keep sampling requirements low. In this case, simulation requirements scale as $\frac{r(n+p)!}{n!p!}$ (r is a collocation ratio with typical values $0.1 \leq r \leq 2$). Additional regression equations can be obtained through the use of derivative information (gradients and Hessians) from each collocation point (refer to `use_derivatives` in the PCE regression specification details in the Dakota Reference Manual [2]), which can aid in scaling with respect to the number of random variables, particularly for adjoint-based derivative approaches.

Various methods can be employed to solve (3.46). The relative accuracy of each method is problem dependent. Traditionally, the most frequently used method has been least squares regression. However when $\boldsymbol{\Psi}$ is under-determined, minimizing the residual with respect to the ℓ_2 norm typically produces poor solutions. Compressed sensing methods have been successfully used to address this limitation [13, 40]. Such methods attempt to only identify the elements of the coefficient vector $\boldsymbol{\alpha}$ with the largest magnitude and enforce as many elements as possible to be zero. Such solutions are often called sparse solutions. Dakota provides algorithms that solve the following formulations:

- Basis Pursuit (BP) [22]

  $$\boldsymbol{\alpha} = \arg\min \|\boldsymbol{\alpha}\|_{\ell_1} \quad \text{such that} \quad \boldsymbol{\Psi}\boldsymbol{\alpha} = \boldsymbol{R} \tag{3.47}$$

  The BP solution is obtained in Dakota, by transforming (3.47) to a linear program which is then solved using the primal-dual interior-point method [15, 22].

- Basis Pursuit DeNoising (BPDN) [22].

  $$\boldsymbol{\alpha} = \arg\min \|\boldsymbol{\alpha}\|_{\ell_1} \quad \text{such that} \quad \|\boldsymbol{\Psi}\boldsymbol{\alpha} - \boldsymbol{R}\|_{\ell_2} \leq \varepsilon \tag{3.48}$$

  The BPDN solution is computed in Dakota by transforming (3.48) to a quadratic cone problem which is solved using the log-barrier Newton method [15, 22].

  When the matrix $\boldsymbol{\Psi}$ is not over-determined the BP and BPDN solvers used in Dakota will not return a solution. In such situations these methods simply return the least squares solution.

- Orthogonal Matching Pursuit (OMP) [34],

  $$\boldsymbol{\alpha} = \arg\min \|\boldsymbol{\alpha}\|_{\ell_0} \quad \text{such that} \quad \|\boldsymbol{\Psi}\boldsymbol{\alpha} - \boldsymbol{R}\|_{\ell_2} \leq \varepsilon \tag{3.49}$$

  OMP is a heuristic method which greedily finds an approximation to (3.49). In contrast to the aforementioned techniques for solving BP and BPDN, which minimize an objective function, OMP constructs a sparse solution by iteratively building up an approximation of the solution vector $\boldsymbol{\alpha}$. The vector is approximated as a linear combination of a subset of active columns of $\boldsymbol{\Psi}$. The active set of columns is built column by column, in a greedy fashion, such that at each iteration the inactive column with the highest correlation (inner product) with the current residual is added.

[5] Due to the discrete nature of index sampling, we enforce unique index samples by sorting and resampling as required.
[6] Generally speaking, dimension quadrature order m_i greater than dimension expansion order p_i.

- Least Angle Regression (LARS) [44] and Least Absolute Shrinkage and Selection Operator (LASSO) [127]

$$\boldsymbol{\alpha} = \arg\min \|\boldsymbol{\Psi}\boldsymbol{\alpha} - \boldsymbol{R}\|_{\ell_2}^2 \quad \text{such that} \|\boldsymbol{\alpha}\|_{\ell_1} \leq \tau \tag{3.50}$$

  A greedy solution can be found to (3.50) using the LARS algorithm. Alternatively, with only a small modification, one can provide a rigorous solution to this global optimization problem, which we refer to as the LASSO solution. Such an approach is identical to the homotopy algorithm of Osborne et al [103]. It is interesting to note that Efron [44] experimentally observed that the basic, faster LARS procedure is often identical to the LASSO solution.

  The LARS algorithm is similar to OMP. LARS again maintains an active set of columns and again builds this set by adding the column with the largest correlation with the residual to the current residual. However, unlike OMP, LARS solves a penalized least squares problem at each step taking a step along an equiangular direction, that is, a direction having equal angles with the vectors in the active set. LARS and OMP do not allow a column (PCE basis) to leave the active set. However if this restriction is removed from LARS (it cannot be from OMP) the resulting algorithm can provably solve (3.50) and generates the LASSO solution.

- Elastic net [145]

$$\boldsymbol{\alpha} = \arg\min \|\boldsymbol{\Psi}\boldsymbol{\alpha} - \boldsymbol{R}\|_{\ell_2}^2 \quad \text{such that} \quad (1-\lambda)\|\boldsymbol{\alpha}\|_{\ell_1} + \lambda\|\boldsymbol{\alpha}\|_{\ell_2}^2 \leq \tau \tag{3.51}$$

  The elastic net was developed to overcome some of the limitations of the LASSO formulation. Specifically: if the $(M \times N)$ Vandermonde matrix $\boldsymbol{\Psi}$ is over-determined $(M > N)$, the LASSO selects at most N variables before it saturates, because of the nature of the convex optimization problem; if there is a group of variables among which the pairwise correlations are very high, then the LASSO tends to select only one variable from the group and does not care which one is selected; and finally if there are high correlations between predictors, it has been empirically observed that the prediction performance of the LASSO is dominated by ridge regression [127]. Here we note that it is hard to estimate the λ penalty in practice and the aforementioned issues typically do not arise very often when solving (3.46). The elastic net formulation can be solved with a minor modification of the LARS algorithm.

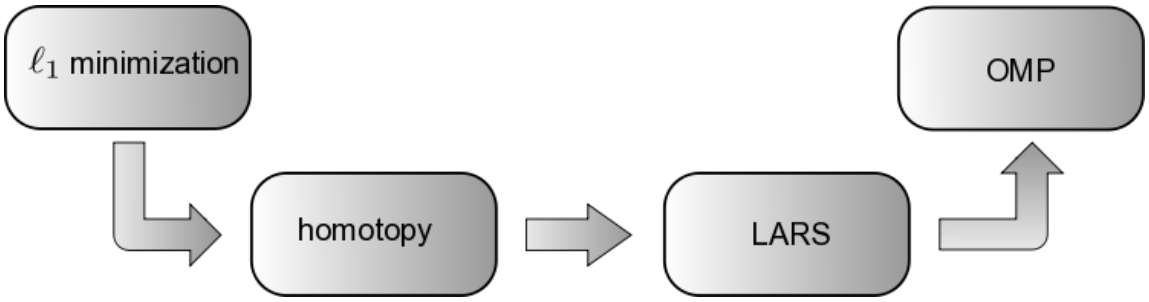


Figure 3.2: Bridging provably convergent ℓ_1 minimization algorithms and greedy algorithms such as OMP. (1) Homotopy provably solves ℓ_1 minimization problems [44]. (2) LARS is obtained from homotopy by removing the sign constraint check. (3) OMP and LARS are similar in structure, the only difference being that OMP solves a least-squares problem at each iteration, whereas LARS solves a linearly penalized least-squares problem. Figure and caption based upon Figure 1 in [39].

OMP and LARS add a PCE basis one step at a time. If $\boldsymbol{\alpha}$ contains only k non-zero terms then these methods will only take k-steps. The homotopy version of LARS also adds only basis at each step, however it can also remove bases, and thus can take more than k steps. For some problems, the LARS and homotopy solutions will coincide. Each step of these algorithm provides a possible estimation of the PCE coefficients. However, without knowledge

of the target function, there is no easy way to estimate which coefficient vector is best. With some additional computational effort (which will likely be minor to the cost of obtaining model simulations), cross validation can be used to choose an appropriate coefficient vector.

3.7.1 Cross validation

Cross validation can be used to find a coefficient vector $\boldsymbol{\alpha}$ that approximately minimizes $\|\hat{f}(\mathbf{x}) - f(\mathbf{x})\|_{L^2(\rho)}$, where f is the target function and $\hat{f}$ is the PCE approximation using $\boldsymbol{\alpha}$. Given training data $\mathbf{X}$ and a set of algorithm parameters $\boldsymbol{\beta}$ (which can be step number in an algorithm such as OMP, or PCE maximum degree), K-folds cross validation divides $\mathbf{X}$ into K sets (folds) $\mathbf{X}_k$, $k = 1, \ldots, K$ of equal size. A PCE $\hat{f}_{\boldsymbol{\beta}}^{-k}(\mathbf{X})$, is built on the training data $\mathbf{X}_{\text{tr}} = \mathbf{X} \setminus \mathbf{X}_k$ with the k-th fold removed, using the tuning parameters $\boldsymbol{\beta}$. The remaining data $\mathbf{X}_k$ is then used to estimate the prediction error. The prediction error is typically approximated by $e(\hat{f}) = \|\hat{f}(\mathbf{x}) - f(\mathbf{x})\|_{\ell_2}$, $\mathbf{x} \in \mathbf{X}_k$ [71]. This process is then repeated K times, removing a different fold from the training set each time.

The cross validation error is taken to be the average of the prediction errors for the K-experiments

$$CV(\hat{f}_{\boldsymbol{\beta}}) = \mathrm{E}[e(\hat{f}_{\boldsymbol{\beta}}^{-k})] = \frac{1}{K}\sum_{k=1}^{K} e(\hat{f}_{\boldsymbol{\beta}}^{-k})$$

We minimize $CV(\hat{f}_{\boldsymbol{\beta}})$ as a surrogate for minimizing $\|\hat{f}_{\boldsymbol{\beta}}(\mathbf{x}) - f(\mathbf{x})\|_{L^2(\rho)}$ and choose the tuning parameters

$$\boldsymbol{\beta}^{\star} = \arg\min CV(\hat{f}_{\boldsymbol{\beta}})\mathrm{Var}[e(\hat{f}_{\boldsymbol{\beta}}^{-k})] \tag{3.52}$$

to construct the final "best" PCE approximation of f that the training data can produce.

3.7.2 Iterative basis selection

When the coefficients of a PCE can be well approximated by a sparse vector, ℓ_1-minimization is extremely effective at recovering the coefficients of that PCE. It is possible, however, to further increase the efficacy of ℓ_1-minimization by leveraging realistic models of structural dependencies between the values and locations of the PCE coefficients. For example [8, 42, 87] have successfully increased the performance of ℓ_1-minimization when recovering wavelet coefficients that exhibit a tree-like structure. In this vein, we propose an algorithm for identifying the large coefficients of PC expansions that form a semi-connected subtree of the PCE coefficient tree.

The coefficients of polynomial chaos expansions often form a multi-dimensional tree. Given an ancestor basis term $\phi_{\boldsymbol{\lambda}}$ of degree $\|\boldsymbol{\lambda}\|_1$ we define the indices of its children as $\boldsymbol{\lambda}+\mathbf{e}_k$, $k = 1, \ldots, d$, where $\mathbf{e}_k = (0, \ldots, 1, \ldots, 0)$ is the unit vector co-directional with the k-th dimension. An example of a typical PCE tree is depicted in Figure 3.3. In this figure, as often in practice, the magnitude of the ancestors of a PCE coefficient is a reasonable indicator of the size of the child coefficient. In practice, some branches (connections) between levels of the tree may be missing. We refer to trees with missing branches as semi-connected trees.

In the following we present a method for estimating PCE coefficients that leverages the tree structure of PCE coefficients to increase the accuracy of coefficient estimates obtained by ℓ_1-minimization.

Typically ℓ_1-minimization is applied to an a priori chosen and fixed basis set Λ. However the accuracy of coefficients obtained by ℓ_1-minimization can be increased by adaptively selecting the PCE basis.

To select a basis for ℓ_1-minimization we employ a four step iterative procedure involving restriction, expansion, identification and selection. The iterative basis selection procedure is outlined in Algorithm 1. A graphical version

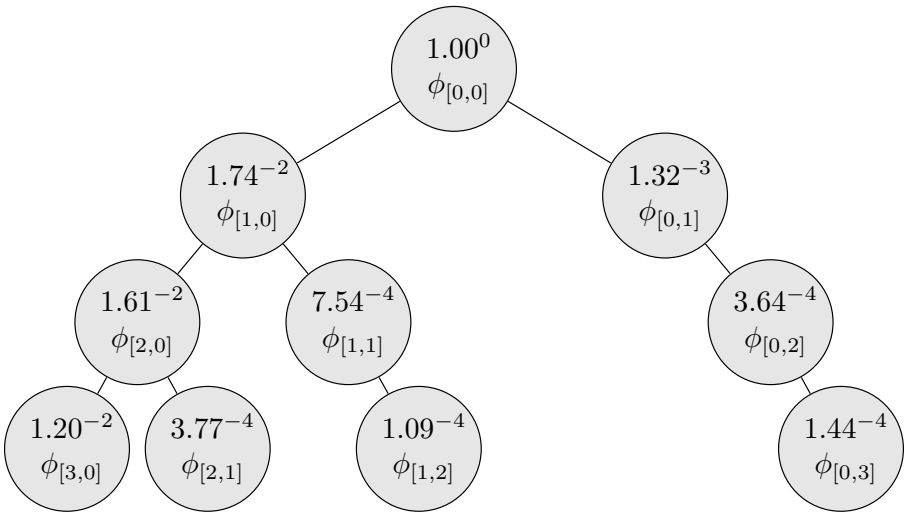


Figure 3.3: Tree structure of the coefficients of a two dimensional PCE with a total-degree basis of order 3. For clarity we only depict one connection per node, but in d dimensions a node of a given degree p will be a child of up to d nodes of degree $p-1$. For example, not only is the basis $\boldsymbol{\phi}_{[1,1]}$ a child of $\boldsymbol{\phi}_{[1,0]}$ (as depicted) but it is also a child of $\boldsymbol{\phi}_{[0,1]}$

of the algorithm is also presented in Figure 3.4. The latter emphasizes the four stages of basis selection, that is restriction, growth, identification and selection. These four stages are also highlighted in Algorithm 1 using the corresponding colors in Figure 3.4.

To initiate the basis selection algorithm, we first define a basis set $\Lambda^{(0)}$ and use ℓ_1-minimization to identify the largest coefficients $\boldsymbol{\alpha}^{(0)}$. The choice of $\Lambda^{(0)}$ can sometimes affect the performance of the basis selection algorithm. We found a good choice to be $\Lambda^{(0)} = \Lambda_{p,1}$, where p is the degree that gives $|\Lambda^d_{p,1}|$ closest to $10M$, i.e. $\Lambda^d_{p,1} = \arg\min_{\Lambda^d_{p,1}\in\{\Lambda^d_{1,1},\Lambda^d_{2,1},\ldots\}} \left||\Lambda^d_{p,1}| - 10M\right|$. Given a basis $\Lambda^{(k)}$ and corresponding coefficients $\boldsymbol{\alpha}^{(k)}$ we reduce the basis to a set $\Lambda^{(k)}_{\varepsilon}$ containing only the terms with non-zero coefficients. This restricted basis is then expanded T times using an algorithm which we will describe in Section 3.7.2.1. ℓ_1-minimization is then applied to each of the expanded basis sets $\Lambda^{(k,t)}$ for $t = 1, \ldots, T$. Each time ℓ_1-minimization is used, we employ cross validation to choose ε. Therefore, at every basis set considered during the evolution of the algorithm we have a measure of the expected accuracy of the PCE coefficients. At each step in the algorithm we choose the basis set

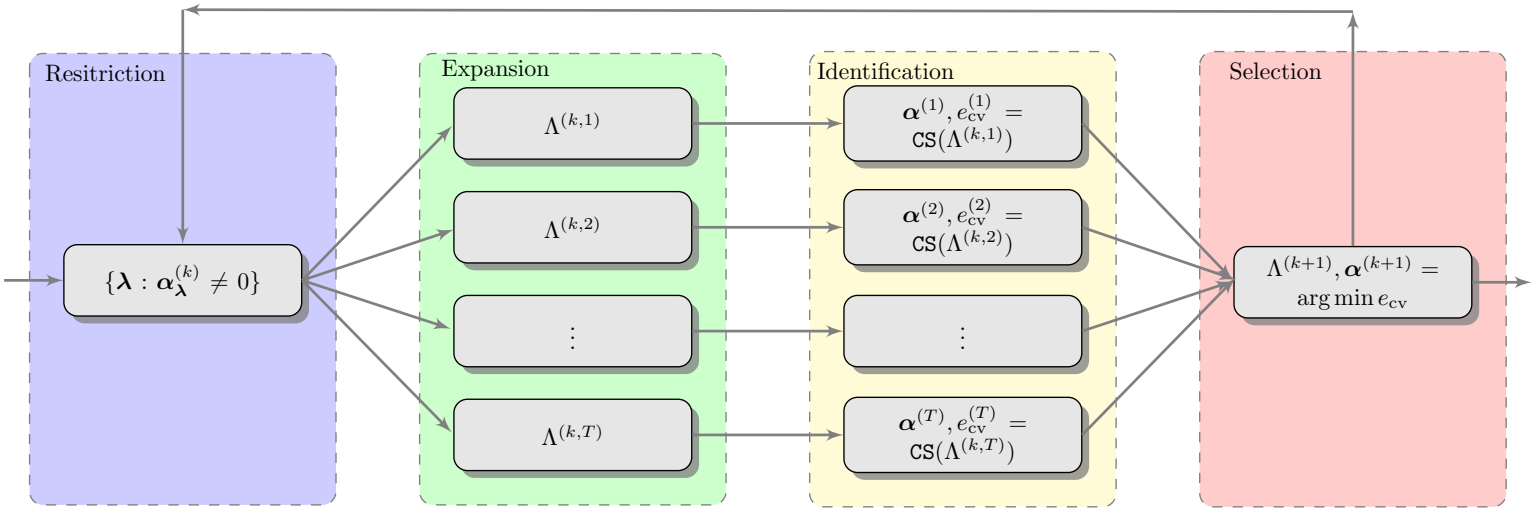


Figure 3.4: Graphical depiction of the basis adaptation algorithm.

that results in the lowest cross validation error.

Algorithm 1: $\Lambda^\star$,$\boldsymbol{\alpha}^\star$=BASIS_SELECTION[$\boldsymbol{\phi}$,$\mathbf{f}$,$\varepsilon$]

$\Lambda^\star = \Lambda^{(0)} = \Lambda_{p,1}^d = \arg\min_{\Lambda_{p,1}^d \in \{\Lambda_{1,1}^d, \Lambda_{2,1}^d, \ldots\}} \left| |\Lambda_{p,1}^d| - 10M \right|$
$\boldsymbol{\alpha}^{(0)}, e_{\text{cv}}^{(0)}$ = ℓ_1-minimization[$\boldsymbol{\phi}(\Lambda^{(0)})$,$\mathbf{f}$]
$T = 3$, $e_{\text{cv}}^\star = \infty$, $k = 1$
while *TRUE* **do**
  $e_{\text{cv}}^{(k)} = \infty$
  $\Lambda^{(k,0)} = \{\boldsymbol{\lambda} : \boldsymbol{\lambda} \in \Lambda^{(k-1)}, \boldsymbol{\alpha}_{\boldsymbol{\lambda}}^{(k)} \neq 0\}$
  for $t \in \{1, \ldots, T\}$ **do**
    $\Lambda^{(k,t)}$ = EXPAND[$\Lambda^{(k,t-1)}$]
    $\boldsymbol{\alpha}^{(k,t)}, e_{\text{cv}}^{(k,t)}$ = ℓ_1-minimization[$\boldsymbol{\phi}(\Lambda^{(k,t)})$,$\mathbf{f}$]
    if $e_{\text{cv}}^{(k,t)} < e_{\text{cv}}^{(k)}$ **then**
      $e_{\text{cv}}^{(k)} = e_{\text{cv}}^{(k,t)}$, $\boldsymbol{\alpha}^{(k)} = \boldsymbol{\alpha}^{(k,t)}$, $\Lambda^{(k)} = \Lambda^{(k,t)}$
    end
  end
  if $e_{\text{cv}}^{(k)} > e_{\text{cv}}^\star$ **then**
    TERMINATE
  end
  $\alpha^\star = \boldsymbol{\alpha}^{(k)}$, $\Lambda^\star = \Lambda^{(k)}$, $e_{\text{cv}}^\star = e_{\text{cv}}^{(k)}$
end

3.7.2.1 Basis expansion

Define $\{\boldsymbol{\lambda} + \mathbf{e}_j : 1 \leq j \leq d\}$ the forward neighborhood of an index $\boldsymbol{\lambda}$ and similarly let $\{\boldsymbol{\lambda} - \mathbf{e}_j : 1 \leq j \leq d\}$ denote the backward neighborhood. To expand a basis set Λ we must first find the forward neighbors $\mathcal{F} =$

$\{\boldsymbol{\lambda} + \mathbf{e}_j : \boldsymbol{\lambda} \in \Lambda, 1 \leq j \leq d\}$ of all indices $\boldsymbol{\lambda} \in \Lambda$. The expanded basis is then given by

$$\Lambda^+ = \Lambda \cup \mathcal{A}, \quad \mathcal{A} = \{\boldsymbol{\lambda} : \boldsymbol{\lambda} \in \mathcal{F}, \boldsymbol{\lambda} - \mathbf{e}_n \in \Lambda \text{ for } 1 \leq n \leq d,\ \lambda_k > 1\}$$

where we have used the following admissibility criteria

$$\boldsymbol{\lambda} - \mathbf{e}_n \in \Lambda \text{ for } 1 \leq n \leq d,\ \lambda_k > 1 \tag{3.53}$$

to target PCE basis indices that are likely to have large PCE coefficients. A forward neighbor is admissible only if its backward neighbors exist in all dimensions. If the backward neighbors do not exist then ℓ_1-minimization has previously identified that the coefficients of these backward neighbors are negligible.

The admissibility criterion is explained graphically in Figure 3.5. In the left graphic, both children of the current index are admissible, because its backwards neighbors exist in every dimension. In the right graphic only the child in the vertical dimension is admissible, as not all parents of the horizontal child exist.

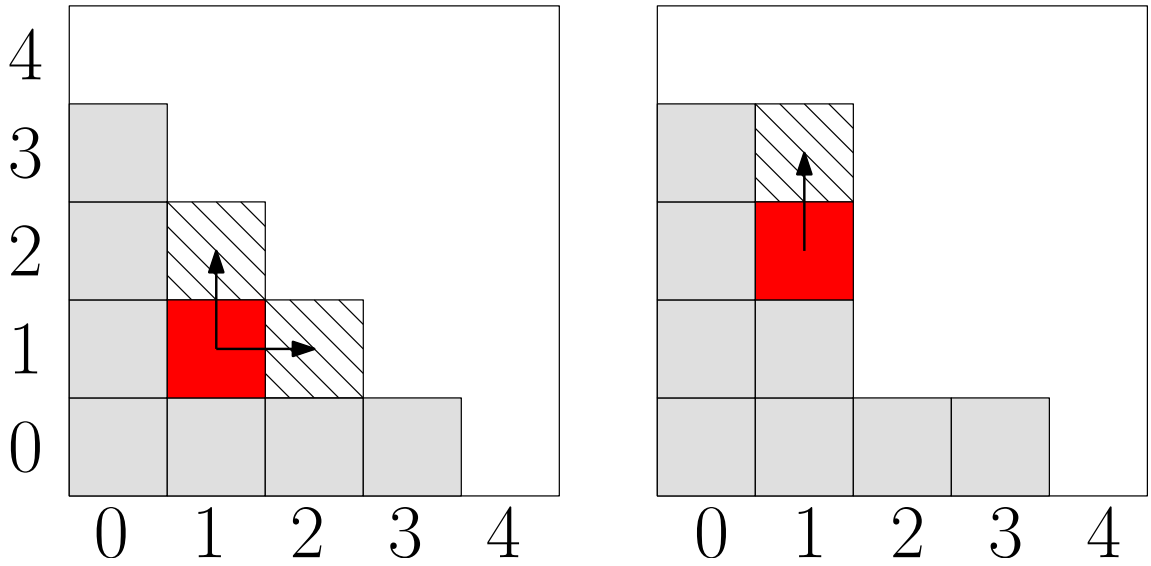


Figure 3.5: Identification of the admissible indices of an index (red). The indices of the current basis Λ are gray and admissible indices are striped. A index is admissible only if its backwards neighbors exists in every dimension.

At the k-th iteration of Algorithm 1, ℓ_1-minimization is applied to $\Lambda^{(k-1)}$ and used to identify the significant coefficients of the PCE and their corresponding basis terms $\Lambda^{(k,0)}$. The set of non-zero coefficients $\Lambda^{(k,0)}$ identified by ℓ_1-minimization is then expanded. The `EXPAND` routine expands an index set by one polynomial degree, but sometimes it may be necessary to expand the basis $\Lambda^{(k)}$ more than once.[7] To generate these higher degree index sets `EXPAND` is applied recursively to $\Lambda^{(k,0)}$ up to a fixed number of T times. Specifically, the following sets are generated

$$\Lambda^{(k,t)} = \Lambda^{(k,t-1)} \cup \{\boldsymbol{\lambda} : \boldsymbol{\lambda} - \mathbf{e}_n \in \Lambda^{(k,t-1)}, 1 \leq n \leq d,\ \lambda_n > 1\}.$$

[7] The choice of $T > 1$ enables the basis selection algorithm to be applied to semi-connected tree structures as well as fully connected trees. Setting $T > 1$ allows us to prevent premature termination of the algorithm if most of the coefficients of the children of the current set $\Lambda^{(k)}$ are small but the coefficients of the children's children are not.

As the number of expansion steps T increases the number of terms in the expanded basis increases rapidly and degradation in the performance of ℓ_1-minimization can result (this is similar to what happens when increasing the degree of a total degree basis). To avoid degradation of the solution, we use cross validation to choose the number of inner expansion steps $t \in [1, T]$.

3.7.3 Orthogonal Least Interpolation

Orthogonal least interpolation (OLI) [95] enables the construction of interpolation polynomials based on arbitrarily located grids in arbitrary dimensions. The interpolation polynomials can be constructed using using orthogonal polynomials corresponding to the probability distribution function of the uncertain random variables.

The algorithm for constructing an OLI is split into three stages: (i) basis determination - transform the orthogonal basis into a polynomial space that is amenable for interpolation; (ii) coefficient determination - determine the interpolatory coefficients on the transformed basis elements; (iii) connection problem - translate the coefficients on the transformed basis to coefficients in the original orthogonal basis These three steps can be achieved by a sequence of LU and QR factorizations.

Orthogonal least interpolation is closesly related to the aforementioned regression methods, in that OLI can be used to build approximations of simulation models when computing structured simulation data, such as sparse grids or cubature nodes, is infeasiable. The interpolants produced by OLI have two additional important properties. Firstly, the orthogonal least interpolant is the lowest-degree polynomial that interpolates the data. Secondly, the least orthogonal interpolation space is monotonic. This second property means that the least interpolant can be extended to new data without the need to completely reconstructing the interpolant. The transformed interpolation basis can simply be extended to include the new necessary basis functions.

3.8 Analytic moments

Mean and covariance of polynomial chaos expansions are available in simple closed form:

$$\mu_i = \langle R_i \rangle \cong \sum_{k=0}^{P} \alpha_{ik} \langle \Psi_k(\boldsymbol{\xi}) \rangle = \alpha_{i0} \tag{3.54}$$

$$\Sigma_{ij} = \langle (R_i - \mu_i)(R_j - \mu_j) \rangle \cong \sum_{k=1}^{P} \sum_{l=1}^{P} \alpha_{ik} \alpha_{jl} \langle \Psi_k(\boldsymbol{\xi}) \Psi_l(\boldsymbol{\xi}) \rangle = \sum_{k=1}^{P} \alpha_{ik} \alpha_{jk} \langle \Psi_k^2 \rangle \tag{3.55}$$

where the norm squared of each multivariate polynomial is computed from Eq. 3.36. These expressions provide exact moments of the expansions, which converge under refinement to moments of the true response functions.

Similar expressions can be derived for stochastic collocation:

$$\mu_i = \langle R_i \rangle \cong \sum_{k=1}^{N_p} r_{ik} \langle \boldsymbol{L}_k(\boldsymbol{\xi}) \rangle = \sum_{k=1}^{N_p} r_{ik} w_k \tag{3.56}$$

$$\Sigma_{ij} = \langle R_i R_j \rangle - \mu_i \mu_j \cong \sum_{k=1}^{N_p} \sum_{l=1}^{N_p} r_{ik} r_{jl} \langle \boldsymbol{L}_k(\boldsymbol{\xi}) \boldsymbol{L}_l(\boldsymbol{\xi}) \rangle - \mu_i \mu_j = \sum_{k=1}^{N_p} r_{ik} r_{jk} w_k - \mu_i \mu_j \tag{3.57}$$

where we have simplified the expectation of Lagrange polynomials constructed at Gauss points and then integrated at these same Gauss points. For tensor grids and sparse grids with fully nested rules, these expectations leave only the weight corresponding to the point for which the interpolation value is one, such that the final equalities in

Eqs. 3.56–3.57 hold precisely. For sparse grids with non-nested rules, however, interpolation error exists at the collocation points, such that these final equalities hold only approximately. In this case, we have the choice of computing the moments based on sparse numerical integration or based on the moments of the (imperfect) sparse interpolant, where small differences may exist prior to numerical convergence. In Dakota, we employ the former approach; i.e., the right-most expressions in Eqs. 3.56–3.57 are employed for all tensor and sparse cases irregardless of nesting. Skewness and kurtosis calculations as well as sensitivity derivations in the following sections are also based on this choice. The expressions for skewness and (excess) kurtosis from direct numerical integration of the response function are as follows:

$$\gamma_{1_i} = \left\langle \left(\frac{R_i - \mu_i}{\sigma_i} \right)^3 \right\rangle \cong \frac{1}{\sigma_i^3} \left[\sum_{k=1}^{N_p} (r_{ik} - \mu_i)^3 w_k \right] \tag{3.58}$$

$$\gamma_{2_i} = \left\langle \left(\frac{R_i - \mu_i}{\sigma_i} \right)^4 \right\rangle - 3 \cong \frac{1}{\sigma_i^4} \left[\sum_{k=1}^{N_p} (r_{ik} - \mu_i)^4 w_k \right] - 3 \tag{3.59}$$

3.9 Local sensitivity analysis: derivatives with respect to expansion variables

Polynomial chaos expansions are easily differentiated with respect to the random variables [113]. First, using Eq. 3.20,

$$\frac{dR}{d\xi_i} = \sum_{j=0}^{P} \alpha_j \frac{d\Psi_j}{d\xi_i}(\boldsymbol{\xi}) \tag{3.60}$$

and then using Eq. 3.19,

$$\frac{d\Psi_j}{d\xi_i}(\boldsymbol{\xi}) = \frac{d\psi_{t_i^j}}{d\xi_i}(\xi_i) \prod_{\substack{k=1 \\ k \neq i}}^{n} \psi_{t_k^j}(\xi_k) \tag{3.61}$$

where the univariate polynomial derivatives $\frac{d\psi}{d\xi}$ have simple closed form expressions for each polynomial in the Askey scheme [1]. Finally, using the Jacobian of the (extended) Nataf variable transformation,

$$\frac{dR}{dx_i} = \frac{dR}{d\boldsymbol{\xi}} \frac{d\boldsymbol{\xi}}{dx_i} \tag{3.62}$$

which simplifies to $\frac{dR}{d\xi_i} \frac{d\xi_i}{dx_i}$ in the case of uncorrelated x_i.

Similar expressions may be derived for stochastic collocation, starting from Eq. 3.26:

$$\frac{dR}{d\xi_i} = \sum_{j=1}^{N_p} r_j \frac{d\boldsymbol{L}_j}{d\xi_i}(\boldsymbol{\xi}) \tag{3.63}$$

where the multidimensional interpolant $\boldsymbol{L}_j$ is formed over either tensor-product quadrature points or a Smolyak sparse grid. For the former case, the derivative of the multidimensional interpolant $\boldsymbol{L}_j$ involves differentiation of Eq. 3.27:

$$\frac{d\boldsymbol{L}_j}{d\xi_i}(\boldsymbol{\xi}) = \frac{dL_{c_i^j}}{d\xi_i}(\xi_i) \prod_{\substack{k=1 \\ k \neq i}}^{n} L_{c_k^j}(\xi_k) \tag{3.64}$$

and for the latter case, the derivative involves a linear combination of these product rules, as dictated by the Smolyak recursion shown in Eq. 3.41. Finally, calculation of $\frac{dR}{dx_i}$ involves the same Jacobian application shown in Eq. 3.62.

3.10 Global sensitivity analysis: variance-based decomposition

In addition to obtaining derivatives of stochastic expansions with respect to the random variables, it is possible to obtain variance-based sensitivity indices from the stochastic expansions. Variance-based sensitivity indices are explained in the Design of Experiments Chapter of the User's Manual [3]. The concepts are summarized here as well. Variance-based decomposition is a global sensitivity method that summarizes 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 index T_i. These indices are also called the Sobol' indices. The main effect sensitivity index corresponds to the fraction of the uncertainty in the output, Y, that can be attributed to input x_i alone. The total effects index corresponds to the fraction of the 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:

$$S_i = \frac{Var_{x_i}[E(Y|x_i)]}{Var(Y)} \tag{3.65}$$

and

$$T_i = \frac{E(Var(Y|x_{-i}))}{Var(Y)} = \frac{Var(Y) - Var(E[Y|x_{-i}])}{Var(Y)} \tag{3.66}$$

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 evaluation of m-dimensional integrals which are typically approximated by Monte-Carlo sampling. However, in stochastic expansion methods, it is possible to obtain the sensitivity indices as analytic functions of the coefficients in the stochastic expansion. The derivation of these results is presented in [125]. The sensitivity indices are printed as a default when running either polynomial chaos or stochastic collocation in Dakota. Note that in addition to the first-order main effects, S_i, we are able to calculate the sensitivity indices for higher order interactions such as the two-way interaction $S_{i,j}$.

3.11 Automated Refinement

Several approaches for refinement of stochastic expansions are currently supported, involving uniform or dimension-adaptive approaches to p- or h-refinement using structured (isotropic, anisotropic, or generalized) or unstructured grids. Specific combinations include:

- uniform refinement with unbiased grids
  - p-refinement: isotropic global tensor/sparse grids (PCE, SC) and regression (PCE only) with global basis polynomials
  - h-refinement: isotropic global tensor/sparse grids with local basis polynomials (SC only)
- dimension-adaptive refinement with biased grids

  - p-refinement: anisotropic global tensor/sparse grids with global basis polynomials using global sensitivity analysis (PCE, SC) or spectral decay rate estimation (PCE only)
  - h-refinement: anisotropic global tensor/sparse grids with local basis polynomials (SC only) using global sensitivity analysis

- goal-oriented dimension-adaptive refinement with greedy adaptation
  - p-refinement: generalized sparse grids with global basis polynomials (PCE, SC)
  - h-refinement: generalized sparse grids with local basis polynomials (SC only)

Each involves incrementing the global grid using differing grid refinement criteria, synchronizing the stochastic expansion specifics for the updated grid, and then updating the statistics and computing convergence criteria. Future releases will support local h-refinement approaches that can replace or augment the global grids currently supported. The sub-sections that follow enumerate each of the first level bullets above.

3.11.1 Uniform refinement with unbiased grids

Uniform refinement involves ramping the resolution of a global structured or unstructured grid in an unbiased manner and then refining an expansion to synchronize with this increased grid resolution. In the case of increasing the order of an isotropic tensor-product quadrature grid or the level of an isotropic Smolyak sparse grid, a p-refinement approach increases the order of the global basis polynomials (Sections 3.1, 3.2.1.1, and 3.2.1.2) in a synchronized manner and an h-refinement approach reduces the approximation range of fixed order local basis polynomials (Sections 3.2.1.3 and 3.2.1.4). And in the case of uniform p-refinement with PCE regression, the collocation oversampling ratio (refer to Methods specification within Dakota Reference Manual [2]) is held fixed, such that an increment in isotropic expansion order is matched with a corresponding increment in the number of structured (probabilistic collocation) or unstructured samples (point collocation) employed in the linear least squares solve.

For uniform refinement, anisotropic dimension preferences are not computed, and the only algorithmic requirements are:

- With the usage of nested integration rules with restricted exponential growth, Dakota must ensure that a change in level results in a sufficient change in the grid; otherwise premature convergence could occur within the refinement process. If no change is initially detected, Dakota continues incrementing the order/level (without grid evaluation) until the number of grid points increases.
- A convergence criterion is required. For uniform refinement, Dakota employs the L^2 norm of the change in the response covariance matrix as a general-purpose convergence metric.

3.11.2 Dimension-adaptive refinement with biased grids

Dimension-adaptive refinement involves ramping the order of a tensor-product quadrature grid or the level of a Smolyak sparse grid anisotropically, that is, using a defined dimension preference. This dimension preference may be computed from local sensitivity analysis, global sensitivity analysis, a posteriori error estimation, or decay rate estimation. In the current release, we focus on global sensitivity analysis and decay rate estimation. In the former case, dimension preference is defined from total Sobol' indices (Eq. 3.66) and is updated on every iteration of the adaptive refinement procedure, where higher variance attribution for a dimension indicates higher preference for that dimension. In the latter case, the spectral decay rates for the polynomial chaos coefficients are estimated using the available sets of univariate expansion terms (interaction terms are ignored). Given a

set of scaled univariate coefficients (scaled to correspond to a normalized basis), the decay rate can be inferred using a technique analogous to Richardson extrapolation. The dimension preference is then defined from the inverse of the rate: slowly converging dimensions need greater refinement pressure. For both of these cases, the dimension preference vector supports anisotropic sparse grids based on a linear index-set constraint (Eq. 3.45) or anisotropic tensor grids (Eq. 3.38) with dimension order scaled proportionately to preference; for both grids, dimension refinement lower bound constraints are enforced to ensure that all previously evaluated points remain in new refined grids.

Given an anisotropic global grid, the expansion refinement proceeds as for the uniform case, in that the p-refinement approach increases the order of the global basis polynomials (Sections 3.1, 3.2.1.1, and 3.2.1.2) in a synchronized manner and an h-refinement approach reduces the approximation range of fixed order local basis polynomials (Sections 3.2.1.3 and 3.2.1.4). Also, the same grid change requirements and convergence criteria described for uniform refinement (Section 3.11.1) are applied in this case.

3.11.3 Goal-oriented dimension-adaptive refinement with greedy adaptation

Relative to the uniform and dimension-adaptive refinement capabilities described previously, the generalized sparse grid algorithm [62] supports greater flexibility in the definition of sparse grid index sets and supports refinement controls based on general statistical quantities of interest (QOI). This algorithm was originally intended for adaptive numerical integration on a hypercube, but can be readily extended to the adaptive refinement of stochastic expansions using the following customizations:

- In addition to hierarchical interpolants in SC, we employ independent polynomial chaos expansions for each active and accepted index set. Pushing and popping index sets then involves increments of tensor chaos expansions (as described in Section 3.6.3) along with corresponding increments to the Smolyak combinatorial coefficients.

- Since we support bases for more than uniform distributions on a hypercube, we exploit rule nesting when possible (i.e., Gauss-Patterson for uniform or transformed uniform variables, and Genz-Keister for normal or transformed normal variables), but we do not require it. This implies a loss of some algorithmic simplifications in [62] that occur when grids are strictly hierarchical.

- In the evaluation of the effect of a trial index set, the goal in [62] is numerical integration and the metric is the size of the increment induced by the trial set on the expectation of the function of interest. It is straightforward to instead measure the effect of a trial index set on response covariance, numerical probability, or other statistical QOI computed by post-processing the resulting PCE or SC expansion. By tying the refinement process more closely to the statistical QOI, the refinement process can become more efficient in achieving the desired analysis objectives.

Hierarchical increments in a variety of statistical QoI may be derived, starting from increments in response mean and covariance. The former is defined from computing the expectation of the difference interpolants in Eqs. 3.31-3.32, and the latter is defined as:

$$\Delta\Sigma_{ij} = \Delta E[R_i R_j] - \mu_{R_i}\Delta E[R_j] - \mu_{R_j}\Delta E[R_i] - \Delta E[R_i]\Delta E[R_j] \tag{3.67}$$

Increments in standard deviation and reliability indices can subsequently be defined, where care is taken to preserve numerical precision through the square root operation (e.g., via Boost sqrt1pm1()).

Given these customizations, the algorithmic steps can be summarized as:

1. *Initialization:* Starting from an initial isotropic or anisotropic reference grid (often the $w = 0$ grid corresponding to a single collocation point), accept the reference index sets as the old set and define active index sets using the admissible forward neighbors of all old index sets.

2. *Trial set evaluation:* Evaluate the tensor grid corresponding to each trial active index set, form the tensor polynomial chaos expansion or tensor interpolant corresponding to it, update the Smolyak combinatorial coefficients, and combine the trial expansion with the reference expansion. Perform necessary bookkeeping to allow efficient restoration of previously evaluated tensor expansions.

3. *Trial set selection:* Select the trial index set that induces the largest change in the statistical QOI, normalized by the cost of evaluating the trial index set (as indicated by the number of new collocation points in the trial grid). In our implementation, the statistical QOI is defined using an L^2 norm of change in CDF/CCDF probability/reliability/response level mappings, when level mappings are present, or L^2 norm of change in response covariance, when level mappings are not present.

4. *Update sets:* If the largest change induced by the trial sets exceeds a specified convergence tolerance, then promote the selected trial set from the active set to the old set and update the active sets with new admissible forward neighbors; return to step 2 and evaluate all trial sets with respect to the new reference point. If the convergence tolerance is satisfied, advance to step 5.

5. *Finalization:* Promote all remaining active sets to the old set, update the Smolyak combinatorial coefficients, and perform a final combination of tensor expansions to arrive at the final result for the statistical QOI.

3.12 Multifidelity methods

In a multifidelity uncertainty quantification approach employing stochastic expansions, we seek to utilize a predictive low-fidelity model to our advantage in reducing the number of high-fidelity model evaluations required to compute high-fidelity statistics to a particular precision. When a low-fidelity model captures useful trends of the high-fidelity model, then the model discrepancy may have either lower complexity, lower variance, or both, requiring less computational effort to resolve its functional form than that required for the original high-fidelity model [97].

To accomplish this goal, an expansion will first be formed for the model discrepancy (the difference between response results if additive correction or the ratio of results if multiplicative correction). These discrepancy functions are the same functions approximated in surrogate-based minimization (see Surrogate Models section within the Models chapter of the User's Manual [3]). The exact discrepancy functions are

$$A(\boldsymbol{\xi}) = R_{hi}(\boldsymbol{\xi}) - R_{lo}(\boldsymbol{\xi}) \tag{3.68}$$

$$B(\boldsymbol{\xi}) = \frac{R_{hi}(\boldsymbol{\xi})}{R_{lo}(\boldsymbol{\xi})} \tag{3.69}$$

Approximating the high-fidelity response functions using approximations of these discrepancy functions then involves

$$\hat{R}_{hi_A}(\boldsymbol{\xi}) = R_{lo}(\boldsymbol{\xi}) + \hat{A}(\boldsymbol{\xi}) \tag{3.70}$$

$$\hat{R}_{hi_B}(\boldsymbol{\xi}) = R_{lo}(\boldsymbol{\xi})\hat{B}(\boldsymbol{\xi}) \tag{3.71}$$

where $\hat{A}(\boldsymbol{\xi})$ and $\hat{B}(\boldsymbol{\xi})$ are stochastic expansion approximations to the exact correction functions:

$$\hat{A}(\boldsymbol{\xi}) = \sum_{j=0}^{P_{hi}} \alpha_j \Psi_j(\boldsymbol{\xi}) \quad \text{or} \quad \sum_{j=1}^{N_{hi}} a_j \boldsymbol{L}_j(\boldsymbol{\xi}) \tag{3.72}$$

$$\hat{B}(\boldsymbol{\xi}) = \sum_{j=0}^{P_{hi}} \beta_j \Psi_j(\boldsymbol{\xi}) \quad \text{or} \quad \sum_{j=1}^{N_{hi}} b_j \boldsymbol{L}_j(\boldsymbol{\xi}) \tag{3.73}$$

where α_j and β_j are the spectral coefficients for a polynomial chaos expansion (evaluated via Eq. 3.35, for example) and a_j and b_j are the interpolation coefficients for stochastic collocation (values of the exact discrepancy evaluated at the collocation points).

Second, an expansion will be formed for the low fidelity surrogate model, where the intent is for the level of resolution to be higher than that required to resolve the discrepancy ($P_{lo} \gg P_{hi}$ or $N_{lo} \gg N_{hi}$; either enforced statically through order/level selections or automatically through adaptive refinement):

$$R_{lo}(\boldsymbol{\xi}) \cong \sum_{j=0}^{P_{lo}} \gamma_j \Psi_j(\boldsymbol{\xi}) \quad \text{or} \quad \sum_{j=1}^{N_{lo}} r_{lo_j} \boldsymbol{L}_j(\boldsymbol{\xi}) \tag{3.74}$$

Then the two expansions are combined (added or multiplied) into a new expansion that approximates the high fidelity model, from which the final set of statistics are generated. For polynomial chaos expansions, this combination entails:

- in the additive case, the high-fidelity expansion is formed by simply overlaying the expansion forms and adding the spectral coefficients that correspond to the same basis polynomials.
- in the multiplicative case, the form of the high-fidelity expansion must first be defined to include all polynomial orders indicated by the products of each of the basis polynomials in the low fidelity and discrepancy expansions (most easily estimated from total-order, tensor, or sum of tensor expansions which involve simple order additions). Then the coefficients of this product expansion are computed as follows (shown generically for $z = xy$ where x, y, and z are each expansions of arbitrary form):

$$\sum_{k=0}^{P_z} z_k \Psi_k(\boldsymbol{\xi}) = \sum_{i=0}^{P_x} \sum_{j=0}^{P_y} x_i y_j \Psi_i(\boldsymbol{\xi}) \Psi_j(\boldsymbol{\xi}) \tag{3.75}$$

$$z_k = \frac{\sum_{i=0}^{P_x} \sum_{j=0}^{P_y} x_i y_j \langle \Psi_i \Psi_j \Psi_k \rangle}{\langle \Psi_k^2 \rangle} \tag{3.76}$$

  where tensors of one-dimensional basis triple products $\langle \psi_i \psi_j \psi_k \rangle$ are typically sparse and can be efficiently precomputed using one dimensional quadrature for fast lookup within the multidimensional triple products.

For stochastic collocation, the high-fidelity expansion generated from combining the low fidelity and discrepancy expansions retains the polynomial form of the low fidelity expansion, for which only the coefficients are updated in order to interpolate the sum or product values (and potentially their derivatives). Since we will typically need to produce values of a less resolved discrepancy expansion on a more resolved low fidelity grid to perform this combination, we utilize the discrepancy expansion rather than the original discrepancy function values for both interpolated and non-interpolated point values (and derivatives), in order to ensure consistency.

Chapter 4

Epistemic Methods

This chapter covers theoretical aspects of methods for propagating epistemic uncertainty.

4.1 Dempster-Shafer theory of evidence (DSTE)

In Dempster-Shafer theory, the event space is defined by a triple $(\mathcal{S}, \mathbb{S}, m)$ which defines $\mathcal{S}$ the universal set, $\mathbb{S}$ a countable collection of subsets of $\mathcal{S}$, and a notional measure m. $\mathcal{S}$ and $\mathbb{S}$ have a similar meaning to that in classical probability theory; the main difference is that $\mathbb{S}$, also known as the focal elements, does not have to be a σ-algebra over $\mathcal{S}$. The operator m is defined to be

$$m(\mathcal{U}) = \begin{cases} > 0 & \text{if } \mathcal{U} \in \mathbb{S} \\ 0 & \text{if } \mathcal{U} \subset \mathcal{S} \text{ and } \mathcal{U} \notin \mathbb{S} \end{cases} \tag{4.1}$$

$$\sum_{\mathcal{U} \in \mathbb{S}} m(\mathcal{U}) = 1 \tag{4.2}$$

where $m(\mathcal{U})$ is known as the basic probability assignment (BPA) of the set $\mathcal{U}$. In the DSTE framework, belief and plausibility are defined as:

$$\mathrm{Bel}(\mathcal{E}) = \sum_{\{\mathcal{U} \mid \mathcal{U} \subset \mathcal{E},\, \mathcal{U} \in \mathbb{S}\}} m(\mathcal{U}) \tag{4.3}$$

$$\mathrm{Pl}(\mathcal{E}) = \sum_{\{\mathcal{U} \mid \mathcal{U} \cap \mathcal{E} \neq \emptyset,\, \mathcal{U} \in \mathbb{S}\}} m(\mathcal{U}) \tag{4.4}$$

The belief $\mathrm{Bel}(\mathcal{E})$ is interpreted to be the minimum likelihood that is associated with the event $\mathcal{E}$. Similarly, the plausibility $\mathrm{Pl}(\mathcal{E})$ is the maximum amount of likelihood that could be associated with $\mathcal{E}$. This particular structure allows us to handle unconventional inputs, such as conflicting pieces of evidence (e.g. dissenting expert opinions), that would be otherwise discarded in an interval analysis or probabilistic framework. The ability to make use of this information results in a commensurately more informed output.

The procedure to compute belief structures involves four major steps:

1. Determine the set of d-dimensional hypercubes that have a nonzero evidential measure
2. Compute the composite evidential measure (BPA) of each hypercube
3. Propagate each hypercube through the model and obtain the response bounds within each hypercube
4. Aggregate the minimum and maximum values of the response per hypercube with the BPAs to obtain cumulative belief and plausibility functions on the response (e.g. calculate a belief structure on the response).

The first step involves identifying combinations of focal elements defined on the inputs that define a hypercube. The second step involves defining an aggregate BPA for that hypercube, which is the product of the BPAs of the individual focal elements defining the hypercube. The third step involves finding the maximum and minimum values of the response value in each hypercube, and this part can be very computationally expensive. Finally, the results over all hypercubes are aggregated to form belief structures on the response.

Chapter 5

Bayesian Methods

This chapter covers various topics relating to Bayesian methods for inferring input parameter distributions for computer models, which is sometimes called "Bayesian calibration of computer models." One common solution approach for Bayesian calibration involves Markov Chain Monte Carlo (MCMC) sampling. Sections 5.1 to 5.4 describe Bayesian fundamentals and then cover specialized approaches for accelerating the MCMC sampling process used within Bayesian inference. Chain diagnostic metrics for analyzing the convergence of the MCMC chain are discussed in Section 5.5. Section 5.6 describes ways of handling a discrepancy between the model estimate and the responses. Section 5.7 describes a way of determining the optimal experimental design to identify high-fidelity runs that can be used to best inform the calibration of a low-fidelity model. This is followed by a discussion of information-theoretic metrics in Section 5.8. Finally, we conclude this chapter with a discussion of a new Bayesian approach in Section 5.9 which does not use MCMC and relies on a measure-theoretic approach for stochastic inference instead of MCMC. In Dakota, the Bayesian methods called QUESO, GPMSA, and DREAM use Markov Chain Monte Carlo sampling. The Bayesian method called WASABI implements the measure-theoretic approach.

5.1 Fundamentals

Bayes Theorem [78], shown in Eq. 5.1, is used for performing inference. In particular, we derive the plausible parameter values based on the prior probability density and the data $\boldsymbol{d}$. A typical case involves the use of a conservative prior notion of an uncertainty, which is then constrained to be consistent with the observational data. The result is the posterior parameter density of the parameters $f_{\boldsymbol{\Theta}|\boldsymbol{D}}\left(\boldsymbol{\theta}|\boldsymbol{d}\right)$.

$$f_{\boldsymbol{\Theta}|\boldsymbol{D}}\left(\boldsymbol{\theta}|\boldsymbol{d}\right) = \frac{f_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right)\mathcal{L}\left(\boldsymbol{\theta};\boldsymbol{d}\right)}{f_{\boldsymbol{D}}\left(\boldsymbol{d}\right)} \tag{5.1}$$

The likelihood function is used to describe how well a model's predictions are supported by the data. The specific likelihood function currently used in Dakota is a Gaussian likelihood. This means that we assume the difference between the model quantity of interest (e.g. result from a computer simulation) and the experimental observations are Gaussian:

$$d_i = q_i(\boldsymbol{\theta}) + \epsilon_i, \tag{5.2}$$

where $\boldsymbol{\theta}$ are the parameters of a model quantity of interest q_i and ϵ_i is a random variable that can encompass both measurement errors on d_i and modeling errors associated with the simulation quantity of interest $q_i(\boldsymbol{\theta})$. If we

have n observations, the probabilistic model defined by Eq. (5.2) results in a likelihood function for $\boldsymbol{\theta}$ as shown in Eq. 5.3:

$$\mathcal{L}(\boldsymbol{\theta};\boldsymbol{d}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma_d}|}} \exp\left(-\frac{1}{2}\boldsymbol{r}^T\boldsymbol{\Sigma_d}^{-1}\boldsymbol{r}\right), \tag{5.3}$$

where the residual vector $\boldsymbol{r}$ is defined from the differences between the model predictions and the corresponding observational data (i.e., $r_i = q_i(\boldsymbol{\theta}) - d_i$ for $i = 1,\ldots,n$), and $\boldsymbol{\Sigma_d}$ is the covariance matrix of the Gaussian data uncertainties.

The negative log-likelihood is comprised of the misfit function

$$M(\boldsymbol{\theta};\boldsymbol{d}) = \frac{1}{2}\boldsymbol{r}^T\boldsymbol{\Sigma_d}^{-1}\boldsymbol{r} \tag{5.4}$$

plus contributions from the leading normalization factor ($\frac{n}{2}\log(2\pi)$ and $\frac{1}{2}\log(|\boldsymbol{\Sigma_d}|)$). It is evident that dropping $\boldsymbol{\Sigma_d}$ from Eq. 5.4 (or equivalently, taking it to be the identity) results in the ordinary least squares (OLS) approach commonly used in deterministic calibration. For a fixed $\boldsymbol{\Sigma_d}$ (no hyper-parameters in the calibration), minimizing the misfit function is equivalent to maximizing the likelihood function and results in a solution known as the maximum likelihood estimate (MLE), which will be the same as the OLS estimate when the residuals have no relative weighting (any multiple of identity in the data covariance matrix).

When incorporating the prior density, the maximum *a posteriori* probability (MAP) point is the solution that maximizes the posterior probability in Eq. 5.1. This point will differ from the MLE for cases of non-uniform prior probability.

In the sections to follow, we describe approaches for preconditioning the MCMC process by computing a locally-accurate proposal density and for jump-starting the MCMC process by pre-solving for the MAP point. Within Dakota, these are separate options: one can configure a run to use either or both, although it is generally advantageous to employ both when the necessary problem structure (i.e., derivative support) is present.

5.2 Proposal Densities

When derivatives of $q(\theta)$ are readily available (e.g., from adjoint-capable simulations or from emulator models such as polynomial chaos, stochastic collocation, or Gaussian processes), we can form derivatives of the misfit function as

$$\nabla_{\boldsymbol{\theta}} M(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\boldsymbol{q}(\boldsymbol{\theta})^T\,\boldsymbol{\Sigma_d}^{-1}\,\boldsymbol{r} \tag{5.5}$$

$$\nabla^2_{\boldsymbol{\theta}} M(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\boldsymbol{q}(\boldsymbol{\theta})^T\,\boldsymbol{\Sigma_d}^{-1}\,\nabla_{\boldsymbol{\theta}}\boldsymbol{q}(\boldsymbol{\theta}) + \nabla^2_{\boldsymbol{\theta}}\boldsymbol{q}(\boldsymbol{\theta})\cdot\left[\boldsymbol{\Sigma_d}^{-1}\,\boldsymbol{r}\right] \tag{5.6}$$

Neglecting the second term in Eq. 5.6 (a three-dimensional Hessian tensor dotted with the residual vector) results in the Gauss-Newton approximation to the misfit Hessian:

$$\nabla^2_{\boldsymbol{\theta}} M(\boldsymbol{\theta}) \approx \nabla_{\boldsymbol{\theta}}\boldsymbol{q}(\boldsymbol{\theta})^T\,\boldsymbol{\Sigma_d}^{-1}\,\nabla_{\boldsymbol{\theta}}\boldsymbol{q}(\boldsymbol{\theta}) \tag{5.7}$$

This approximation requires only gradients of the residuals, enabling its use in cases where models or model emulators only provide first-order derivative information. Since the second term in Eq. 5.6 includes the residual vector, it becomes less important as the residuals are driven toward zero. This makes the Gauss-Newton approximation a good approximation for solutions with small residuals. It also has the feature of being at least positive semi-definite, whereas the full misfit Hessian may be indefinite in general.

We are interested in preconditioning the MCMC sampling using an accurate local representation of the curvature of the posterior distribution, so we will define the MCMC proposal covariance to be the inverse of the Hessian of

the negative log posterior. From Eq. 5.1 and simplifying notation to π_{post} for the posterior and π_0 for the prior, we have

$$\nabla^2_{\boldsymbol{\theta}} \left[-\log(\pi_{\text{post}}(\boldsymbol{\theta})) \right] = \nabla^2_{\boldsymbol{\theta}} M(\boldsymbol{\theta}) - \nabla^2_{\boldsymbol{\theta}} \left[\log(\pi_0(\boldsymbol{\theta})) \right] \tag{5.8}$$

A typical approach for defining a proposal density is to utilize a multivariate normal (MVN) distribution with mean centered at the current point in the chain and prescribed covariance. Thus, in the specific case of an MVN proposal, we will utilize the fact that the Hessian of the negative log prior for a normal prior distribution is just the inverse covariance:

$$-\nabla^2_{\boldsymbol{\theta}} \left[\log(\pi_0(\boldsymbol{\theta})) \right] = \boldsymbol{\Sigma_0}^{-1} \tag{5.9}$$

For non-normal prior distributions, this is not true and, in the case of uniform or exponential priors, the Hessian of the negative log prior is in fact zero. However, as justified by the approximation of an MVN proposal distribution and the desire to improve the conditioning of the resulting Hessian, we will employ Eq. 5.9 for all prior distribution types.

From here, we follow [108] and decompose the prior covariance into its Cholesky factors, resulting in

$$\begin{aligned} \boldsymbol{H_{\text{nlpost}}} &= \boldsymbol{H_M} + \boldsymbol{\Sigma_0}^{-1} && (5.10) \\ &= \boldsymbol{H_M} + \boldsymbol{L_0}^{-T} \boldsymbol{L_0}^{-1} && (5.11) \\ &= \boldsymbol{L_0}^{-T} \left[\boldsymbol{L_0}^T \boldsymbol{H_M} \boldsymbol{L_0} + \boldsymbol{I} \right] \boldsymbol{L_0}^{-1} && (5.12) \end{aligned}$$

where we again simplify notation to represent $\nabla^2_{\boldsymbol{\theta}} \left[-\log(\pi_{\text{post}}(\boldsymbol{\theta})) \right]$ as $\boldsymbol{H_{\text{nlpost}}}$ and $\nabla^2_{\boldsymbol{\theta}} M(\boldsymbol{\theta})$ as $\boldsymbol{H_M}$. The inverse of this matrix is then

$$\boldsymbol{H}^{-1}_{\boldsymbol{\text{nlpost}}} = \boldsymbol{L_0} \left[\boldsymbol{L_0}^T \boldsymbol{H_M} \boldsymbol{L_0} + \boldsymbol{I} \right]^{-1} \boldsymbol{L_0}^T \tag{5.13}$$

Note that the use of $\boldsymbol{\Sigma_0}^{-1}$ for the Hessian of the negative log prior in Eq. 5.9 provides some continuity between the default proposal covariance and the proposal covariance from Hessian-based preconditioning: if the contributions from $\boldsymbol{H_M}$ are neglected, then $\boldsymbol{H}^{-1}_{\boldsymbol{\text{nlpost}}} = \boldsymbol{\Sigma_0}$, the default.

To address the indefiniteness of $\boldsymbol{H_M}$ (or to reduce the cost for large-scale problems by using a low-rank Hessian approximation), we perform a symmetric eigenvalue decomposition of this prior-preconditioned misfit and truncate any eigenvalues below a prescribed tolerance, resulting in

$$\boldsymbol{L_0}^T \boldsymbol{H_M} \boldsymbol{L_0} \approx \boldsymbol{V}_r \boldsymbol{\Lambda}_r \boldsymbol{V}_r^T. \tag{5.14}$$

for a matrix $\boldsymbol{V}_r$ of truncated eigenvectors and a diagonal matrix of truncated eigenvalues $\boldsymbol{\Lambda}_r = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_r)$. We then apply the Sherman-Morrison-Woodbury formula to invert the sum of the decomposed matrix and identity as

$$\left[\boldsymbol{V}_r \boldsymbol{\Lambda}_r \boldsymbol{V}_r^T + \boldsymbol{I} \right]^{-1} = \boldsymbol{I} - \boldsymbol{V}_r \boldsymbol{D}_r \boldsymbol{V}_r^T. \tag{5.15}$$

for $\boldsymbol{D}_r = \text{diag}(\frac{\lambda_1}{\lambda_1+1}, \frac{\lambda_2}{\lambda_2+1}, \ldots, \frac{\lambda_r}{\lambda_r+1})$. We now arrive at our final result for the covariance of the MVN proposal density:

$$\boldsymbol{\Sigma_{MVN}} = \boldsymbol{H}^{-1}_{\boldsymbol{\text{nlpost}}} \approx \boldsymbol{L_0} \left[\boldsymbol{I} - \boldsymbol{V}_r \boldsymbol{D}_r \boldsymbol{V}_r^T \right] \boldsymbol{L_0}^T \tag{5.16}$$

5.3 Pre-solve for MAP point

When an emulator model is in use, it is inexpensive to pre-solve for the MAP point by finding the optimal values for $\boldsymbol{\theta}$ that maximize the log posterior (minimize the negative log posterior):

$$\boldsymbol{\theta}_{MAP} = \arg\min_{\boldsymbol{\theta}} \left[-\log(\pi_{\text{post}}(\boldsymbol{\theta})) \right] \tag{5.17}$$

This effectively eliminates the burn-in procedure for an MCMC chain where some initial portion of the Markov chain is discarded, as the MCMC chain can instead be initiated from a high probability starting point: the MAP solution. Further, a full Newton optimization solver can be used with the Hessian defined from Eq. 5.8, irregardless of whether the misfit Hessian is a full Hessian (residual values, gradients, and Hessians are available for Eq 5.6) or a Gauss-Newton Hessian (residual gradients are available for Eq 5.7). Note that, in this case, there is no MVN approximation as in §5.2, so we will not employ Eq. 5.9. Rather, we employ the actual Hessians of the negative log priors for the prior distributions in use.

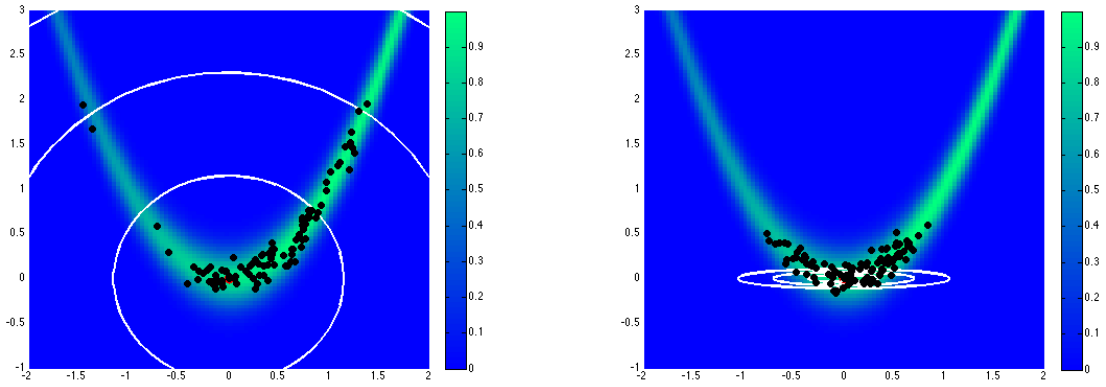
5.4 Rosenbrock Example

Defining two residuals as:

$$r_1 = 10(\theta_2 - \theta_1^2) \tag{5.18}$$
$$r_2 = 1 - \theta_1 \tag{5.19}$$

with $\boldsymbol{d} = \mathbf{0}$ and $\boldsymbol{\Sigma_d} = \text{diag}(\mathbf{.5})$, it is evident from Eq. 5.4 that $M(\theta; d)$ is exactly the Rosenbrock function[1] with its well-known banana-shaped contours.

Assuming a uniform prior on $[-2, 2]$, Figure 5.1 shows the effect of different proposal covariance components, with the default prior covariance ($\boldsymbol{\Sigma_{MVN}} = \boldsymbol{\Sigma_0}$) in Figure 5.1(a) and a misfit Hessian-based proposal covariance ($\boldsymbol{\Sigma_{MVN}} = \boldsymbol{H_M^{-1}}$) in Figure 5.1(b). Rejection rates for 2000 MCMC samples were 73.4% for the former and



(a) Proposal covariance defined from uniform prior.

(b) Proposal covariance defined from misfit Hessian.

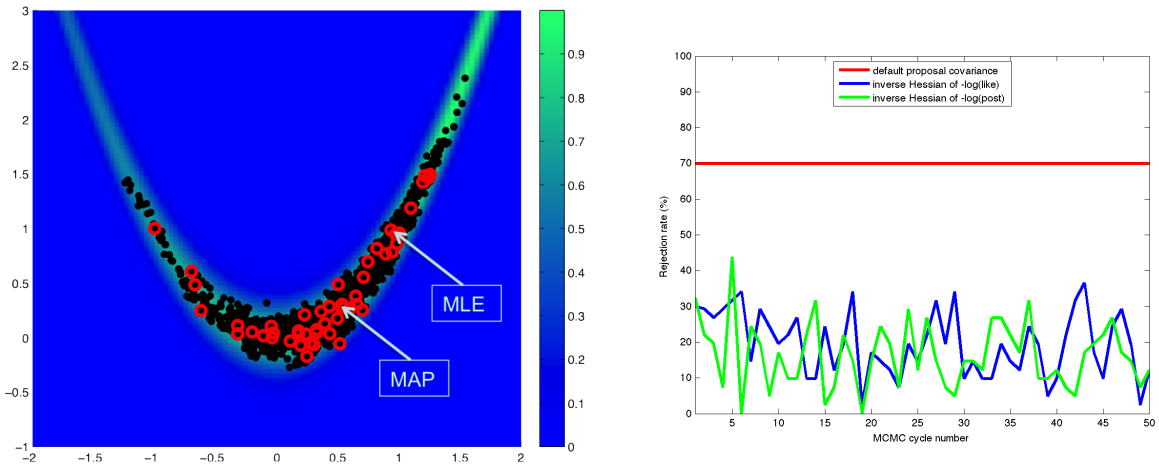
Figure 5.1: Depiction of proposal covariance at (0,0) with contours at one/two/three standard deviations. 2000 MCMC samples are performed and every 20th sample is plotted.

25.6% for the latter. Reducing the number of MCMC samples to 40, for purposes of assessing local proposal accuracy, results in a similar 72.5% rejection rate for prior-based proposal covariance and a reduced 17.5% rate for misfit Hessian-based proposal covariance. The prior-based proposal covariance only provides a global scaling and omits information on the structure of the likelihood; as a result, the rejection rates are relatively high for this problem and are not a strong function of location or chain length. The misfit Hessian-based proposal covariance, on the other hand, provides accurate local information on the structure of the likelihood, resulting in low rejection

[1] The two-dimensional Rosenbrock test function is defined as $100(x_2 - x_1^2)^2 + (1 - x_1)^2$

rates for samples in the vicinity of this Hessian update. Once the chain moves away from this vicinity, however, the misfit Hessian-based approach may become inaccurate and actually impede progress. This implies the need to regularly update a Hessian-based proposal covariance to sustain these MCMC improvements.

In Figure 5.2, we show a result for a total of 2000 MCMC samples initiated from $(-1, 1)$, where we restart the chain with an updated Hessian-based proposal covariance every 40 samples (Dakota specification: `samples = 2000 proposal_updates = 50`). This case uses a standard normal prior, resulting in differences in the MLE and MAP estimates, as shown in Figure 5.2(a). Figure 5.2(b) shows the history of rejection rates for each of the 50 chains for misfit Hessian-based proposals ($\boldsymbol{\Sigma_{MVN}} = \boldsymbol{H_M^{-1}}$) and negative log posterior Hessian-based proposals ($\boldsymbol{\Sigma_{MVN}} = \boldsymbol{H_{\mathbf{nlpost}}^{-1}}$) compared to the rejection rate for a single 2000-sample chain using prior-based proposal covariance ($\boldsymbol{\Sigma_{MVN}} = \boldsymbol{\Sigma_0}$). A standard normal prior is not a strong prior in this case, and the posterior



(a) Restarted chain.

(b) Rejection rates.

Figure 5.2: MCMC with Hessian-based proposals and standard normal prior. Left shows chain with 2000 total samples (black points) and 50 proposal updates (red circles) using the inverse of the misfit Hessian. Right shows rejection rates for misfit Hessian-based and posterior Hessian-based proposals compared to default prior covariance proposal.

is likelihood dominated. This leads to similar performance from the two Hessian-based proposals, with average rejection rates of 70%, 19.5%, and 16.4% for prior-based, misfit Hessian-based, and posterior Hessian-based cases, respectively.

5.5 Chain Diagnostics

The implementation of a number of metrics for assessing the convergence of the MCMC chain drawn during Bayesian calibration is undergoing active development in Dakota. As of Dakota 6.10, `confidence_intervals` is the only diagnostic implemented.

5.5.1 Confidence Intervals

Suppose g is a function that represents some characteristic of the probability distribution π underlying the MCMC chain [54], such as the mean or variance. Then under the standard assumptions of an MCMC chain, the true

expected value of this function, $\mathbb{E}_\pi g$ can be approximated by taking the mean over the n samples in the MCMC chain, denoted $X = \{X_1, X_2, \ldots, X_n\}$,

$$\bar{g}_n = \frac{1}{n} \sum_{i=1}^{n} g(X_i). \tag{5.20}$$

The error in this approximation converges to zero, such that

$$\sqrt{n}\,(\bar{g}_n - \mathbb{E}_\pi g) \to \mathcal{N}(0, \sigma_g^2), \quad n \to \infty. \tag{5.21}$$

Thus, in particular, we would like to estimate the variance of this error, σ_g^2. Let $\hat{\sigma}_n^2$ be a consistent estimator of the true variance σ_g^2. Then

$$\bar{g}_n \pm t_* \frac{\hat{\sigma}_n}{\sqrt{n}} \tag{5.22}$$

is an asymptotically valid interval estimator of $\mathbb{E}_\pi g$, where t_* is a Student's t quantile. In Dakota, confidence intervals are computed for the mean and variance of the parameters and of the responses, all confidence intervals are given at the 95th percentile, and $\hat{\sigma}_n$ is calculated via non-overlapping batch means, or "batch means" for simplicity.

When batch means is used to calculate $\hat{\sigma}_n$, the MCMC chain X is divided into blocks of equal size. Thus, we have a_n batches of size b_n, and $n = a_n b_n$. Then the batch means estimate of σ_g^2 is given by

$$\hat{\sigma}_n^2 = \frac{b_n}{a_n - 1} \sum_{k=0}^{a_n - 1} \left(\bar{g}_k - \bar{g}_n\right)^2, \tag{5.23}$$

where $\bar{g}_k$ is the expected value of g for batch $k = 0, \ldots, a_n - 1$,

$$\bar{g}_k = \frac{1}{b_n} \sum_{i=1}^{b_n} g\left(X_{k b_n + i}\right). \tag{5.24}$$

It has been found that an appropriate batch size is $b_n = \lfloor \sqrt{n} \rfloor$. Further discussion and comparison to alternate estimators of σ_g^2 can be found in [54].

Confidence intervals may be used as a chain diagnostic by setting fixed-width stopping rules [115]. For example, if the width of one or more intervals is below some threshold value, that may indicate that enough samples have been drawn. Common choices for the threshold value include:

- Fixed width: ϵ
- Relative magnitude: $\epsilon \|\bar{g}_n\|$
- Relative standard deviation: $\epsilon \|\hat{\sigma}_n\|$

If the chosen threshold is exceeded, `samples` may need to be increased. Sources [54, 115] suggest increasing the sample size by 10% and reevaluating the diagnostics for signs of chain convergence.

If `output` is set to `debug`, the sample mean and variance for each batch (for each parameter and response) is output to the screen. The user can then analyze the convergence of these batch means in order to deduce whether the MCMC chain has converged.

5.6 Model Discrepancy

Whether in a Bayesian setting or otherwise, the goal of model calibration is to minimize the difference between the observational data d_i and the corresponding model response $q_i(\boldsymbol{\theta})$. That is, one seeks to minimize the misfit 5.4. For a given set of data, this formulation explicitly depends on model parameters that are to be adjusted and implicitly on conditions that may vary between experiments, such as temperature or pressure. These experimental conditions can be represented in Dakota by configuration variables, in which case Eq. 5.2 can be rewritten,

$$d_i(x) = q_i(\boldsymbol{\theta}, x) + \epsilon_i, \tag{5.25}$$

where x represents the configuration variables. Updated forms of the likelihood 5.3 and misfit 5.4 are easily obtained.

It is often the case that the calibrated model provides an insufficient fit to the experimental data. This is generally attributed to model form or structural error, and can be corrected to some extent with the use of a model discrepancy term. The seminal work in model discrepancy techniques, Kennedy and O'Hagan [81], introduces an additive formulation

$$d_i(x) = q_i\left(\boldsymbol{\theta}, x\right) + \delta_i(x) + \epsilon_i, \tag{5.26}$$

where $\delta_i(x)$ represents the model discrepancy. For scalar responses, δ_i depends *only* on the configuration variables, and one discrepancy model is calculated for *each* observable d_i, $i = 1, \ldots, n$, yielding $\delta_1, \ldots, \delta_n$. For field responses in which the observational data and corresponding model responses are also functions of independent field coordinates such as time or space, Eq. 5.26 can be rewritten as

$$d(t, x) = q(t, \boldsymbol{\theta}, x) + \delta(t, x) + \epsilon. \tag{5.27}$$

In this case, a single, global discrepancy model δ is calculated over the entire field. The current model discrepancy implementation in Dakota has not been tested for cases in which scalar and field responses are mixed.

The Dakota implementation of model discrepancy for scalar responses also includes the calculation of prediction intervals for each prediction configuration $x_{k,new}$. These intervals capture the uncertainty in the discrepancy approximation as well as the experimental uncertainty in the response functions. It is assumed that the uncertainties, representated by their respective variance values, are combined additively for each observable i such that

$$\Sigma_{total,i}(x) = \Sigma_{\delta,i}(x) + \sigma^2_{exp,i}(x)I, \tag{5.28}$$

where $\Sigma_{\delta,i}$ is the variance of the discrepancy function, and $\sigma^2_{exp,i}$ is taken from the user-provided experimental variances. The experimental variance provided for parameter calibration may vary for the same observable from experiment to experiment, thus $\sigma^2_{exp,i}$ is taken to be the maximum variance given for each observable. That is,

$$\sigma^2_{exp,i} = \max_j \sigma^2_i(x_j), \tag{5.29}$$

where $\sigma^2_i(x_j)$ is the variance provided for the i^{th} observable d_i, computed or measured with the configuration variable x_j. When a Gaussian process discrepancy function is used, the variance is calculated according to Eq. 6.4. For polynomial discrepancy functions, the variance is given by Eq. 6.25.

It should be noted that a Gaussian process discrepancy function is used when the response is a field instead of a scalar; the option to use a polynomial discrepancy function has not yet been activated. The variance of the discrepancy function $\Sigma_{\delta,i}$ is calculated according to Eq. 6.4. Future work includes extending this capability to include polynomial discrepancy formulations for field responses, as well as computation of prediction intervals which include experimental variance information.

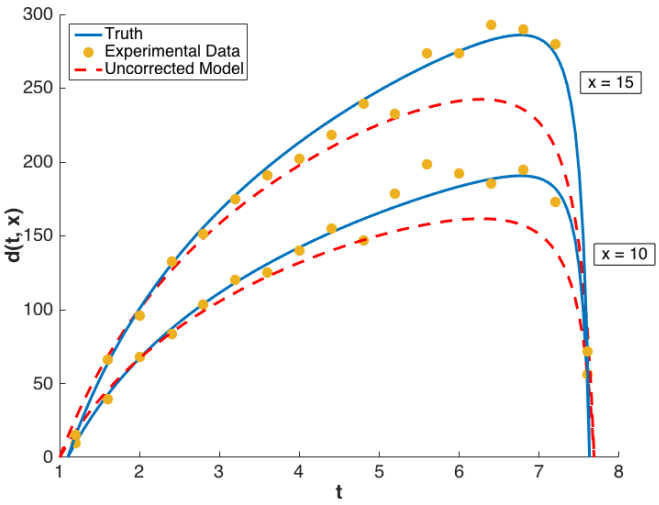


Figure 5.3: Graphs of the uncorrected model output $m(t, x)$, the truth $y(t, x)$, and experimental data $d(t, x)$ for configurations $x = 10$ and $x = 15$.

5.6.1 Scalar Responses Example

For the purposes of illustrating the model discrepancy capability implemented in Dakota, consider the following example. Let the "truth" be given by

$$y(t, x) = 10.5x \log(t - 0.1) - \frac{x}{(t - 0.1 - \theta^*)^2}, \tag{5.30}$$

where t is the independent variable, x is the configuration parameter, and θ^* is 7.75, the "true" value of the parameter θ. Let the "model" be given by

$$m(t, \theta, x) = \frac{10x \log(t)(t - \theta)^2 - x}{(t - 8)^2}. \tag{5.31}$$

Again, t is the independent variable and x is the configuration parameter, and θ now represents the model parameter to be calibrated. It is clear from the given formulas that the model is structurally different from the truth and will be inadequate.

The "experimental" data is produced by considering two configurations, $x = 10$ and $x = 15$. Data points are taken over the range $t \in [1.2, 7.6]$ at intervals of length $\Delta t = 0.4$. Normally distributed noise ϵ_i is added such that

$$d_i(x_j) = y(t_i, x_j) + \epsilon_i, \tag{5.32}$$

with $i = 1, \ldots, 17$ and $j = 1, 2$. Performing a Bayesian update in Dakota yields a posterior distribution of θ that is tightly peaked around the value $\bar{\theta} = 7.9100$. Graphs of $m(t, \bar{\theta}, 10)$ and $m(t, \bar{\theta}, 15)$ are compared to $y(t, 10)$ and $y(t, 15)$, respectively, for $t \in [1.2, 7.6]$ in Figure 5.3, from which it is clear that the model insufficiently captures the given experimental data.

Following the Bayesian update, Dakota calculates the model discrepancy values

$$\delta_i(x_j) = d_i(x_j) - m_i(\bar{\theta}, x_j) \tag{5.33}$$

for the experimental data points, *i.e.* for $i = 1, \ldots, 17$ and $j = 1, 2$. Dakota then approximates the model discrepancy functions $\delta_1(x), \ldots \delta_{17}(x)$, and computes the responses and prediction intervals of the corrected model $m_i(\bar{\theta}, x_{j,new}) + \delta_i(x_{j,new})$ for each prediction configuration. The prediction intervals have a radius of two times the standard deviation calculated with 5.28. The discrepancy function in this example was taken to be a Gaussian process with a quadratic trend, which is the default setting for the model discrepancy capability in Dakota.

The prediction configurations are taken to be $x_{new} = 5, 5.5, \ldots, 20$. Examples of two corrected models are shown in Figure 5.4. The substantial overlap in the measurement error bounds and the corrected model prediction intervals indicate that the corrected model is sufficiently accurate. This conclusion is supported by Figure 5.5, in which the "truth" models for three prediction figurations are compared to the corrected model output. In each case, the truth falls within the prediction intervals.

5.6.2 Field Responses Example

To illustrate the model discrepancy capability for field responses, consider the stress-strain experimental data shown in Figure 5.6. The configuration variable in this example represents temperature. Unlike the example discussed in the previous section, the domain of the independent variable (strain) differs from temperature to temperature, as do the shapes of the stress curves. This presents a challenge for the simulation model as well as the discrepancy model.

Let the "model" be given by

$$m(t, \boldsymbol{\theta}, x) = \theta_1 \left[\frac{\log(100t + 1)}{x^0.5} - \frac{1}{x^{0.2}\left(100t - 1.05\left(\frac{x}{100} - 6.65\right)^2 \theta_2\right)^2} \right], \tag{5.34}$$

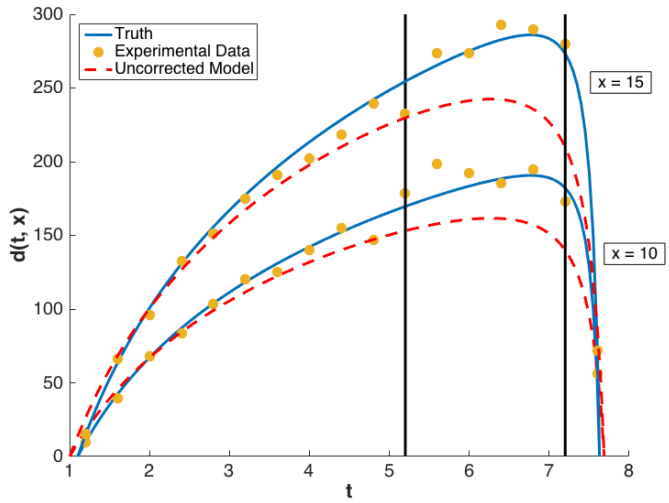
where t is the independent variable (strain) and x is the configuration parameter (temperature). Note that there are two parameters to be calibrated, $\boldsymbol{\theta} = (\theta_1, \theta_2)$.

The average and standard deviation of the experimental data at temperatures $x = 373.15$, $x = 673.15$, and $x = 973.15$ are calculated and used as calibration data. The four remaining temperatures will be used to evaluate the performance of the corrected model. The calibration data and the resulting calibrated model are shown in Figure 5.7. With experimental data, the observations may not be taken at the same independent field coordinates, so the keyword `interpolate` can be used in the `responses` block of the Dakota input file. The uncorrected model does not adequately capture the experimental data.

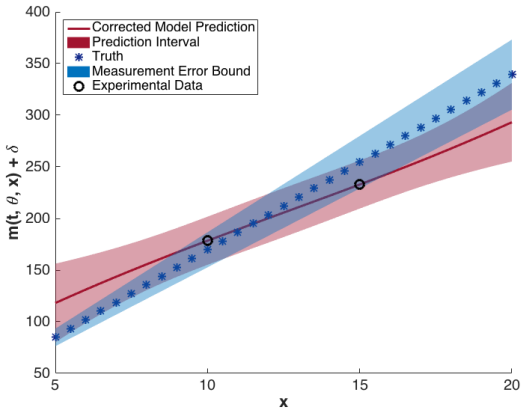
Following the Bayesian update, Dakota calculates the build points of the model discrepancy,

$$\delta(t_i, x_j) = d(t_i, x_j) - m(t_i, \boldsymbol{\theta}, x_j), \tag{5.35}$$

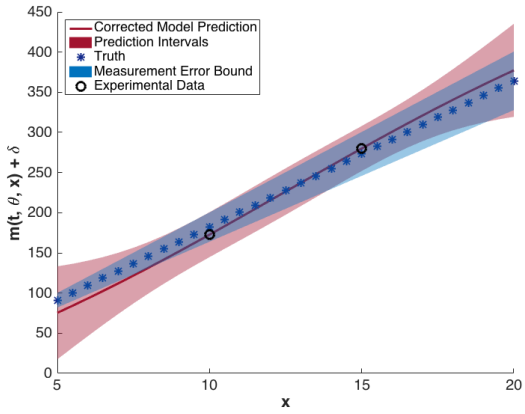
for each experimental data point. Dakota then approximates the global discrepancy function $\delta(t, x)$ and computes the corrected model responses $m(t_i, \boldsymbol{\theta}, x_{j,new}) + \delta(t_i, x_{j,new})$ and variance of the discrepancy model $\sigma_{\delta, x_{j,new}}$ for each prediction configuration. The corrected model for the prediction configurations is shown in Figure 5.8. The corrected model is able to capture the shape of the stress-strain curve quite well in all four cases; however, the point of failure is difficult to capture for the extrapolated temperatures. The difference in shape and point of failure between temperatures may also explain the large variance in the discrepancy model.



(a) Locations of the corrected models shown in (b) and (c) below.



(b) Corrected model values with prediction intervals for t = 5.2



(c) Corrected model values with prediction intervals for t = 7.2

Figure 5.4: Illustrated of corrected model for two observables. Note that the models shown in (b) and (c) are functions of the configuration variable x. Therefore, they traverse the vertical lines shown in (a).

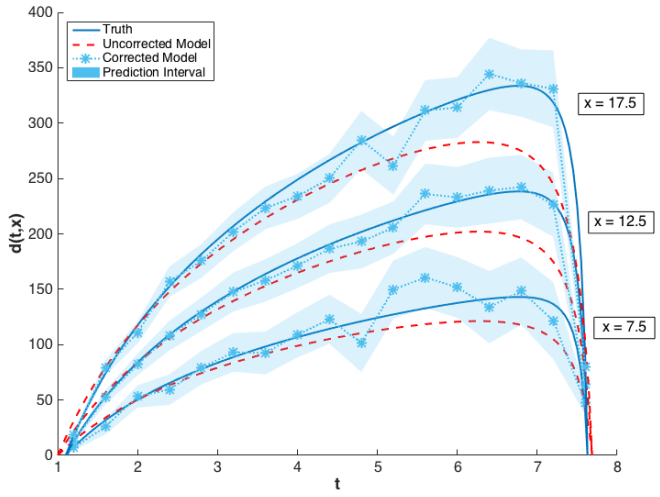


Figure 5.5: The graphs of $y(t, x)$ for $x = 7.5, 12.5, 17.5$ are compared to the corrected model and its prediction intervals. The uncorrected model is also shown to illustrate its inadequacy.

5.7 Experimental Design

Experimental design algorithms seek to add observational data that informs model parameters and reduces their uncertainties. Typically, the observational data $\boldsymbol{d}$ used in the Bayesian update 5.1 is taken from physical experiments. However, it is also common to use the responses or output from a high-fidelity model as $\boldsymbol{d}$ in the calibration of a low-fidelity model. Furthermore, this calibration can be done with a single Bayesian update or iteratively with the use of experimental design. The context of experimental design mandates that the high-fidelity model or physical experiment depend on design conditions or configurations, such as temperature or spatial location. After a preliminary Bayesian update using an initial set of high-fidelity (or experimental) data, the next "best" design points are determined and used in the high-fidelity model to augment $\boldsymbol{d}$, which is used in subsequent Bayesian updates of the low-fidelity model parameters.

The question then becomes one of determining the meaning of "best." In information theory, the mutual information is a measure of the reduction in the uncertainty of one random variable due to the knowledge of another [31]. Recast into the context of experimental design, the mutual information represents how much the proposed experiment and resulting observation would reduce the uncertainties in the model parameters. Therefore, given a set of experimental design conditions, that which maximizes the mutual information is the most desirable. This is the premise that motivates the Bayesian experimental design algorithm implemented in Dakota.

The initial set of high-fidelity data may be either user-specified or generated within Dakota by performing Latin Hypercube Sampling on the space of configuration variables specified in the input file. If Dakota-generated, the design variables will be run through the high-fidelity model specified by the user to produce the initial data set. Whether user-specified or Dakota-generated, this initial data is used in a Bayesian update of the low-fidelity model parameters.

It is important to note that the low-fidelity model depends on both parameters to be calibrated $\boldsymbol{\theta}$ and the design conditions $\boldsymbol{\xi}$. During Bayesian calibration, $\boldsymbol{\xi}$ are not calibrated; they do, however, play an integral role in the

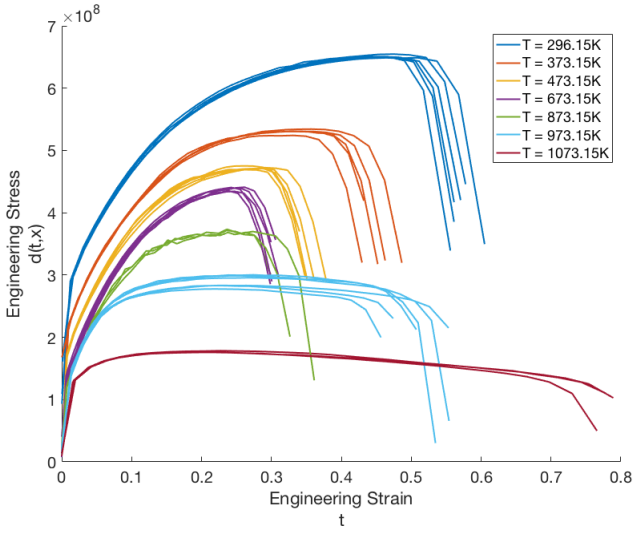


Figure 5.6: Graphs of the experimental data $d(t,x)$ for configurations (temperatures) ranging from $x = 296.15K$ to $x = 1073.15K$.

calculation of the likelihood. Let us rewrite Bayes' Rule as

$$f_{\Theta|D}\left(\boldsymbol{\theta}|\boldsymbol{d}(\boldsymbol{\xi})\right) = \frac{f_{\Theta}\left(\boldsymbol{\theta}\right)\mathcal{L}\left(\boldsymbol{\theta};\boldsymbol{d}(\boldsymbol{\xi})\right)}{f_{D}\left(\boldsymbol{d}(\boldsymbol{\xi})\right)}, \tag{5.36}$$

making explicit the dependence of the data on the design conditions. As in Section 5.1, the difference between

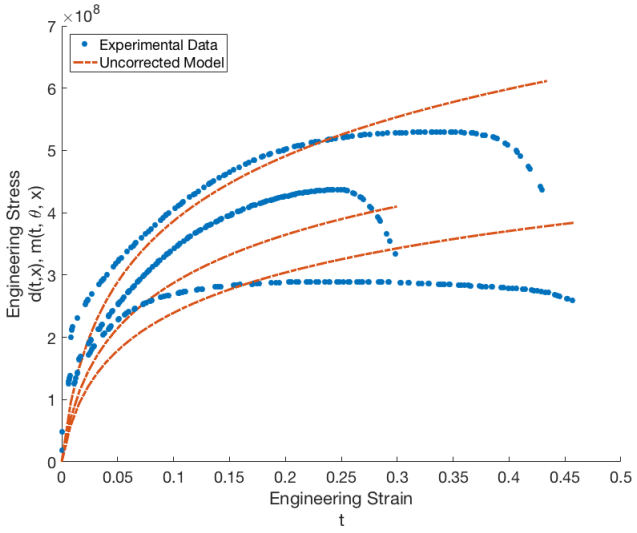
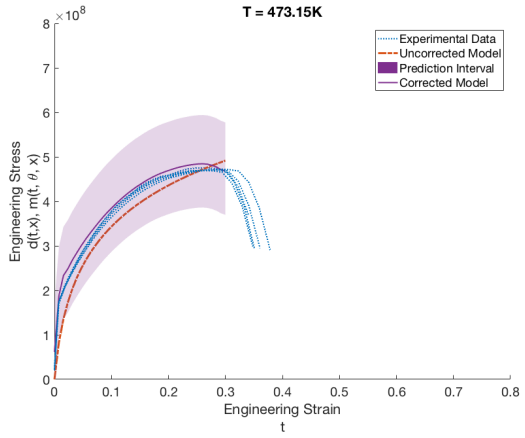
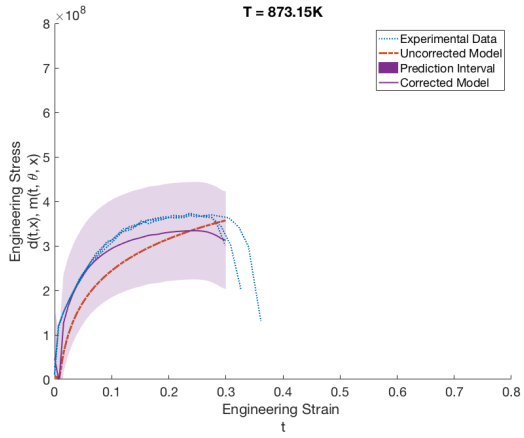


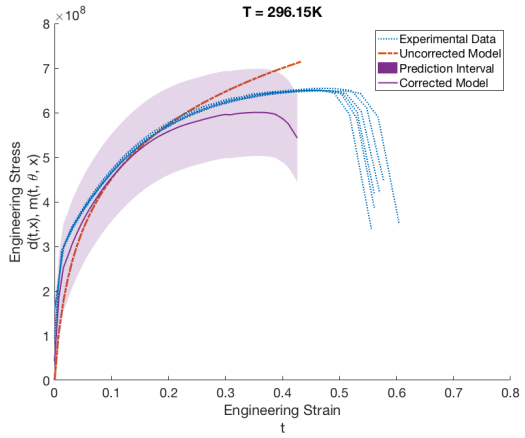
Figure 5.7: Graphs of the calibration data $d(t,x)$ and uncorrected calibrated model $m(t,\boldsymbol{\theta},x)$ for configurations (temperatures) $x = 373.15K$, $x = 673.15K$, and $x = 973.15K$.



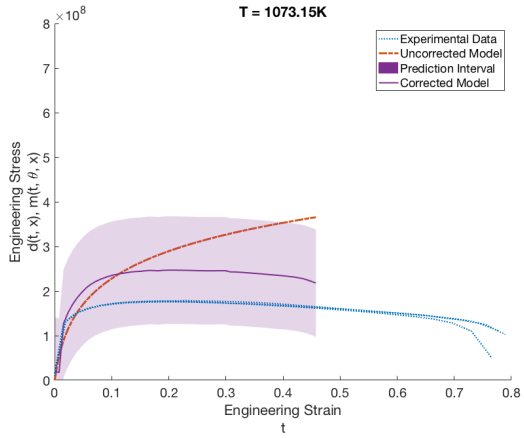
(a) Corrected model for interpolation configuration $x = 473.15K$



(b) Corrected model for interpolation configuration $x = 873.15K$



(c) Corrected model for extrapolation configuration $x = 273.15K$



(d) Corrected model for extrapolation configuration $x = 1073.15K$

Figure 5.8: Illustration of corrected model for four prediction configurations. Two prediction configurations are interpolations of the calibration configurations, and two prediction configurations are extrapolations of the calibration configurations. The corrected model is able to capture the shape of the interpolated configurations quite well. The corrected model for the extrapolated configurations also performs quite well, but it is unable to capture the point of failure. In all cases, the experimental data falls within the bounds of the prediction intervals. The "prediction intervals" have a width of $0.5\sigma_\delta$ and do not incorporate experimental variance.

the high-fidelity and low-fidelity model responses is assumed to be Gaussian such that

$$d_i(\boldsymbol{\xi_j}) = q_i(\boldsymbol{\theta}, \boldsymbol{\xi}_j) + \epsilon_i, \tag{5.37}$$

where $\boldsymbol{\xi}_j$ are the configuration specifications of the jth experiment. The experiments are considered to be independent, making the misfit

$$M(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi})) = \frac{1}{2}\sum_{j=1}^{m}\left(\boldsymbol{d}(\boldsymbol{\xi}_j) - \boldsymbol{q}(\boldsymbol{\theta}, \boldsymbol{\xi}_j)\right)^T \boldsymbol{\Sigma}_{\boldsymbol{d}}^{-1}\left(\boldsymbol{d}(\boldsymbol{\xi}_j) - \boldsymbol{q}(\boldsymbol{\theta}, \boldsymbol{\xi}_j)\right). \tag{5.38}$$

At the conclusion of the initial calibration, a set of candidate design conditions is proposed. As before, these may be either user-specified or generated within Dakota via Latin Hypercube Sampling of the design space. Among these candidates, we seek that which maximizes the mutual information,

$$\boldsymbol{\xi}^* = \arg\max_{\boldsymbol{\xi}_j} I(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi}_j)), \tag{5.39}$$

where the mutual information is given by

$$I(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi}_j)) = \iint f_{\boldsymbol{\Theta},\boldsymbol{D}}\left(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi}_j)\right) \log \frac{f_{\boldsymbol{\Theta},\boldsymbol{D}}\left(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi}_j)\right)}{f_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}\right) f_{\boldsymbol{D}}\left(\boldsymbol{d}(\boldsymbol{\xi}_j)\right)} d\boldsymbol{\theta} d\boldsymbol{d}. \tag{5.40}$$

The mutual information must, therefore, be computed for each candidate design point $\boldsymbol{\xi}_j$. There are two k-nearest neighbor methods available in Dakota that can be used to approximate Eq. 5.40, both of which are derived in [85]. Within Dakota, the posterior distribution $f_{\boldsymbol{\Theta}|\boldsymbol{D}}\left(\boldsymbol{\theta}|\boldsymbol{d}(\boldsymbol{\xi})\right)$ is given by MCMC samples. From these, N samples are drawn and run through the low-fidelity model with $\boldsymbol{\xi}_j$ fixed. This creates a matrix whose rows consist of the vector $\boldsymbol{\theta}^i$ and the low-fidelity model responses $\tilde{\boldsymbol{d}}(\boldsymbol{\theta}^i, \boldsymbol{\xi}_j)$ for $i = 1, \ldots, N$. These rows represent the joint distribution between the parameters and model responses. For each row X_i, the distance to its k^{th}-nearest neighbor among the other rows is approximated $\varepsilon_i = \|X_i - X_{k(i)}\|_\infty$. As noted in [89], k is often taken to be six. The treatment of the marginal distributions is where the two mutual information algorithms differ. In the first algorithm, the marginal distributions are considered by calculating $n_{\boldsymbol{\theta},i}$, which is the number of parameter samples that lie within ε_i of $\boldsymbol{\theta}^i$, and $n_{\boldsymbol{d},i}$, which is the number of responses that lie within ε_i of $\tilde{\boldsymbol{d}}(\boldsymbol{\theta}^i, \boldsymbol{\xi}_j)$. The mutual information then is approximated as [85]

$$I(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi}_j)) \approx \psi(k) + \psi(N) - \frac{1}{N-1}\sum_{i=1}^{N}\left[\psi(n_{\boldsymbol{\theta},i}) - \psi(n_{\boldsymbol{d},i})\right], \tag{5.41}$$

where $\psi(\cdot)$ is the digamma function.

In the second mutual information approximation method, X_i and all of its k-nearest neighbors such that $\|X_i - X_l\|_\infty < \varepsilon_i$ are projected into the marginal subspaces for $\boldsymbol{\theta}$ and $\tilde{\boldsymbol{d}}$. The quantity $\varepsilon_{\boldsymbol{\theta},i}$ is then defined as the radius of the l_∞-ball containing all $k+1$ projected values of $\boldsymbol{\theta}_l$. Similarly, $\varepsilon_{\boldsymbol{d},i}$ is defined as the radius of the l_∞-ball containing all $k+1$ projected values of $\tilde{\boldsymbol{d}}(\boldsymbol{\theta}_l, \boldsymbol{\xi}_j)$ [58]. In this version of the mutual information calculation, $n_{\boldsymbol{\theta},i}$ is the number of parameter samples that lie within $\varepsilon_{\boldsymbol{\theta},i}$ of $\boldsymbol{\theta}^i$, and $n_{\boldsymbol{d},i}$ is the number of responses that lie within $\varepsilon_{\boldsymbol{d},i}$ of $\tilde{\boldsymbol{d}}(\boldsymbol{\theta}^i, \boldsymbol{\xi}_j)$. The mutual information then is approximated as [85]

$$I(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi}_j)) \approx \psi(k) + \psi(N) - \frac{1}{k} - \frac{1}{N-1}\sum_{i=1}^{N}\left[\psi(n_{\boldsymbol{\theta},i}) - \psi(n_{\boldsymbol{d},i})\right]. \tag{5.42}$$

By default, Dakota uses Eq. 5.41 to approximate the mutual information. The user may decide to use Eq. 5.42 by including the keyword `ksg2` in the Dakota input script. An example can be found in [2]. Users also have the option of specifying statistical noise in the low-fidelity model through the `simulation_variance` keyword. When this option is included in the Dakota input file, a random "error" is added to the low-fidelity model responses when the matrix X is built. This random error is normally distributed, with variance equal to `simulation_variance`.

Once the optimal design $\boldsymbol{\xi}^*$ is identified, it is run through the high-fidelity model to produce a new data point $\boldsymbol{d}(\boldsymbol{\xi}^*)$, which is added to the calibration data. Theoretically, the current posterior $f_{\boldsymbol{\Theta}|\boldsymbol{D}}\left(\boldsymbol{\theta}|\boldsymbol{d}(\boldsymbol{\xi})\right)$ would become the prior in the new Bayesian update, and the misfit would compare the low-fidelity model output *only* to the new data point. However, as previously mentioned, we do not have the posterior distribution; we merely have a set of samples of it. Thus, each time the set of data is modified, the *user-specified* prior distribution is used and a full Bayesian update is performed from scratch. If none of the three stopping criteria is met, $\boldsymbol{\xi}^*$ is removed from the set of candidate points, and the mutual information is approximated for those that remain using the newly updated parameters. These stopping criteria are:

- the user-specified maximum number of high-fidelity model evaluations is reached (this does not include those needed to create the initial data set)
- the relative change in mutual information from one iteration to the next is sufficiently small (less than 5%)
- the set of proposed candidate design conditions has been exhausted

If any one of these criteria is met, the algorithm is considered complete.

5.7.1 Batch Point Selection

The details of the experimental design algorithm above assume only one optimal design point is being selected for each iteration of the algorithm. The user may specify the number of optimal design points to be concurrently selected by including the `batch_size` in the input script. The optimality condition 5.39 is then replaced by

$$\{\boldsymbol{\xi}^*\} = \arg\max I\left(\boldsymbol{\theta}, \{\boldsymbol{d}(\boldsymbol{\xi})\}\right), \tag{5.43}$$

where $\{\boldsymbol{\xi}^*\} = \{\boldsymbol{\xi}_1^*, \boldsymbol{\xi}_2^*, \ldots, \boldsymbol{\xi}_s^*\}$ is the set of optimal designs, s being defined by `batch_size`. If the set of design points from which the optimal designs are selected is of size m, finding $\{\boldsymbol{\xi}^*\}$ as in 5.43 would require $m!/(m-s)!$ mutual information calculations, which may become quite costly. Dakota therefore implements a greedy batch point selection algorithm in which the first optimal design,

$$\boldsymbol{\xi}_1^* = \arg\max_{\boldsymbol{\xi}_j} I(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi}_j)), \tag{5.44}$$

is identified, and then used to find the second,

$$\boldsymbol{\xi}_2^* = \arg\max_{\boldsymbol{\xi}_j} I(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi}_j)|\boldsymbol{d}(\boldsymbol{\xi}_1^*)). \tag{5.45}$$

Generally, the i^{th} selected design will satisfy

$$\boldsymbol{\xi}_i^* = \arg\max_{\boldsymbol{\xi}_j} I(\boldsymbol{\theta}, \boldsymbol{d}(\boldsymbol{\xi}_j)|\boldsymbol{d}(\boldsymbol{\xi}_1^*), \ldots, \boldsymbol{d}(\boldsymbol{\xi}_{i-1}^*)). \tag{5.46}$$

The same mutual information calculation algorithms 5.41 and 5.42 described above are applied when calculating the conditional mutual information. The additional low-fidelity model information is appended to the responses portion of the matrix X, and the calculation of ε_i or $\varepsilon_{\boldsymbol{d},i}$ as well as $n_{\boldsymbol{d},i}$ are adjusted accordingly.

5.8 Information Theoretic Tools

The notion of the entropy of a random variable was introduced by C.E. Shannon in 1948 [119]. So named for its resemblance to the statistical mechanical entropy, the Shannon entropy (or simply the entropy), is characterized by the probability distribution of the random variable being investigated. For a random variable $X \in \mathcal{X}$ with probability distribution function p, the entropy h is given by

$$h(p) = -\int_{\mathcal{X}} p(x) \log p(x) dx. \tag{5.47}$$

The entropy captures the average uncertainty in a random variable [31], and is therefore quite commonly used in predictive science. The entropy also provides the basis for other information measures, such as the relative entropy and the mutual information, both of which compare the information content between two random variables but have different purposes and interpretations.

The relative entropy provides a measure of the difference between two probability distributions. It is characterized by the Kullback-Leibler Divergence,

$$D_{KL}(p\|q) = \int p(x) \log \frac{p(x)}{q(x)} dx, \tag{5.48}$$

which can also be written as

$$D_{KL}(p\|q) = h(p,q) - h(p), \tag{5.49}$$

where $h(p,q)$ is the cross entropy of two distributions,

$$h(p,q) = \int p(x) \log q(x) dx. \tag{5.50}$$

Because it is not symmetric ($D_{KL}(p\|q) \neq D_{KL}(q\|p)$), the Kullback-Leibler Divergence is sometimes referred to as a pseudo-metric. However, it is non-negative, and equals zero if and only if $p = q$.

As in Section 5.7, the Kullback-Leibler Divergence is approximated with the k-nearest neighbor method advocated in [107]. Let the distributions p and q be represented by a collection of samples of size n and m, respectively. For each sample x_i in p, let $\nu_k(i)$ be the distance to it's k^{th}-nearest neighbor among the remaining samples of p. Furthermore, let $\rho_k(i)$ be the distance between x_i and its k^{th}-nearest neighbor among the samples of q. If either of these distances is zero, the first non-zero neighbor distance is found, yielding a more general notation: $\nu_{k_i}(i)$ and $\rho_{l_i}(i)$, where k_i and l_i are the new neighbor counts and are greather than or equal to k. Then

$$D_{KL}(p\|q) \approx \frac{d}{n} \sum_{i=1}^{n} \left[\log \frac{\nu_{k_i}(i)}{\rho_{l_i}(i)} \right] + \frac{1}{n} \sum_{i=1}^{n} [\psi(l_i) - \psi(k_i)] + \log \frac{m}{n-1}, \tag{5.51}$$

where $\psi(\cdot)$ is the digamma function. In Dakota, k is taken to be six.

The Kullback-Leibler Divergence is used within Dakota to quantify the amount of information gained during Bayesian calibration,

$$IG(f_{\boldsymbol{\Theta}|\boldsymbol{D}}(\boldsymbol{\theta}|\boldsymbol{d}); f_{\boldsymbol{\Theta}}(\boldsymbol{\theta})) = D_{KL}(f_{\boldsymbol{\Theta}|\boldsymbol{D}}(\boldsymbol{\theta}|\boldsymbol{d}) \| f_{\boldsymbol{\Theta}}(\boldsymbol{\theta})). \tag{5.52}$$

If specified in the input file, the approximate value will be output to the screen at the end of the calibration.

In the presence of two (possibly multi-variate) random variables, the mutual information quantifies how much information they contain about each other. In this sense, it is a measure of the mutual dependence of two random variables. For continuous X and Y,

$$I(X,Y) = \iint p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \, dx \, dy, \tag{5.53}$$

where $p(x,y)$ is the joint pdf of X and Y, while $p(x)$ and $p(y)$ are the marginal pdfs of X and Y, respectively. The mutual information is symmetric and non-negative, with zero indicating the independence of X and Y. It is related to the Kullback-Leibler Divergence through the expression

$$I(X,Y) = D_{KL}(p(x,y)\|p(x)p(y)). \tag{5.54}$$

The uses of the mutual information within Dakota have been noted in Section 5.7.

5.9 Measure-theoretic Stochastic Inversion

In this section we present an overview of a specific implementation of the measure-theoretic approach for solving a stochastic inverse problem that incorporates prior information and Bayes' rule to define a unique solution. This approach differs from the standard Bayesian counterpart described in previous sections in that the posterior satisfies a consistency requirement with the model and the observed data. The material in this section is based on the foundational work in [19, 133]. A more thorough description of this consistent Bayesian approach and a comparison with the standard Bayesian approach can be found in [19] and an extension to solve an optimal experimental design problem can be found in [133].

Let $M(Y,\lambda)$ denote a deterministic model with solution $Y(\lambda)$ that is an implicit function of model parameters $\lambda \in \mathbf{\Lambda} \subset \mathbb{R}^n$. The set $\mathbf{\Lambda}$ represents the largest physically meaningful domain of parameter values, and, for simplicity, we assume that $\mathbf{\Lambda}$ is compact. We assume we are only concerned with computing a relatively small set of quantities of interest (QoI), $\{Q_i(Y)\}_{i=1}^m$, where each Q_i is a real-valued functional dependent on the model solution Y. Since Y is a function of parameters λ, so are the QoI and we write $Q_i(\lambda)$ to make this dependence explicit. Given a set of QoI, we define the QoI map $Q(\lambda) := (Q_1(\lambda), \cdots, Q_m(\lambda))^\top : \mathbf{\Lambda} \to \mathcal{D} \subset \mathbb{R}^m$ where $\mathcal{D} := Q(\mathbf{\Lambda})$ denotes the range of the QoI map.

We assume $(\mathbf{\Lambda}, \mathcal{B}_{\mathbf{\Lambda}}, \mu_{\mathbf{\Lambda}})$ and $(\mathcal{D}, \mathcal{B}_{\mathcal{D}}, \mu_{\mathcal{D}})$ are measure spaces and let $\mathcal{B}_{\mathbf{\Lambda}}$ and $\mathcal{B}_{\mathcal{D}}$ denote the Borel σ-algebras inherited from the metric topologies on $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. We also assume that the QoI map Q is at least piecewise smooth implying that Q is a measurable map between the measurable spaces $(\mathbf{\Lambda}, \mathcal{B}_{\mathbf{\Lambda}})$ and $(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$. For any $A \in \mathcal{B}_{\mathcal{D}}$, we then have

$$Q^{-1}(A) = \{\lambda \in \mathbf{\Lambda} \mid Q(\lambda) \in A\} \in \mathcal{B}_{\mathbf{\Lambda}}, \quad \text{and} \quad Q(Q^{-1}(A)) = A.$$

Furthermore, given any $B \in \mathcal{B}_{\mathbf{\Lambda}}$,

$$B \subseteq Q^{-1}(Q(B)), \tag{5.55}$$

although we note that in most cases $B \neq Q^{-1}(Q(B))$ even when $n = m$.

Finally, we assume that an observed probability measure, $P_{\mathcal{D}}^{\text{obs}}$, is given on $(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$, and the stochastic inverse problem seeks a probability measure $P_{\mathbf{\Lambda}}$ such that the subsequent push-forward measure induced by the map, $Q(\lambda)$, satisfies

$$P_{\mathbf{\Lambda}}(Q^{-1}(A)) = P_{\mathcal{D}}^{Q(P_{\mathbf{\Lambda}})}(A) = P_{\mathcal{D}}^{\text{obs}}(A), \tag{5.56}$$

for any $A \in \mathcal{B}_{\mathcal{D}}$.

This inverse problem may not have a unique solution, i.e., there may be multiple probability measures that have the proper push-forward measure. A unique solution may be obtained by imposing additional constraints or structure on the stochastic inverse problem. The approach we consider in this section incorporates prior information and Bayes rule to construct a unique solution to the stochastic inverse problem. The prior probability measure and the map induce a push-forward measure $P_{\mathcal{D}}^{Q(\text{prior})}$ on $\mathcal{D}$, which is defined for all $A \in \mathcal{B}_{\mathcal{D}}$,

$$P_{\mathcal{D}}^{Q(\text{prior})}(A) = P_{\mathbf{\Lambda}}^{\text{prior}}(Q^{-1}(A)). \tag{5.57}$$

We assume that all of the probability measures (prior, observed and push-forward of the prior) are absolutely continuous with respect to some reference measure and can be described in terms of a probability densities. We use $\pi_{\mathbf{\Lambda}}^{\text{prior}}$, $\pi_{\mathcal{D}}^{\text{obs}}$ and $\pi_{\mathcal{D}}^{Q(\text{prior})}$ to denote the probability densities associated with $P_{\mathbf{\Lambda}}^{\text{prior}}$, $P_{\mathcal{D}}^{\text{obs}}$ and $P_{\mathcal{D}}^{Q(\text{prior})}$ respectively. From [19], the following posterior probability density, when interpreted through a disintegration theorem, solves the stochastic inverse problem:

$$\pi_{\mathbf{\Lambda}}^{\text{post}}(\lambda) = \pi_{\mathbf{\Lambda}}^{\text{prior}}(\lambda) \frac{\pi_{\mathcal{D}}^{\text{obs}}(Q(\lambda))}{\pi_{\mathcal{D}}^{Q(\text{prior})}(Q(\lambda))}, \quad \lambda \in \mathbf{\Lambda}. \tag{5.58}$$

One can immediately observe that if $\pi_{\mathcal{D}}^{Q(\text{prior})} = \pi_{\mathcal{D}}^{\text{obs}}$, i.e., if the prior solves the stochastic inverse problem in the sense that the push-forward of the prior matches the observations, then the posterior will be equal to the prior. Approximating this posterior density requires an approximation of the push-forward of the prior, which is simply a forward propagation of uncertainty.

Chapter 6

Surrogate Models

This chapter deals with the theory behind Dakota's surrogate models, which are also known as response surfaces and meta-models.

6.1 Kriging and Gaussian Process Models

In this discussion of Kriging and Gaussian Process (GP) models, vectors are indicated by a single underline and matrices are indicated by a double underline. Capital letters indicate data, or functions of data, that is used to construct an emulator. Lower case letters indicate arbitrary points, i.e. points where the simulator may or may not have been evaluated, and functions of arbitrary points. Estimates, approximations, and models are indicated by hats. For instance, $\hat{f}(\underline{x})$ is a model/emulator of the function $f(\underline{x})$ and $\hat{y}$ is the emulator's prediction or estimate of the true response $y = f(\underline{x})$ evaluated at the point $\underline{x}$. A tilde indicates a reordering of points/equations, with the possible omission of some but not all points. N is the number of points in the sample design and M is the number of input dimensions.

6.1.1 Kriging & Gaussian Processes: Function Values Only

The set of interpolation techniques known as Kriging, also referred to as Gaussian Processes, 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[32]. 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.

A Kriging emulator, $\hat{f}(\underline{x})$, consists of a trend function (frequently a least squares fit to the data, $\underline{g}(\underline{x})^T \underline{\beta}$) plus a Gaussian process error model, $\epsilon(\underline{x})$, that is used to correct the trend function.

$$\hat{f}(\underline{x}) = \underline{g}(\underline{x})^T \underline{\beta} + \epsilon(\underline{x})$$

This represents an estimated distribution for the unknown true surface, $f(\underline{x})$. The error model, $\epsilon(\underline{x})$, makes an adjustment to the trend function so that the emulator will interpolate, and have zero uncertainty at, the data points it was built from. The covariance between the error at two arbitrary points, $\underline{x}$ and $\underline{x'}$, is modeled as

$$\mathrm{Cov}\left(y(\underline{x}), y(\underline{x'})\right) = \mathrm{Cov}\left(\epsilon(\underline{x}), \epsilon(\underline{x'})\right) = \sigma^2 \, r(\underline{x}, \underline{x'}).$$

Here σ^2 is known as the unadjusted variance and $r(\underline{x}, \underline{x'})$ is a correlation function. Measurement error can be modeled explicitly by modifying this to

$$\mathrm{Cov}\left(\epsilon(\underline{x}), \epsilon(\underline{x'})\right) = \sigma^2 \, r(\underline{x}, \underline{x'}) + \Delta^2 \delta(\underline{x} - \underline{x'})$$

where

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

and Δ^2 is the variance of the measurement error. In this work, the term "nugget" refers to the ratio $\eta = \frac{\Delta^2}{\sigma^2}$.

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, $g(\underline{x})\,\beta \equiv 0$. Universal Kriging [93] uses a general polynomial trend model $\underline{g}(\underline{x})^T \, \underline{\beta}$ whose coefficients are determined by least squares regression. Ordinary Kriging is essentially universal Kriging with a trend order of zero, i.e. the trend function is treated as an unknown constant and $g(\underline{x}) = 1$. N_β is the number of basis functions in $\underline{g}(\underline{x})$ and therefore number of elements in the vector $\underline{\beta}$.

For a finite number of sample points, N, there will be uncertainty about the most appropriate value of the vector, $\underline{\beta}$, which can therefore be described as having a distribution of possible values. If one assumes zero prior knowledge about this distribution, which is referred to as the "vague prior" assumption, then the maximum likelihood value of $\underline{\beta}$ can be computed via least squares generalized by the inverse of the error model's correlation matrix, $\underline{\underline{R}}$

$$\underline{\hat{\beta}} = \left(\underline{\underline{G}}^T \underline{\underline{R}}^{-1} \underline{\underline{G}}\right)^{-1} \left(\underline{\underline{G}}^T \underline{\underline{R}}^{-1} \underline{Y}\right).$$

Here $\underline{\underline{G}}$ is a N by N_β matrix that contains the evaluation of the least squares basis functions at all points in $\underline{\underline{X}}$, such that $G_{i,l} = g_l\left(\underline{X_i}\right)$. The real, symmetric, positive-definite correlation matrix, $\underline{\underline{R}}$, of the error model contains evaluations of the correlation function, r, at all pairwise combination of points (rows) in the sample design, $\underline{\underline{X}}$.

$$R_{i,j} = R_{j,i} = r\left(\underline{X_i}, \underline{X_j}\right) = r\left(\underline{X_j}, \underline{X_i}\right)$$

There are many options for r, among them are the following families of correlation functions:

- **Powered-Exponential**

$$r\left(\underline{X_i}, \underline{X_j}\right) = \exp\left(-\sum_{k=1}^{M} \theta_k \left|X_{i,k} - X_{j,k}\right|^\gamma\right) \tag{6.1}$$

  where $0 < \gamma \leq 2$ and $0 < \theta_k$.

- **Matern**

$$r\left(\underline{X_i}, \underline{X_j}\right) = \prod_{k=1}^{M} \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\theta_k \left|X_{i,k} - X_{j,k}\right|\right)^\nu \mathcal{K}_\nu\left(\theta_k \left|X_{i,k} - X_{j,k}\right|\right)$$

  where $0 < \nu$, $0 < \theta_k$, and $\mathcal{K}_\nu(\cdot)$ is the modified Bessel function of order ν; $\nu = s + \frac{1}{2}$ is the smallest value of ν which results in a Kriging model that is s times differentiable.

- **Cauchy**

$$r\left(\underline{X_i}, \underline{X_j}\right) = \prod_{k=1}^{M} \left(1 + \theta_k \left|X_{i,k} - X_{j,k}\right|^{\gamma}\right)^{-\nu}$$

  where $0 < \gamma \leq 2$, $0 < \nu$, and $0 < \theta_k$.

Gneiting et al. [68] provide a more thorough discussion of the properties of and relationships between these three families. Some additional correlation functions include the Dagum family [10] and cubic splines.

The squared exponential or Gaussian correlation function (Equation 6.1 with $\gamma = 2$) was selected to be the first correlation function implemented in Dakota on the basis that its infinite smoothness or differentiability should aid in leveraging the anticipated and hopefully sparse data. For the Gaussian correlation function, the correlation parameters, $\underline{\theta}$, are related to the correlation lengths, $\underline{L}$, by

$$\theta_k = \frac{1}{2\, L_k^2}. \tag{6.2}$$

Here, the correlation lengths, $\underline{L}$, are analogous to standard deviations in the Gaussian or normal distribution and often have physical meaning. The adjusted (by data) mean of the emulator is a best linear unbiased estimator of the unknown true function,

$$\hat{y} = \mathrm{E}\left(\hat{f}\left(\underline{x}\right) | \underline{f}\left(\underline{\underline{X}}\right)\right) = \underline{g}\left(\underline{x}\right)^T \underline{\hat{\beta}} + \underline{r}\left(\underline{x}\right)^T \underline{\underline{R}}^{-1} \underline{\epsilon}. \tag{6.3}$$

Here, $\underline{\epsilon} = \left(\underline{Y} - \underline{\underline{G}}\, \underline{\hat{\beta}}\right)$ is the known vector of differences between the true outputs and trend function at all points in $\underline{\underline{X}}$ and the vector $\underline{r}\left(\underline{x}\right)$ is defined such that $r_i\left(\underline{x}\right) = r\left(\underline{x}, \underline{X_i}\right)$. This correction can be interpreted as the projection of prior belief (the least squares fit) into the span of the data. The adjusted mean of the emulator will interpolate the data that the Kriging model was built from as long as its correlation matrix, $\underline{\underline{R}}$, is numerically non-singular.

Ill-conditioning of $\underline{\underline{R}}$ and other matrices is a recognized as a significant challenge for Kriging. Davis and Morris [35] gave a thorough review of six factors affecting the condition number of matrices associated with Kriging (from the perspective of semivariograms rather than correlation functions). They concluded that "Perhaps the best advice we can give is to be mindful of the condition number when building and solving kriging systems."

In the context of estimating the optimal $\underline{\theta}$, Martin [92] stated that Kriging's "three most prevalent issues are (1) ill-conditioned correlation matrices,(2) multiple local optimum, and (3) long ridges of near optimal values." Because of the second issue, global optimization methods are more robust than local methods. Martin used constrained optimization to address ill-conditioning of $\underline{\underline{R}}$.

Rennen [114] advocated that ill-conditioning be handled by building Kriging models from a uniform subset of available sample points. That option has been available in Dakota's "Gaussian process" model (a separate implementation from Dakota's "Kriging" model) since version 4.1 [52]. Note that Kriging/Gaussian-Process models will not exactly interpolate the discarded points. The implicit justification for this type of approach is that the row or columns of an ill-conditioned matrix contain a significant amount of duplicate information, and that when discarded, duplicate information should be easy to predict.

As of version 5.2, Dakota's `kriging` model has a similar "discard near duplicate points" capability. However, it explicitly addresses the issue of unique information content. Points are **not** discarded prior to the construction

of the Kriging model. Instead, for each vector $\underline{\theta}$ examined that results in an ill-conditioned correlation matrix, $\underline{\underline{R}}$, a pivoted Cholesky factorization of $\underline{\underline{R}}$ is performed. This ranks the points according to how much unique information they contain. Note that the definition of information content depends on $\underline{\theta}$. Low information points are then discarded until $\underline{\underline{R}}$ is no longer ill-conditioned, i.e. until it tightly meets a constraint on condition number. This can be done efficiently using a bisection search that calls LAPACK's fast estimate of the (reciprocal of the) condition number. The possibly, and often, improper subset of points used to construct the Kriging model is the one associated with the chosen $\underline{\theta}$. Methods for selecting $\underline{\theta}$ are discussed below. Since the points that are discarded are the ones that contain the least unique information, they are the ones that are easiest to predict and provide maximum improvement to the condition number.

Adding a nugget, η, to the diagonal entries of $\underline{\underline{R}}$ is a popular approach for both accounting for measurement error in the data and alleviating ill-conditioning. However, doing so will cause the Kriging model to smooth or approximate rather than interpolate the data. Methods for choosing a nugget include:

- Choosing a nugget based on the variance of measurement error (if any); this will be an iterative process if σ^2 is not known in advance.
- Iteratively adding a successively larger nugget until $\underline{\underline{R}} + \eta \underline{\underline{I}}$ is no longer ill-conditioned.
- Exactly calculating the minimum nugget needed for a target condition number from $\underline{\underline{R}}$'s maximum λ_{max} and minimum λ_{min} eigenvalues. The condition number of $\underline{\underline{R}} + \eta \underline{\underline{I}}$ is $\frac{\lambda_{max}+\eta}{\lambda_{min}+\eta}$. However, calculating eigenvalues is computationally expensive. Since Kriging's $\underline{\underline{R}}$ matrix has all ones on the diagonal, its trace and therefore sum of eigenvalues is N. Consequently, a nugget value of $\eta = \frac{N}{\text{target condition number}-1}$ will always alleviate ill-conditioning. A smaller nugget that is also guaranteed to alleviate ill-conditioning can be calculated from LAPACK's fast estimate of the reciprocal of $\underline{\underline{R}}$'s condition number, $\text{rcond}\left(\underline{\underline{R}}\right)$.
- Treating η as another parameter to be selected by the same process used to choose $\underline{\theta}$. Two such approaches are discussed below.

The Kriging model's adjusted variance is commonly used as a spatially varying measure of uncertainty. Knowing where, and by how much, the model "doubts" its own predictions helps build user confidence in the predictions and can be utilized to guide the selection of new sample points during optimization or to otherwise improve the surrogate. The adjusted variance is

$$\begin{aligned}
\text{Var}\left(\hat{y}\right) &= \text{Var}\left(\hat{f}\left(\underline{x}\right) | \underline{f}\left(\underline{\underline{X}}\right)\right) \\
&= \hat{\sigma}^2 \left(1 - \underline{r}\left(\underline{x}\right)^T \underline{\underline{R}}^{-1} \underline{r}\left(\underline{x}\right) + ... \right. \\
&\quad \left. \left(\underline{g}\left(\underline{x}\right)^T - \underline{r}\left(\underline{x}\right)^T \underline{\underline{R}}^{-1} \underline{\underline{G}}\right) \left(\underline{\underline{G}}^T \underline{\underline{R}}^{-1} \underline{\underline{G}}\right)^{-1} \left(\underline{g}\left(\underline{x}\right)^T - \underline{r}\left(\underline{x}\right)^T \underline{\underline{R}}^{-1} \underline{\underline{G}}\right)^T \right)
\end{aligned}$$

where the maximum likelihood estimate of the unadjusted variance is

$$\hat{\sigma}^2 = \frac{\underline{\epsilon}^T \underline{\underline{R}}^{-1} \underline{\epsilon}}{N - N_\beta}.$$

There are two types of numerical approaches to choosing $\underline{\theta}$. One of these is to use Bayesian techniques such as Markov Chain Monte Carlo to obtain a distribution represented by an ensemble of vectors $\underline{\theta}$. In this case, evaluating the emulator's mean involves taking a weighted average of Equation 6.3 over the ensemble of $\underline{\theta}$ vectors.

The other, more common, approach to constructing a Kriging model involves using optimization to find the set of correlation parameters $\underline{\theta}$ that maximizes the likelihood of the model given the data. Dakota's `gaussian_process` and `kriging` models use the maximum likelihood approach. It is equivalent, and more convenient to maximize the natural logarithm of the likelihood, which assuming a vague prior is,

$$\log\left(\text{lik}\left(\underline{\theta}\right)\right) = -\frac{1}{2}\Bigg(\left(N - N_\beta\right)\left(\frac{\hat{\sigma}^2}{\sigma^2} + \log\left(\sigma^2\right) + \log(2\pi)\right) + ... \\ \log\left(\det\left(\underline{\underline{R}}\right)\right) + \log\left(\det\left(\underline{\underline{G}}^T\underline{\underline{R}}^{-1}\underline{\underline{G}}\right)\right)\Bigg).$$

And, if one substitutes the maximum likelihood estimate $\hat{\sigma}^2$ in for σ^2, then it is equivalent to minimize the following objective function

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

Because of the division by $N - N_\beta$, this "per-equation" objective function is mostly independent of the number of sample points, N. It is therefore useful for comparing the (estimated) "goodness" of Kriging models that have different numbers of sample points, e.g. when an arbitrary number of points can be discarded by the pivoted Cholesky approach described above.

Note that the determinant of $\underline{\underline{R}}$ (and $\left(\underline{\underline{G}}^T\underline{\underline{R}}^{-1}\underline{\underline{G}}\right)$) can be efficiently calculated as the square of the product of the diagonals of its Cholesky factorization. However, this will often underflow, i.e. go to zero, making its log incorrectly go to $-\infty$. A more accurate and robust calculation of $\log\left(\det\left(\underline{\underline{R}}\right)\right)$ can be achieved by taking twice the sum of the log of the diagonals of $\underline{\underline{R}}$'s Cholesky factorization.

Also note, that in the absence of constraints, maximizing the likelihood would result in singular $\underline{\underline{R}}$ which makes the emulator incapable of reproducing the data from which it was built. This is because a singular $\underline{\underline{R}}$ makes $\log\left(\det\left(\underline{\underline{R}}\right)\right) = -\infty$ and the *estimate* of likelihood infinite. Constraints are therefore required. Two types of constraints are used in Dakota's `kriging` models.

The first of these is an explicit constraint on LAPACK's fast estimate of the (reciprocal of the) condition number, $2^{-40} < \text{rcond}\left(\underline{\underline{R}}\right)$. The value 2^{-40} was chosen based on the assumption that double precision arithmetic is used. Double precision numbers have 52 bits of precision. Therefore the $2^{-40} < \text{rcond}\left(\det\left(\underline{\underline{R}}\right)\right)$ implies that at least the leading three significant figures should be uncorrupted by round off error. In Dakota 5.2, this constraint is used to determine how many points can be retained in the pivoted Cholesky approach to subset selection described above.

The second, is a box constraint defining a small "feasible" region in correlation length space to search during the maximum likelihood optimization. Many global optimizers, such as the DIRECT (DIvision of RECTangles) used by Dakota's Gaussian Process (as the only option) and Kriging (as the default option) models, require a box constraint definition for the range of acceptable parameters. By default, Dakota's `kriging` model defines the input space to be the smallest hyper-rectangle that contains the sample design. The user has the option to define a larger input space that includes a region where they wish to extrapolate. Note that the emulator can be evaluated at points outside the defined input space, but this definition helps to determine the extent of the "feasible" region of correlation lengths. Let the input space be normalized to the unit hypercube centered at the origin. The average

distance between nearest neighboring points is then

$$d = \left(\frac{1}{N}\right)^{1/M}.$$

Dakota's "feasible" range of correlation lengths, $\underline{L}$, for the Gaussian correlation function is

$$\frac{d}{4} \leq L_k \leq 8d.$$

This range was chosen based on correlation lengths being analogous to the standard deviation in the Gaussian or Normal distribution. If the correlation lengths are set to $L_k = d/4$, then nearest neighboring points "should be" roughly four "standard deviations" away making them almost completely uncorrelated with each other. $\underline{\underline{R}}$ would then be a good approximation of the identity matrix and have a condition number close to one. In the absence of a pathological spacing of points, this range of $\underline{L}$ should contain some non-singular $\underline{\underline{R}}$. $L_k = 8d$ implies approximately 32% trust in what points 8 neighbors away have to say and 5% trust in what points 16 neighbors away have to say. It is possible that the optimal correlation lengths are larger than $8d$; but if so, then either almost all of the same information will be contained in more nearby neighbors, or it was not appropriate to use the squared-exponential/Gaussian correlation function. When other correlation functions are added to the Dakota Kriging implementation, each will be associated with its own range of appropriate correlation lengths chosen by similar reasoning. A different definition of d could be used for non-hypercube input domains.

6.1.2 Gradient Enhanced Kriging

This section focuses on the incorporation of derivative information into Kriging models and challenges in their implementation. Special attention is paid to conditioning issues.

There are at least three basic approaches for incorporating derivative information into Kriging. These are

1. **Indirect**: The sample design is augmented with fictitious points nearby actual sample points which are predicted from derivative information and then a Kriging model is built from the augmented design.

2. **Co-Kriging**: The derivatives with respect to each input variables are treated as separate but correlated output variables and a Co-Kriging model is built for the set of output variables. This would use $\begin{pmatrix} M+2 \\ 2 \end{pmatrix}$ $\underline{\theta}$ vectors.

3. **Direct**: The relationship between the response value and its derivatives is leveraged to use a single $\underline{\theta}$ by assuming

$$\mathrm{Cov}\left(y\left(\underline{x}^1\right), \frac{\partial y\left(\underline{x}^2\right)}{\partial x_k^2}\right) = \frac{\partial}{\partial x_k^2}\left(\mathrm{Cov}\left(y\left(\underline{x}^1\right), y\left(\underline{x}^2\right)\right)\right). \tag{6.4}$$

Dakota 5.2 and later includes an implementation of the direct approach, herein referred to simply as Gradient Enhanced (universal) Kriging (GEK). The equations for GEK can be derived by assuming Equation 6.4 and then taking the same steps used to derive function value only Kriging. The superscript on $\underline{x}$ in Equation 6.4 and below indicates whether it's the 1st or 2nd input to $r\left(\underline{x}^1, \underline{x}^2\right)$. Note that when the first and second arguments are the same, the derivative of $r(\ ,\ )$ with respect to the first argument is equal in magnitude but opposite in sign compared to the derivative with respect to the second argument. The GEK equations can also be obtained by

starting from a Kriging model and making the following substitutions $\underline{Y} \rightarrow \underline{Y_\nabla}$, $\underline{\underline{G}} \rightarrow \underline{\underline{G_\nabla}}$, $\underline{r} \rightarrow \underline{r_\nabla}$, $\underline{\underline{R}} \rightarrow \underline{\underline{R_\nabla}}$, and $N \rightarrow N_\nabla = N\,(1+M)$, where N_∇ is the number of equations rather than the number of points,

$$\underline{Y_\nabla} = \begin{bmatrix} \underline{Y} \\ \frac{\partial \underline{Y}}{\partial X_{:,1}} \\ \frac{\partial \underline{Y}}{\partial X_{:,2}} \\ \vdots \\ \frac{\partial \underline{Y}}{\partial X_{:,M}} \end{bmatrix}, \qquad \underline{\underline{G_\nabla}} = \begin{bmatrix} \underline{\underline{G}} \\ \frac{\partial \underline{\underline{G}}}{\partial X_{:,1}} \\ \frac{\partial \underline{\underline{G}}}{\partial X_{:,2}} \\ \vdots \\ \frac{\partial \underline{\underline{G}}}{\partial X_{:,M}} \end{bmatrix}, \qquad \underline{r_\nabla} = \begin{bmatrix} \underline{r} \\ \frac{\partial \underline{r}}{\partial X_{:,1}} \\ \frac{\partial \underline{r}}{\partial X_{:,2}} \\ \vdots \\ \frac{\partial \underline{r}}{\partial X_{:,M}} \end{bmatrix}$$

$$\underline{\underline{R_\nabla}} = \begin{bmatrix} \underline{\underline{R}} & \frac{\partial \underline{\underline{R}}}{\partial X^2_{:,1}} & \frac{\partial \underline{\underline{R}}}{\partial X^2_{:,2}} & \cdots & \frac{\partial \underline{\underline{R}}}{\partial X^2_{:,M}} \\ \frac{\partial \underline{\underline{R}}}{\partial X^1_{:,1}} & \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,1} \partial X^2_{:,1}} & \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,1} \partial X^2_{:,2}} & \cdots & \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,1} \partial X^2_{:,M}} \\ \frac{\partial \underline{\underline{R}}}{\partial X^1_{:,2}} & \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,2} \partial X^2_{:,1}} & \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,2} \partial X^2_{:,2}} & \cdots & \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,2} \partial X^2_{:,M}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \underline{\underline{R}}}{\partial X^1_{:,M}} & \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,M} \partial X^2_{:,1}} & \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,M} \partial X^2_{:,2}} & \cdots & \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,M} \partial X^2_{:,M}} \end{bmatrix}$$

$$\frac{\partial \underline{\underline{R}}}{\partial X^1_{:,I}} = -\left(\frac{\partial \underline{\underline{R}}}{\partial X^1_{:,I}}\right)^T = -\frac{\partial \underline{\underline{R}}}{\partial X^2_{:,I}} = \left(\frac{\partial \underline{\underline{R}}}{\partial X^2_{:,I}}\right)^T$$

$$\frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,I} \partial X^2_{:,J}} = \left(\frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,I} \partial X^2_{:,J}}\right)^T = \frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,J} \partial X^2_{:,I}} = \left(\frac{\partial^2 \underline{\underline{R}}}{\partial X^1_{:,J} \partial X^2_{:,I}}\right)^T$$

Here capital I and J are scalar indices for the input dimension (column) of the sample design, $\underline{\underline{X}}$. Note that for the Gaussian correlation function

$$\frac{\partial^2 R_{j,j}}{\partial X^1_{j,I} \partial X^2_{j,I}} = 2\theta_I$$

and has units of length^{-2}. Two of the conditions necessary for a matrix to qualify as a correlation matrix are that all of its elements must be dimensionless and all of its diagonal elements must identically equal one. Since $\underline{\underline{R_\nabla}}$ does not satisfy these requirements, it technically does not qualify as a "correlation matrix." However, referring to $\underline{\underline{R_\nabla}}$ as such is a small abuse of terminology and allows GEK to use the same naming conventions as Kriging.

A straight-forward implementation of GEK tends be significantly more accurate than Kriging given the same sample design provided that the

- Derivatives are accurate
- Derivatives are not infinite (or nearly so)

- Function is sufficiently smooth, and
- $\underline{\underline{R_\nabla}}$ is not ill-conditioned (this can be problematic).

If gradients can be obtained cheaply (e.g. by automatic differentiation or adjoint techniques) and the previous conditions are met, GEK also tends to outperform Kriging for the same computational budget. Previous works, such as Dwight[43], state that the direct approach to GEK is significantly better conditioned than the indirect approach. While this is true, (direct) GEK's $\underline{\underline{R_\nabla}}$ matrix can still be, and often is, horribly ill-conditioned compared to Kriging's $\underline{\underline{R}}$ for the same $\underline{\theta}$ and $\underline{\underline{X}}$

In the literature, ill-conditioning is often attributed to the choice of the correlation function. Although a different correlation function may alleviate the ill-conditioning for some problems, the root cause of the ill-conditioning is a poorly spaced sample design. Furthermore, a sufficiently bad sample design could make any interpolatory Kriging model, gradient enhanced or otherwise, ill-conditioned, regardless of the choice of correlation function. This root cause can be addressed directly by discarding points/equations.

Discarding points/equations is conceptually similar to using a Moore-Penrose pseudo inverse of $\underline{\underline{R_\nabla}}$. However, there are important differences. A pseudo inverse handles ill-conditioning by discarding small singular values, which can be interpreted as throwing away the information that is least present while keeping all of what is most frequently duplicated. This causes a Kriging model to not interpolate any of the data points used to construct it while using some information from all rows.

An alternate strategy is to discard additional copies of the information that is most duplicated and keep more of the barely present information. In the context of eigenvalues, this can be described as decreasing the maximum eigenvalue and increasing the minimum eigenvalue by a smaller amount than a pseudo inverse. The result is that the GEK model will exactly fit all of the retained information. This can be achieved using a pivoted Cholesky factorization, such as the one developed by Lucas [90] to determine a reordering $\underline{\underline{\tilde{R}_\nabla}}$ and dropping equations off its end until it tightly meets the constraint on rcond. However, a straight-forward implementation of this is neither efficient nor robust.

In benchmarking tests, Lucas' level 3 pivoted Cholesky implementation was not competitive with the level 3 LAPACK non-pivoted Cholesky in terms of computational efficiency. In some cases, it was an order of magnitude slower. Note that Lucas' level 3 implementation can default to his level 2 implementation and this may explain some of the loss in performance.

More importantly, applying pivoted Cholesky to $\underline{\underline{R_\nabla}}$ tends to sort derivative equations to the top/front and function value equations to the end. This occurred even when $\underline{\underline{R_\nabla}}$ was equilibrated to have ones for all of its diagonal elements. The result was that for many points at least some of the derivative equations were retained while the function values at the same points were discarded. This can (and often did) significantly degrade the accuracy of the GEK predictions. The implication is that derivative equations contain more information than, but are not as reliable as, function value equations.

To address computational efficiency and robustness, Dakota's pivoted Cholesky approach for GEK was modified to:

- Equilibrate $\underline{\underline{R_\nabla}}$ to improve the accuracy of the Cholesky factorization; this is beneficial because $\underline{\underline{R_\nabla}}$ can be poorly scaled. Theorem 4.1 of van der Sluis [131] states that if $\underline{\underline{a}}$ is a real, symmetric, positive definite n by

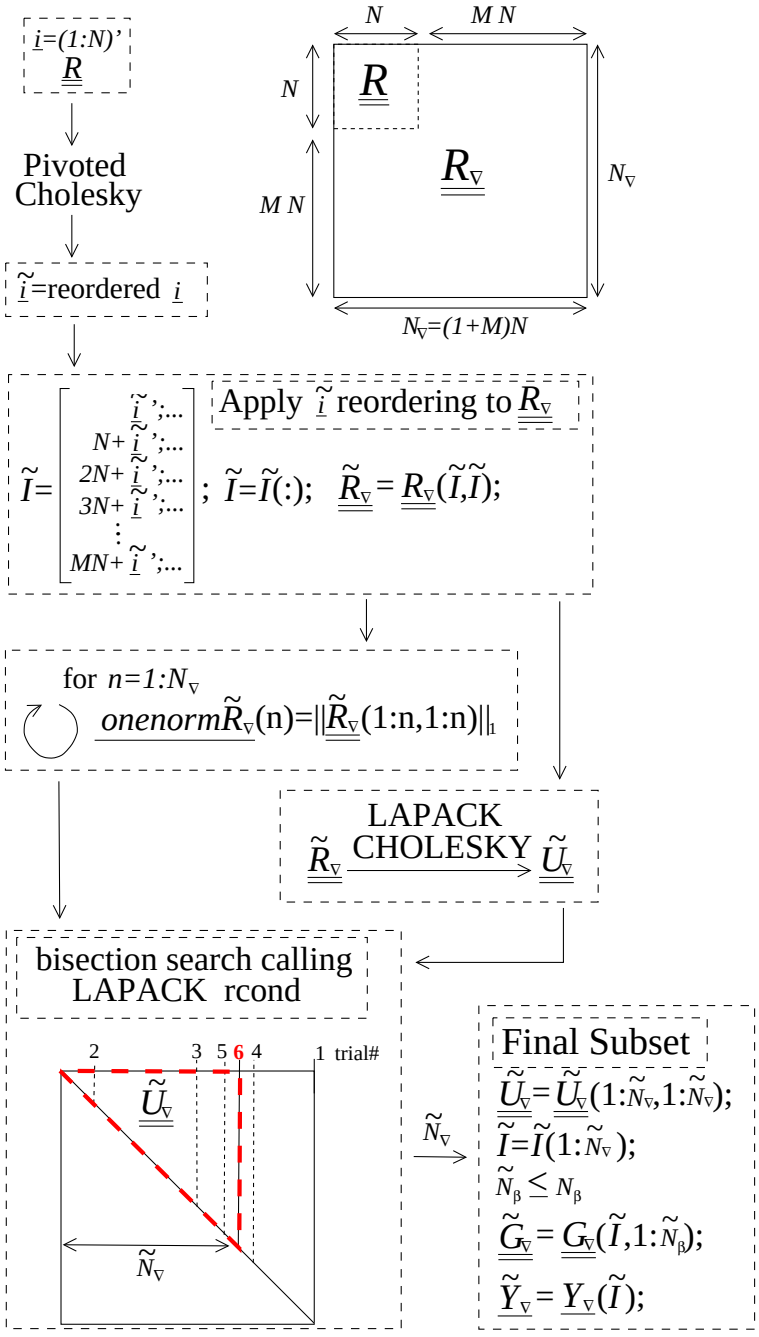


Figure 6.1: A diagram with pseudo code for the pivoted Cholesky algorithm used to select the subset of equations to retain when $\underline{\underline{R}}_\nabla$ is ill-conditioned. Although it is part of the algorithm, the equilibration of $\underline{\underline{R}}_\nabla$ is not shown in this figure. The pseudo code uses MATLAB notation.

n matrix and the diagonal matrix $\underline{\underline{\alpha}}$ contains the square roots of the diagonals of $\underline{\underline{a}}$, then the equilibration

$$\breve{\underline{\underline{a}}} = \underline{\underline{\alpha}}^{-1} \underline{\underline{a}} \, \underline{\underline{\alpha}}^{-1},$$

minimizes the 2-norm condition number of $\breve{\underline{\underline{a}}}$ (with respect to solving linear systems) over all such symmetric scalings, to within a factor of n. The equilibrated matrix $\breve{\underline{\underline{a}}}$ will have all ones on the diagonal.

- Perform pivoted Cholesky on $\underline{\underline{R}}$, instead of $\underline{\underline{R_\nabla}}$, to rank points according to how much new information they contain. This ranking was reflected by the ordering of points in $\tilde{\underline{\underline{R}}}$.
- Apply the ordering of points in $\tilde{\underline{\underline{R}}}$ to whole points in $\underline{\underline{R_\nabla}}$ to produce $\tilde{\underline{\underline{R}}}_\nabla$. Here a whole point means the function value at a point immediately followed by the derivatives at the same point.
- Perform a LAPACK non-pivoted Cholesky on the equilibrated $\tilde{\underline{\underline{R}}}_\nabla$ and drop equations off the end until it satisfies the constraint on rcond. LAPACK's rcond estimate requires the 1-norm of the original (reordered) matrix as input so the 1-norms for all possible sizes of $\tilde{\underline{\underline{R}}}_\nabla$ are precomputed (using a rank one update approach) and stored prior to the Cholesky factorization. A bisection search is used to efficiently determine the number of equations that need to be retained/discarded. This requires $\text{ceil}\left(\log 2 \left(N_\nabla\right)\right)$ or fewer evaluations of rcond. These rcond calls are all based off the same Cholesky factorization of $\tilde{\underline{\underline{R}}}_\nabla$ but use different numbers of rows/columns, $\tilde{N}_\nabla$.

This algorithm is visually depicted in Figure 6.1. Because inverting/factorizing a matrix with n rows and columns requires $\mathcal{O}\left(n^3\right)$ flops, the cost to perform pivoted Cholesky on $\underline{\underline{R}}$ will be much less than, i.e. $\mathcal{O}\left((1+M)^{-3}\right)$, that of $\underline{\underline{R_\nabla}}$ when the number of dimensions M is large. It will also likely be negligible compared to the cost of performing LAPACK's non-pivoted Cholesky on $\tilde{\underline{\underline{R}}}_\nabla$.

6.1.3 Experimental Gaussian Process

The experimental semi-parametric Gaussian process (GP) regression model in Dakota's surrogates module contains two types of variables: hyperparameters $\boldsymbol{\theta}$ that influence the kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ and polynomial trend coefficients $\boldsymbol{\beta}$. Building or fitting a GP involves determining these variables given a data matrix $\boldsymbol{X}$ and vector of response values $\boldsymbol{y}$ by minimizing the negative log marginal likelihood function $J(\boldsymbol{\theta}, \boldsymbol{\beta})$.

We consider an anisotropic squared-exponential kernel augmented with a white noise term, sometimes referred to as a "nugget" or "jitter" in the literature:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left\{ -\frac{1}{2} \sum_{m=1}^{d} \frac{\left(x_m - x'_m\right)^2}{\ell_m^2} \right\} + \eta^2 \delta_{\boldsymbol{x}\boldsymbol{x}'}. \tag{6.5}$$

The parameter σ^2 is a scaling term that affects the range of the output space. The vector $\boldsymbol{\ell} \in \mathbb{R}^d$ contains the correlation lengths for each dimension of the input space. The smaller a given ℓ_m the larger the variation of the function along that dimension. Lastly, the η^2 parameter determines the size of the white noise term.

All of these quantities are strictly positive except for the nugget which is non-negative and will sometimes be omitted or set to a fixed value. These parameters may vary over several orders of magnitude such that variable scaling can have a significant effect on the performance of the optimizer used to find $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$. We perform log-scaling for each of the hyperparameters $\boldsymbol{\theta}$ according to the transformation

$$\theta_i = \log\left(\frac{p_i}{p_{\text{ref}}}\right), \tag{6.6}$$

with $p_{\text{ref}} = 1$ for all parameters for simplicity. Note that we take the original variables to be $\{\sigma, \boldsymbol{\ell}, \eta\}$, not their squares as is done by some authors. The log-transformed version of the kernel is

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp(2\theta_0)\exp\left\{-\frac{1}{2}\sum_{m=1}^{d}(x_m - x'_m)^2\exp(-2\theta_m)\right\} + \exp(2\theta_{-1})\delta_{\boldsymbol{x}\boldsymbol{x}'}, \tag{6.7}$$

where the hyperparameters are ordered so that the log-transformed variables $\sigma = \theta_0$ and $\eta = \theta_{-1}$ are at the beginning and end of the vector of hyperparameters $\boldsymbol{\theta}$, respectively. The polynomial regression coefficients $\boldsymbol{\beta}$ are left unscaled.

An important quantity in Gaussian process regression is the *Gram matrix* $\boldsymbol{G}$:

$$G_{ij} := k(\boldsymbol{x}^i, \boldsymbol{x}^j), \tag{6.8}$$

where if the layout of the surrogate build samples matrix $\boldsymbol{X}$ is number of samples N by dimension of the feature space d the quantities $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ represent the i^{th} and j^{th} rows of this matrix, respectively. The Gram matrix is positive definite provided there are no repeated samples, but it can be ill-conditioned if sample points are close together. Factorization of the (dense) Gram matrix is the computational bottleneck in Gaussian process regression.

The derivatives of the Gram matrix with respect to the hyperparameters are used in the regression procedure. To compute them it is helpful to partition the Gram matrix into two components, one that depends on the squared-exponential piece of the kernel $\boldsymbol{K}$ and another associated with the nugget term

$$\boldsymbol{G} = \boldsymbol{K} + \exp(2\theta_{-1})\boldsymbol{I}. \tag{6.9}$$

Let $\boldsymbol{D}_m$ denote the matrix of component-wise squared distances between points in the space of build points for dimension m:

$$D_{ijm} = \left(x_m^i - x_m^j\right)^2. \tag{6.10}$$

The derivatives of $\boldsymbol{G}$ are

$$\begin{aligned}
\frac{d\boldsymbol{G}}{d\theta_0} &= 2\boldsymbol{K}, \\
\frac{d\boldsymbol{G}}{d\theta_m} &= \exp\left(-2\theta_m\right)\boldsymbol{D}_m \odot \boldsymbol{K}, \quad m = 1\ldots d, \\
\frac{d\boldsymbol{G}}{d\theta_{-1}} &= 2\exp(2\theta_{-1})\boldsymbol{I},
\end{aligned} \tag{6.11}$$

where $\odot$ denotes the Hadamard (i.e. element-wise) product.

A polynomial trend function is obtained by multiplying a basis matrix for the samples $\boldsymbol{\Phi}$ by the trend coefficients $\boldsymbol{\beta}$. The objective function for maximum likelihood estimation J depends on the residual $\boldsymbol{z} := \boldsymbol{y} - \Phi\boldsymbol{\beta}$.

$$J(\boldsymbol{\theta}, \boldsymbol{\beta}) = \frac{1}{2}\log\left(\det\left(\boldsymbol{G}\right)\right) + \frac{1}{2}\boldsymbol{z}^T\boldsymbol{G}^{-1}\boldsymbol{z} + \frac{N}{2}\log\left(2\pi\right). \tag{6.12}$$

The gradient of the objective function can be computed analytically using the derivatives of the Gram matrix. First we introduce

$$\begin{aligned} \boldsymbol{\alpha} &:= \boldsymbol{G}^{-1}\boldsymbol{z}, \\ \boldsymbol{Q} &:= -\frac{1}{2}\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T - \boldsymbol{G}^{-1}\right). \end{aligned} \tag{6.13}$$

The gradient is

$$\begin{aligned} \frac{dJ}{d\theta_0} &= \sum_{i,j} 2Q_{ij}K_{ij}, \\ \frac{dJ}{d\theta_m} &= \sum_{i,j} \left[\boldsymbol{Q} \odot \boldsymbol{K}\right]_{ij} D_{ijm}\exp(-2\theta_m), \quad m = 1, \ldots, d, \\ \frac{dJ}{d\theta_{-1}} &= \text{trace}(2\boldsymbol{Q})\exp(2\theta_{-1}), \\ \frac{dJ}{d\boldsymbol{\beta}} &= -\boldsymbol{\Phi}^T\boldsymbol{\alpha}. \end{aligned} \tag{6.14}$$

We use the Rapid Optimization Library's [84] gradient-based, bound-constrained implementation of the optimization algorithm L-BFGS-B [20] to minimize J. This is a local optimization method so it is typically run multiple times from different initial guesses to increase the chances of finding the global minimum.

Once the regression problem has been solved, we may wish to evaluate the GP at a set of predictions points $\boldsymbol{X}^*$ with an associated basis matrix $\boldsymbol{\Phi}^*$. Some useful quantities are

$$\begin{aligned} P^*_{ij} &= k\left(\left(\boldsymbol{x}^*\right)^i, \boldsymbol{x}^j\right), \\ G^{**}_{ij} &= k\left(\left(\boldsymbol{x}^*\right)^i, \left(\boldsymbol{x}^*\right)^j\right), \\ \boldsymbol{H} &= \boldsymbol{\Phi}^T\boldsymbol{G}^{-1}\boldsymbol{\Phi}, \\ \boldsymbol{R}^* &= \boldsymbol{\Phi}^* - \boldsymbol{P}^*\boldsymbol{G}^{-1}\boldsymbol{\Phi}. \end{aligned} \tag{6.15}$$

The mean and covariance of the GP at the prediction points are

$$\begin{aligned} \boldsymbol{\mu}^* &= \boldsymbol{\Phi}^*\boldsymbol{\beta} + \boldsymbol{P}^*\boldsymbol{\alpha}, \\ \boldsymbol{\Sigma}^{**} &= \boldsymbol{G}^{**} - \boldsymbol{P}^*\boldsymbol{G}^{-1}\left(\boldsymbol{P}^*\right)^T + \boldsymbol{R}^*\boldsymbol{H}^{-1}\left(\boldsymbol{R}^*\right)^T. \end{aligned} \tag{6.16}$$

6.2 Polynomial Models

A preliminary discussion of the surrogate polynomial models available in Dakota is presented in the Surrogate Models Chapter of the User's Manual [3], with select details reproduced here. For ease of notation, the discussion in this section assumes that the model returns a single response $\hat{f}$ and that the design parameters x have dimension n.

For each point x, a linear polynomial model is approximated by

$$\hat{f}(x) \approx c_0 + \sum_{i=1}^{n} c_i x_i, \tag{6.17}$$

a quadratic polynomial model is

$$\hat{f}(x) \approx c_0 + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} \sum_{j \geq i}^{n} c_{ij} x_i x_j, \tag{6.18}$$

and a cubic polynomial model is

$$\hat{f}(x) \approx c_0 + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} \sum_{j \geq i}^{n} c_{ij} x_i x_j + \sum_{i=1}^{n} \sum_{j \geq i}^{n} \sum_{k \geq j}^{n} c_{ijk} x_i x_j x_k. \tag{6.19}$$

In these equations, c_0, c_i, c_{ij}, and c_{ijk} are the polynomial coefficients that are determined during the approximation of the surrogate. Furthermore, each point x corresponds to a vector v_x with entries given by the terms 1, x_i, $x_i x_j$, and $x_i x_j x_k$. The length n_c of this vector corresponds to the number of coefficients present in the polynomial expression. For the linear, quadratic, and cubic polynomials, respectively,

$$n_c = n + 1, \tag{6.20}$$

$$n_c = \frac{(n+1)(n+2)}{2}, \tag{6.21}$$

$$n_c = \frac{n^3 + 6n^2 + 11n + 6}{6}. \tag{6.22}$$

Let the matrix X be such that each row corresponds to the vector of a training point. Thus, given m training points, $X \in \mathbb{R}^{m \times n_c}$. Approximating the polynomial model then amounts to approximating the solution to

$$Xb = Y, \tag{6.23}$$

where b is the vector of the polynomial coefficients described above, and Y contains the true responses $y(x)$. Details regarding the solution or approximate solution of this equation can be found in most mathematical texts and are not discussed here. For any of the training points, the estimated variance is given by

$$\sigma^2 \left(\hat{f}(x) \right) = MSE \left(v_x^T (X^T X)^{-1} v_x \right), \tag{6.24}$$

where MSE is the mean-squared error of the approximation over all of the training points. For any new design parameter, the prediction variance is given by

$$\sigma^2 \left(\hat{f}(x_{new}) \right) = MSE \left(1 + v_{x_{new}}^T (X^T X)^{-1} v_{x_{new}} \right). \tag{6.25}$$

Additional discussion and detail can be found in [96].

Chapter 7

Surrogate-Based Local Minimization

A generally-constrained nonlinear programming problem takes the form

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}_l \leq \mathbf{g}(\mathbf{x}) \leq \mathbf{g}_u \\ & \mathbf{h}(\mathbf{x}) = \mathbf{h}_t \\ & \mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u \end{aligned} \tag{7.1}$$

where $\mathbf{x} \in \Re^n$ is the vector of design variables, and f, $\mathbf{g}$, and $\mathbf{h}$ are the objective function, nonlinear inequality constraints, and nonlinear equality constraints, respectively[1]. Individual nonlinear inequality and equality constraints are enumerated using i and j, respectively (e.g., g_i and h_j). The corresponding surrogate-based optimization (SBO) algorithm may be formulated in several ways and applied to either optimization or least-squares calibration problems. In all cases, SBO solves a sequence of k approximate optimization subproblems subject to a trust region constraint Δ^k; however, many different forms of the surrogate objectives and constraints in the approximate subproblem can be explored. In particular, the subproblem objective may be a surrogate of the original objective or a surrogate of a merit function (most commonly, the Lagrangian or augmented Lagrangian), and the subproblem constraints may be surrogates of the original constraints, linearized approximations of the surrogate constraints, or may be omitted entirely. Each of these combinations is shown in Table 7.1, where black indicates an inappropriate combination, gray indicates an acceptable combination, and blue indicates a common combination.

Initial approaches to nonlinearly-constrained SBO optimized an approximate merit function which incorporated the nonlinear constraints [116, 5]:

$$\begin{aligned} \text{minimize} \quad & \hat{\Phi}^k(\mathbf{x}) \\ \text{subject to} \quad & \| \mathbf{x} - \mathbf{x}_c^k \|_\infty \leq \Delta^k \end{aligned} \tag{7.2}$$

[1] Any linear constraints are not approximated and may be added without modification to all formulations

Table 7.1: SBO approximate subproblem formulations.

	Original Objective	Lagrangian	Augmented Lagrangian
No constraints			TRAL
Linearized constraints		SQP-like	
Original constraints	Direct surrogate		IPTRSAO

where the surrogate merit function is denoted as $\hat{\Phi}(\mathbf{x})$, $\mathbf{x}_c$ is the center point of the trust region, and the trust region is truncated at the global variable bounds as needed. The merit function to approximate was typically chosen to be a standard implementation [132, 101, 66] of the augmented Lagrangian merit function (see Eqs. 7.11–7.12), where the surrogate augmented Lagrangian is constructed from individual surrogate models of the objective and constraints (approximate and assemble, rather than assemble and approximate). In Table 7.1, this corresponds to row 1, column 3, and is known as the trust-region augmented Lagrangian (TRAL) approach. While this approach was provably convergent, convergence rates to constrained minima have been observed to be slowed by the required updating of Lagrange multipliers and penalty parameters [106]. Prior to converging these parameters, SBO iterates did not strictly respect constraint boundaries and were often infeasible. A subsequent approach (IPTRSAO [106]) that sought to directly address this shortcoming added explicit surrogate constraints (row 3, column 3 in Table 7.1):

$$\begin{aligned} \text{minimize} \quad & \hat{\Phi}^k(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}_l \leq \hat{\mathbf{g}}^k(\mathbf{x}) \leq \mathbf{g}_u \\ & \hat{\mathbf{h}}^k(\mathbf{x}) = \mathbf{h}_t \\ & \| \mathbf{x} - \mathbf{x}_c^k \|_\infty \leq \Delta^k \ . \end{aligned} \tag{7.3}$$

While this approach does address infeasible iterates, it still shares the feature that the surrogate merit function may reflect inaccurate relative weightings of the objective and constraints prior to convergence of the Lagrange multipliers and penalty parameters. That is, one may benefit from more feasible intermediate iterates, but the process may still be slow to converge to optimality. The concept of this approach is similar to that of SQP-like SBO approaches [5] which use linearized constraints:

$$\begin{aligned} \text{minimize} \quad & \hat{\Phi}^k(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}_l \leq \hat{\mathbf{g}}^k(\mathbf{x}_c^k) + \nabla\hat{\mathbf{g}}^k(\mathbf{x}_c^k)^T(\mathbf{x} - \mathbf{x}_c^k) \leq \mathbf{g}_u \\ & \hat{\mathbf{h}}^k(\mathbf{x}_c^k) + \nabla\hat{\mathbf{h}}^k(\mathbf{x}_c^k)^T(\mathbf{x} - \mathbf{x}_c^k) = \mathbf{h}_t \\ & \| \mathbf{x} - \mathbf{x}_c^k \|_\infty \leq \Delta^k \ . \end{aligned} \tag{7.4}$$

in that the primary concern is minimizing a composite merit function of the objective and constraints, but under the restriction that the original problem constraints may not be wildly violated prior to convergence of Lagrange multiplier estimates. Here, the merit function selection of the Lagrangian function (row 2, column 2 in Table 7.1; see also Eq. 7.10) is most closely related to SQP, which includes the use of first-order Lagrange multiplier updates (Eq. 7.16) that should converge more rapidly near a constrained minimizer than the zeroth-order updates (Eqs. 7.13-7.14) used for the augmented Lagrangian.

All of these previous constrained SBO approaches involve a recasting of the approximate subproblem objective and constraints as a function of the original objective and constraint surrogates. A more direct approach is to use a formulation of:

$$\begin{aligned} \text{minimize} \quad & \hat{f}^k(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}_l \leq \hat{\mathbf{g}}^k(\mathbf{x}) \leq \mathbf{g}_u \\ & \hat{\mathbf{h}}^k(\mathbf{x}) = \mathbf{h}_t \\ & \| \mathbf{x} - \mathbf{x}_c^k \|_\infty \leq \Delta^k \end{aligned} \tag{7.5}$$

This approach has been termed the direct surrogate approach since it optimizes surrogates of the original objective and constraints (row 3, column 1 in Table 7.1) without any recasting. It is attractive both from its simplicity and potential for improved performance, and is the default approach taken in Dakota. Other Dakota defaults include the use of a filter method for iterate acceptance (see Section 7.1), an augmented Lagrangian merit function (see Section 7.2), Lagrangian hard convergence assessment (see Section 7.3), and no constraint relaxation (see Section 7.4).

Table 7.2: Sample trust region ratio logic.

Ratio Value	Surrogate Accuracy	Iterate Acceptance	Trust Region Sizing
$\rho^k \leq 0$	poor	reject step	shrink
$0 < \rho^k \leq 0.25$	marginal	accept step	shrink
$0.25 < \rho^k < 0.75$ or $\rho^k > 1.25$	moderate	accept step	retain
$0.75 \leq \rho^k \leq 1.25$	good	accept step	expand[2]

While the formulation of Eq. 7.2 (and others from row 1 in Table 7.1) can suffer from infeasible intermediate iterates and slow convergence to constrained minima, each of the approximate subproblem formulations with explicit constraints (Eqs. 7.3-7.5, and others from rows 2-3 in Table 7.1) can suffer from the lack of a feasible solution within the current trust region. Techniques for dealing with this latter challenge involve some form of constraint relaxation. Homotopy approaches [106, 105] or composite step approaches such as Byrd-Omojokun [102], Celis-Dennis-Tapia [21], or MAESTRO [5] may be used for this purpose (see Section 7.4).

After each of the k iterations in the SBO method, the predicted step is validated by computing $f(\mathbf{x}_*^k)$, $\mathbf{g}(\mathbf{x}_*^k)$, and $\mathbf{h}(\mathbf{x}_*^k)$. One approach forms the trust region ratio ρ^k which measures the ratio of the actual improvement to the improvement predicted by optimization on the surrogate model. When optimizing on an approximate merit function (Eqs. 7.2–7.4), the following ratio is natural to compute

$$\rho^k = \frac{\Phi(\mathbf{x}_c^k) - \Phi(\mathbf{x}_*^k)}{\hat{\Phi}(\mathbf{x}_c^k) - \hat{\Phi}(\mathbf{x}_*^k)} \ . \tag{7.6}$$

The formulation in Eq. 7.5 may also form a merit function for computing the trust region ratio; however, the omission of this merit function from explicit use in the approximate optimization cycles can lead to synchronization problems with the optimizer.

Once computed, the value for ρ^k can be used to define the step acceptance and the next trust region size Δ^{k+1} using logic similar to that shown in Table 7.2. Typical factors for shrinking and expanding are 0.5 and 2.0, respectively, but these as well as the threshold ratio values are tunable parameters in the algorithm (see Surrogate-Based Method controls in the Dakota Reference Manual [2]). In addition, the use of discrete thresholds is not required, and continuous relationships using adaptive logic can also be explored [140, 141]. Iterate acceptance or rejection completes an SBO cycle, and the cycles are continued until either soft or hard convergence criteria (see Section 7.3) are satisfied.

7.1 Iterate acceptance logic

When a surrogate optimization is completed and the approximate solution has been validated, then the decision must be made to either accept or reject the step. The traditional approach is to base this decision on the value of the trust region ratio, as outlined previously in Table 7.2. An alternate approach is to utilize a filter method [55], which does not require penalty parameters or Lagrange multiplier estimates. The basic idea in a filter method is to apply the concept of Pareto optimality to the objective function and constraint violations and only accept an iterate if it is not dominated by any previous iterate. Mathematically, a new iterate is not dominated if at least one of the following:

$$\text{either} \quad f < f^{(i)} \quad \text{or} \quad c < c^{(i)} \tag{7.7}$$

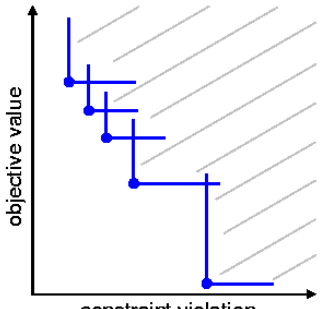


Figure 7.1: Depiction of filter method.

[2] Exception: retain if $\mathbf{x}_*^k$ in trust region interior for design of experiments-based surrogates (global data fits, S-ROM, global E-ROM)

is true for all i in the filter, where c is a selected norm of the constraint violation. This basic description can be augmented with mild requirements to prevent point accumulation and assure convergence, known as a slanting filter [55]. Figure 7.1 illustrates the filter concept, where objective values are plotted against constraint violation for accepted iterates (blue circles) to define the dominated region (denoted by the gray lines). A filter method relaxes the common enforcement of monotonicity in constraint violation reduction and, by allowing more flexibility in acceptable step generation, often allows the algorithm to be more efficient.

The use of a filter method is compatible with any of the SBO formulations in Eqs. 7.2–7.5.

7.2 Merit functions

The merit function $\Phi(\mathbf{x})$ used in Eqs. 7.2-7.4,7.6 may be selected to be a penalty function, an adaptive penalty function, a Lagrangian function, or an augmented Lagrangian function. In each of these cases, the more flexible inequality and equality constraint formulations with two-sided bounds and targets (Eqs. 7.1,7.3-7.5), have been converted to a standard form of $\mathbf{g}(\mathbf{x}) \leq 0$ and $\mathbf{h}(\mathbf{x}) = 0$ (in Eqs. 7.8,7.10-7.16). The active set of inequality constraints is denoted as $\mathbf{g}^+$.

The penalty function employed in this paper uses a quadratic penalty with the penalty schedule linked to SBO iteration number

$$\Phi(\mathbf{x}, r_p) = f(\mathbf{x}) + r_p \mathbf{g}^+(\mathbf{x})^T \mathbf{g}^+(\mathbf{x}) + r_p \mathbf{h}(\mathbf{x})^T \mathbf{h}(\mathbf{x}) \tag{7.8}$$

$$r_p = e^{(k+\text{offset})/10} \tag{7.9}$$

The adaptive penalty function is identical in form to Eq. 7.8, but adapts r_p using monotonic increases in the iteration offset value in order to accept any iterate that reduces the constraint violation.

The Lagrangian merit function is

$$\Phi(\mathbf{x}, \boldsymbol{\lambda}_g, \boldsymbol{\lambda}_h) = f(\mathbf{x}) + \boldsymbol{\lambda}_g^T \mathbf{g}^+(\mathbf{x}) + \boldsymbol{\lambda}_h^T \mathbf{h}(\mathbf{x}) \tag{7.10}$$

for which the Lagrange multiplier estimation is discussed in Section 7.3. Away from the optimum, it is possible for the least squares estimates of the Lagrange multipliers for active constraints to be zero, which equates to omitting the contribution of an active constraint from the merit function. This is undesirable for tracking SBO progress, so usage of the Lagrangian merit function is normally restricted to approximate subproblems and hard convergence assessments.

The augmented Lagrangian employed in this paper follows the sign conventions described in [132]

$$\Phi(\mathbf{x}, \boldsymbol{\lambda}_\psi, \boldsymbol{\lambda}_h, r_p) = f(\mathbf{x}) + \boldsymbol{\lambda}_\psi^T \boldsymbol{\psi}(\mathbf{x}) + r_p \boldsymbol{\psi}(\mathbf{x})^T \boldsymbol{\psi}(\mathbf{x}) + \boldsymbol{\lambda}_h^T \mathbf{h}(\mathbf{x}) + r_p \mathbf{h}(\mathbf{x})^T \mathbf{h}(\mathbf{x}) \tag{7.11}$$

$$\psi_i = \max\left\{ g_i, -\frac{\lambda_{\psi_i}}{2r_p} \right\} \tag{7.12}$$

where $\boldsymbol{\psi}(\mathbf{x})$ is derived from the elimination of slack variables for the inequality constraints. In this case, simple zeroth-order Lagrange multiplier updates may be used:

$$\boldsymbol{\lambda}_\psi^{k+1} = \boldsymbol{\lambda}_\psi^k + 2r_p \boldsymbol{\psi}(\mathbf{x}) \tag{7.13}$$

$$\boldsymbol{\lambda}_h^{k+1} = \boldsymbol{\lambda}_h^k + 2r_p \mathbf{h}(\mathbf{x}) \tag{7.14}$$

The updating of multipliers and penalties is carefully orchestrated [24] to drive reduction in constraint violation of the iterates. The penalty updates can be more conservative than in Eq. 7.9, often using an infrequent application of a constant multiplier rather than a fixed exponential progression.

7.3 Convergence assessment

To terminate the SBO process, hard and soft convergence metrics are monitored. It is preferable for SBO studies to satisfy hard convergence metrics, but this is not always practical (e.g., when gradients are unavailable or unreliable). Therefore, simple soft convergence criteria are also employed which monitor for diminishing returns (relative improvement in the merit function less than a tolerance for some number of consecutive iterations).

To assess hard convergence, one calculates the norm of the projected gradient of a merit function whenever the feasibility tolerance is satisfied. The best merit function for this purpose is the Lagrangian merit function from Eq. 7.10. This requires a least squares estimation for the Lagrange multipliers that best minimize the projected gradient:

$$\nabla_x \Phi(\mathbf{x}, \boldsymbol{\lambda}_g, \boldsymbol{\lambda}_h) = \nabla_x f(\mathbf{x}) + \boldsymbol{\lambda}_g^T \nabla_x \mathbf{g}^+(\mathbf{x}) + \boldsymbol{\lambda}_h^T \nabla_x \mathbf{h}(\mathbf{x}) \tag{7.15}$$

where gradient portions directed into active global variable bounds have been removed. This can be posed as a linear least squares problem for the multipliers:

$$\mathbf{A}\boldsymbol{\lambda} = -\nabla_x f \tag{7.16}$$

where $\mathbf{A}$ is the matrix of active constraint gradients, $\boldsymbol{\lambda}_g$ is constrained to be non-negative, and $\boldsymbol{\lambda}_h$ is unrestricted in sign. To estimate the multipliers using non-negative and bound-constrained linear least squares, the NNLS and BVLS routines [88] from NETLIB are used, respectively.

7.4 Constraint relaxation

The goal of constraint relaxation is to achieve efficiency through the balance of feasibility and optimality when the trust region restrictions prevent the location of feasible solutions to constrained approximate subproblems (Eqs. 7.3-7.5, and other formulations from rows 2-3 in Table 7.1). The SBO algorithm starting from infeasible points will commonly generate iterates which seek to satisfy feasibility conditions without regard to objective reduction [105].

One approach for achieving this balance is to use *relaxed constraints* when iterates are infeasible with respect to the surrogate constraints. We follow Perez, Renaud, and Watson [106], and use a *global homotopy* mapping the relaxed constraints and the surrogate constraints. For formulations in Eqs. 7.3 and 7.5 (and others from row 3 in Table 7.1), the relaxed constraints are defined from

$$\tilde{\mathbf{g}}^k(\mathbf{x}, \tau) = \hat{\mathbf{g}}^k(\mathbf{x}) + (1-\tau)\mathbf{b}_g \tag{7.17}$$

$$\tilde{\mathbf{h}}^k(\mathbf{x}, \tau) = \hat{\mathbf{h}}^k(\mathbf{x}) + (1-\tau)\mathbf{b}_h \tag{7.18}$$

For Eq. 7.4 (and others from row 2 in Table 7.1), the original surrogate constraints $\hat{\mathbf{g}}^k(\mathbf{x})$ and $\hat{\mathbf{h}}^k(\mathbf{x})$ in Eqs. 7.17-7.18 are replaced with their linearized forms ($\hat{\mathbf{g}}^k(\mathbf{x}_c^k) + \nabla\hat{\mathbf{g}}^k(\mathbf{x}_c^k)^T(\mathbf{x} - \mathbf{x}_c^k)$ and $\hat{\mathbf{h}}^k(\mathbf{x}_c^k) + \nabla\hat{\mathbf{h}}^k(\mathbf{x}_c^k)^T(\mathbf{x} - \mathbf{x}_c^k)$, respectively). The approximate subproblem is then reposed using the relaxed constraints as

$$\begin{aligned} \text{minimize} \quad & \hat{f}^k(\mathbf{x}) \text{ or } \hat{\Phi}^k(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}_l \leq \tilde{\mathbf{g}}^k(\mathbf{x}, \tau^k) \leq \mathbf{g}_u \\ & \tilde{\mathbf{h}}^k(\mathbf{x}, \tau^k) = \mathbf{h}_t \\ & \| \mathbf{x} - \mathbf{x}_c^k \|_\infty \leq \Delta^k \end{aligned} \tag{7.19}$$

in place of the corresponding subproblems in Eqs. 7.3-7.5. Alternatively, since the relaxation terms are constants for the k^{th} iteration, it may be more convenient for the implementation to constrain $\hat{\mathbf{g}}^k(\mathbf{x})$ and $\hat{\mathbf{h}}^k(\mathbf{x})$ (or their

linearized forms) subject to relaxed bounds and targets ($\tilde{\mathbf{g}}_l^k$, $\tilde{\mathbf{g}}_u^k$, $\tilde{\mathbf{h}}_t^k$). The parameter τ is the homotopy parameter controlling the extent of the relaxation: when $\tau = 0$, the constraints are fully relaxed, and when $\tau = 1$, the surrogate constraints are recovered. The vectors $\mathbf{b}_g, \mathbf{b}_h$ are chosen so that the starting point, $\mathbf{x}^0$, is feasible with respect to the fully relaxed constraints:

$$\mathbf{g}_l \leq \tilde{\mathbf{g}}^0(\mathbf{x}^0, 0) \leq \mathbf{g}_u \tag{7.20}$$

$$\tilde{\mathbf{h}}^0(\mathbf{x}^0, 0) = \mathbf{h}_t \tag{7.21}$$

At the start of the SBO algorithm, $\tau^0 = 0$ if $\mathbf{x}^0$ is infeasible with respect to the unrelaxed surrogate constraints; otherwise $\tau^0 = 1$ (i.e., no constraint relaxation is used). At the start of the k^{th} SBO iteration where $\tau^{k-1} < 1$, τ^k is determined by solving the subproblem

$$\begin{aligned} \text{maximize} \quad & \tau^k \\ \text{subject to} \quad & \mathbf{g}_l \leq \tilde{\mathbf{g}}^k(\mathbf{x}, \tau^k) \leq \mathbf{g}_u \\ & \tilde{\mathbf{h}}^k(\mathbf{x}, \tau^k) = \mathbf{h}_t \\ & \| \mathbf{x} - \mathbf{x}_c^k \|_\infty \leq \Delta^k \\ & \tau^k \geq 0 \end{aligned} \tag{7.22}$$

starting at $(\mathbf{x}_*^{k-1}, \tau^{k-1})$, and then adjusted as follows:

$$\tau^k = \min\left\{1, \tau^{k-1} + \alpha\left(\tau_{\max}^k - \tau^{k-1}\right)\right\} \tag{7.23}$$

The adjustment parameter $0 < \alpha < 1$ is chosen so that that the feasible region with respect to the relaxed constraints has positive volume within the trust region. Determining the optimal value for α remains an open question and will be explored in future work.

After τ^k is determined using this procedure, the problem in Eq. 7.19 is solved for $\mathbf{x}_*^k$. If the step is accepted, then the value of τ^k is updated using the current iterate $\mathbf{x}_*^k$ and the validated constraints $\mathbf{g}(\mathbf{x}_*^k)$ and $\mathbf{h}(\mathbf{x}_*^k)$:

$$\tau^k = \min\left\{1, \min_i \tau_i, \min_j \tau_j\right\} \tag{7.24}$$

$$\text{where } \tau_i = 1 + \frac{\min\left\{g_i(\mathbf{x}_*^k) - gl_i, gu_i - g_i(\mathbf{x}_*^k)\right\}}{b_{g_i}} \tag{7.25}$$

$$\tau_j = 1 - \frac{|h_j(\mathbf{x}_*^k) - h_{t_j}|}{b_{h_j}} \tag{7.26}$$

Figure 7.2 illustrates the SBO algorithm on a two-dimensional problem with one inequality constraint starting from an infeasible point, $\mathbf{x}^0$. The minimizer of the problem is denoted as $\mathbf{x}^*$. Iterates generated using the surrogate constraints are shown in red, where feasibility is achieved first, and then progress is made toward the optimal point. The iterates generated using the relaxed constraints are shown in blue, where a balance of satisfying feasibility and optimality has been achieved, leading to fewer overall SBO iterations.

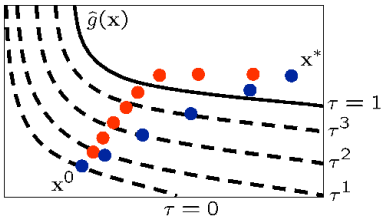


Figure 7.2: Illustration of SBO iterates using surrogate (red) and relaxed (blue) constraints.

The behavior illustrated in Fig. 7.2 is an example where using the relaxed constraints over the surrogate constraints may improve the overall performance of the SBO algorithm by reducing the number of iterations performed. This improvement comes at the cost of solving the minimization subproblem in Eq. 7.22, which can be significant in some cases (i.e., when the cost of evaluating $\hat{\mathbf{g}}^k(\mathbf{x})$ and $\hat{\mathbf{h}}^k(\mathbf{x})$ is not negligible, such as with multifidelity or ROM surrogates). As shown in the numerical experiments involving the Barnes

problem presented in [106], the directions toward constraint violation reduction and objective function reduction may be in opposing directions. In such cases, the use of the relaxed constraints may result in an *increase* in the overall number of SBO iterations since feasibility must ultimately take precedence.

Chapter 8

Effcient Global Optimization

Efficient Global Optimization (EGO) was developed to facilitate the unconstrained minimization of expensive implicit response functions. The method builds an initial Gaussian process model as a global surrogate for the response function, then intelligently selects additional samples to be added for inclusion in a new Gaussian process model in subsequent iterations. The new samples are selected based on how much they are expected to improve the current best solution to the optimization problem. When this expected improvement is acceptably small, the globally optimal solution has been found. The application of this methodology to equality-constrained reliability analysis is the primary contribution of EGRA.

Efficient global optimization was originally proposed by Jones et al. [79] and has been adapted into similar methods such as sequential kriging optimization (SKO) [77]. The main difference between SKO and EGO lies within the specific formulation of what is known as the expected improvement function (EIF), which is the feature that sets all EGO/SKO-type methods apart from other global optimization methods. The EIF is used to select the location at which a new training point should be added to the Gaussian process model by maximizing the amount of improvement in the objective function that can be expected by adding that point. A point could be expected to produce an improvement in the objective function if its predicted value is better than the current best solution, or if the uncertainty in its prediction is such that the probability of it producing a better solution is high. Because the uncertainty is higher in regions of the design space with fewer observations, this provides a balance between exploiting areas of the design space that predict good solutions, and exploring areas where more information is needed.

The general procedure of these EGO-type methods is:

1. Build an initial Gaussian process model of the objective function.

2. Find the point that maximizes the EIF. If the EIF value at this point is sufficiently small, stop.

3. Evaluate the objective function at the point where the EIF is maximized. Update the Gaussian process model using this new point. Go to Step 2.

The following sections discuss the construction of the Gaussian process model used, the form of the EIF, and then a description of how that EIF is modified for application to reliability analysis.

8.1 Gaussian Process Model

Gaussian process (GP) models are set apart from other surrogate models because they provide not just a predicted value at an unsampled point, but also and estimate of the prediction variance. This variance gives an indication of the uncertainty in the GP model, which results from the construction of the covariance function. This function is based on the idea that when input points are near one another, the correlation between their corresponding outputs will be high. As a result, the uncertainty associated with the model's predictions will be small for input points which are near the points used to train the model, and will increase as one moves further from the training points.

It is assumed that the true response function being modeled $G(\mathbf{u})$ can be described by: [32]

$$G(\mathbf{u}) = \mathbf{h}(\mathbf{u})^T \boldsymbol{\beta} + Z(\mathbf{u}) \tag{8.1}$$

where $\mathbf{h}()$ is the trend of the model, $\boldsymbol{\beta}$ is the vector of trend coefficients, and $Z()$ is a stationary Gaussian process with zero mean (and covariance defined below) that describes the departure of the model from its underlying trend. The trend of the model can be assumed to be any function, but taking it to be a constant value has been reported to be generally sufficient. [118] For the work presented here, the trend is assumed constant and $\boldsymbol{\beta}$ is taken as simply the mean of the responses at the training points. The covariance between outputs of the Gaussian process $Z()$ at points $\mathbf{a}$ and $\mathbf{b}$ is defined as:

$$Cov\left[Z(\mathbf{a}), Z(\mathbf{b})\right] = \sigma_Z^2 R(\mathbf{a}, \mathbf{b}) \tag{8.2}$$

where σ_Z^2 is the process variance and $R()$ is the correlation function. There are several options for the correlation function, but the squared-exponential function is common [118], and is used here for $R()$:

$$R(\mathbf{a}, \mathbf{b}) = \exp\left[-\sum_{i=1}^{d} \theta_i (a_i - b_i)^2\right] \tag{8.3}$$

where d represents the dimensionality of the problem (the number of random variables), and θ_i is a scale parameter that indicates the correlation between the points within dimension i. A large θ_i is representative of a short correlation length.

The expected value $\mu_G()$ and variance $\sigma_G^2()$ of the GP model prediction at point $\mathbf{u}$ are:

$$\mu_G(\mathbf{u}) = \mathbf{h}(\mathbf{u})^T \boldsymbol{\beta} + \mathbf{r}(\mathbf{u})^T \mathbf{R}^{-1}(\mathbf{g} - \mathbf{F}\boldsymbol{\beta}) \tag{8.4}$$

$$\sigma_G^2(\mathbf{u}) = \sigma_Z^2 - \begin{bmatrix}\mathbf{h}(\mathbf{u})^T & \mathbf{r}(\mathbf{u})^T\end{bmatrix} \begin{bmatrix}\mathbf{0} & \mathbf{F}^T \\ \mathbf{F} & \mathbf{R}\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{h}(\mathbf{u}) \\ \mathbf{r}(\mathbf{u})\end{bmatrix} \tag{8.5}$$

where $\mathbf{r}(\mathbf{u})$ is a vector containing the covariance between $\mathbf{u}$ and each of the n training points (defined by Eq. 8.2), $\mathbf{R}$ is an $n \times n$ matrix containing the correlation between each pair of training points, $\mathbf{g}$ is the vector of response outputs at each of the training points, and $\mathbf{F}$ is an $n \times q$ matrix with rows $\mathbf{h}(\mathbf{u}_i)^T$ (the trend function for training point i containing q terms; for a constant trend $q=1$). This form of the variance accounts for the uncertainty in the trend coefficients $\boldsymbol{\beta}$, but assumes that the parameters governing the covariance function (σ_Z^2 and $\boldsymbol{\theta}$) have known values.

The parameters σ_Z^2 and $\boldsymbol{\theta}$ are determined through maximum likelihood estimation. This involves taking the log of the probability of observing the response values $\mathbf{g}$ given the covariance matrix $\mathbf{R}$, which can be written as: [118]

$$\log\left[p(\mathbf{g}|\mathbf{R})\right] = -\frac{1}{n}\log|\mathbf{R}| - \log(\hat{\sigma}_Z^2) \tag{8.6}$$

where $|\mathbf{R}|$ indicates the determinant of $\mathbf{R}$, and $\hat{\sigma}_Z^2$ is the optimal value of the variance given an estimate of $\boldsymbol{\theta}$ and is defined by:

$$\hat{\sigma}_Z^2 = \frac{1}{n}(\mathbf{g} - \mathbf{F}\boldsymbol{\beta})^T \mathbf{R}^{-1}(\mathbf{g} - \mathbf{F}\boldsymbol{\beta}) \tag{8.7}$$

Maximizing Eq. 8.6 gives the maximum likelihood estimate of $\boldsymbol{\theta}$, which in turn defines σ_Z^2.

8.2 Acquisition Functions

The acquisition function determines the location of the next sampling point or refinement points, in the sense that maximizing the acquisition function yields the next sampling point, as

$$\mathbf{u}^* = \arg\max_{\mathbf{u}} a(\mathbf{u}). \tag{8.8}$$

8.2.1 Expected Improvement

The expected improvement function is used to select the location at which a new training point should be added. The EIF is defined as the expectation that any point in the search space will provide a better solution than the current best solution based on the expected values and variances predicted by the GP model. An important feature of the EIF is that it provides a balance between exploiting areas of the design space where good solutions have been found, and exploring areas of the design space where the uncertainty is high. First, recognize that at any point in the design space, the GP prediction $\hat{G}()$ is a Gaussian distribution:

$$\hat{G}(\mathbf{u}) \sim \mathcal{N}\left(\mu_G(\mathbf{u}), \sigma_G(\mathbf{u})\right) \tag{8.9}$$

where the mean $\mu_G()$ and the variance $\sigma^2_G()$ were defined in Eqs. 8.4 and 8.5, respectively. The EIF is defined as: [79]

$$EI\big(\hat{G}(\mathbf{u})\big) \equiv E\left[\max\Big(G(\mathbf{u}^*) - \hat{G}(\mathbf{u}), 0\Big)\right] \tag{8.10}$$

where $G(\mathbf{u}^*)$ is the current best solution chosen from among the true function values at the training points (henceforth referred to as simply G^*). This expectation can then be computed by integrating over the distribution $\hat{G}(\mathbf{u})$ with G^* held constant:

$$EI\big(\hat{G}(\mathbf{u})\big) = \int_{-\infty}^{G^*} (G^* - G)\ \hat{G}(\mathbf{u})\ dG \tag{8.11}$$

where G is a realization of $\hat{G}$. This integral can be expressed analytically as: [79]

$$EI\big(\hat{G}(\mathbf{u})\big) = (G^* - \mu_G)\ \Phi\left(\frac{G^* - \mu_G}{\sigma_G}\right) + \sigma_G\ \phi\left(\frac{G^* - \mu_G}{\sigma_G}\right) \tag{8.12}$$

where it is understood that μ_G and σ_G are functions of $\mathbf{u}$. Rewritting in a more compact manner and dropping the subscript $_G$,

$$a_{\text{EI}}(\mathbf{u}, \{\mathbf{u}_i, y_i\}_{i=1}^N, \theta)) = \sigma(\mathbf{u}) \cdot (\gamma(\mathbf{u})\Phi(\gamma(\mathbf{u})) + \phi(\gamma(\mathbf{u}))), \tag{8.13}$$

where $\gamma(\mathbf{u}) = \frac{G^* - \mu(\mathbf{u})}{\sigma(\mathbf{u})}$. This equation defines the expected improvement acquisition function for an unknown $\mathbf{u}$.

The point at which the EIF is maximized is selected as an additional training point. With the new training point added, a new GP model is built and then used to construct another EIF, which is then used to choose another new training point, and so on, until the value of the EIF at its maximized point is below some specified tolerance. In Ref. [77] this maximization is performed using a Nelder-Mead simplex approach, which is a local optimization method. Because the EIF is often highly multimodal [79] it is expected that Nelder-Mead may fail to converge to the true global optimum. In Ref. [79], a branch-and-bound technique for maximizing the EIF is used, but was found to often be too expensive to run to convergence. In Dakota, an implementation of the DIRECT global optimization algorithm is used [57].

It is important to understand how the use of this EIF leads to optimal solutions. Eq. 8.12 indicates how much the objective function value at $\mathbf{x}$ is expected to be less than the predicted value at the current best solution. Because the GP model provides a Gaussian distribution at each predicted point, expectations can be calculated. Points with

good expected values and even a small variance will have a significant expectation of producing a better solution (exploitation), but so will points that have relatively poor expected values and greater variance (exploration).

The application of EGO to reliability analysis, however, is made more complicated due to the inclusion of equality constraints (see Eqs. 2.16-2.17). For inverse reliability analysis, this extra complication is small. The response being modeled by the GP is the objective function of the optimization problem (see Eq. 2.17) and the deterministic constraint might be handled through the use of a merit function, thereby allowing EGO to solve this equality-constrained optimization problem. Here the problem lies in the interpretation of the constraint for multimodal problems as mentioned previously. In the forward reliability case, the response function appears in the constraint rather than the objective. Here, the maximization of the EIF is inappropriate because feasibility is the main concern. This application is therefore a significant departure from the original objective of EGO and requires a new formulation. For this problem, the expected feasibility function is introduced.

8.2.2 Probability Improvement Acquisition Function

The probability of improvement (PI) acquisition function is proposed by [86], using the same argument that the GP prediction is a Gaussian distribution. Similar to Equation 8.13, the PI acquisition function is given by

$$a_{\text{PI}}(\mathbf{u}) = \Phi(\gamma(\mathbf{u})). \tag{8.14}$$

Generally speaking, the EI acquisition function performs better than the PI acquisition function.

8.2.3 Lower-Confidence Bound Acquisition Function

Another form of acquisition is lower-confidence bound (LCB), proposed recently by Srinivas et al. [122, 123], which has shown to perform very well. The LCB acquisition function takes the form of

$$a_{\text{LCB}}(\mathbf{u}) = -\mu(\mathbf{u}) + \kappa\sigma(\mathbf{u}), \tag{8.15}$$

where κ is a hyper-parameter describing the acquisition exploitation-exploration balance. In many cases in design optimization, $\kappa = 2$ is preferred, but relaxing this κ as a function of iterations is also possible, cf. Daniel et al. [33], as

$$\kappa = \sqrt{\nu\gamma_n}, \quad \nu = 1, \quad \gamma_n = 2\log\left(\frac{N^{d/2+2}\pi^2}{3\delta}\right), \tag{8.16}$$

and d is the dimensionality of the problem, and $\delta \in (0, 1)$ [123].

8.3 Batch-sequential parallel

The batch-sequential parallelization is mainly motivated by exploiting the computational resource, where multiple sampling point $\mathbf{u}$ can be queried concurrently on a high-performance computing platform. The benefit of batch implementation is that the physical time to converge to the optimal solution is significantly reduced with a factor of $\sqrt{K}$, where K is the batch size. While there are many flavors of batch-sequential parallelization, as well as asynchronous parallelization in EGO and Bayesian optimization, we mainly review the theory of GP-BUCB by Desautels et al. [37], GP-UCB-PE by Contal et al [30], and pBO-2GP-3B by Tran et al [129]. The parallelization feature of EGO is sometimes referred to as lookahead or non-myopic Bayesian optimization in the literature, especially in the machine learning community.

The approach by Desautels et al. [37] mainly advocates for the "hallucination" scheme or heuristic liar, in which the unknown observation at the currently querying sampling point $\mathbf{u}^*$ is *temporarily* assumed as the posterior mean $\mu(\mathbf{u}^*)$. Then, the underlying GP model updates based on this assumption and locates other points in the same batch, until the batch is filled. After the whole batch is constructed, it is then queried, and all the responses are received at once when the batch is completed. Contal et al. [30] extended from the work of Desautels et al. [37] and proved that including pure exploration (i.e. sampling at $\mathbf{u}^*$ where $\sigma(\mathbf{u})$ is maximum) increases the efficiency. Tran et al. [129] adopted two aforementioned approaches and extended for known and unknown constraints.

Chapter 9

Dimension Reduction Strategies

In this section dimension reduction strategies are introduced. All dimension reduction strategies are based on the idea of finding the important directions in the original input space in order to approximate the response on a lower dimensional space. Once a lower dimensional space is identified, several UQ strategies can be deployed on it making the UQ studies less computational expensive.

In the following two approaches are introduced, namely the Active Subspace method [25] and the Basis Adaptation [128].

9.1 Active Subspace Models

The idea behind active subspaces is to find directions in the input variable space in which the quantity of interest is nearly constant. After rotation of the input variables, this method can allow significant dimension reduction. Below is a brief summary of the process.

1. Compute the gradient of the quantity of interest, $q = f(\mathbf{x})$, at several locations sampled from the full input space,
$$\nabla_{\mathbf{x}} f_i = \nabla f(\mathbf{x}_i).$$

2. Compute the eigendecomposition of the matrix $\hat{\mathbf{C}}$,
$$\hat{\mathbf{C}} = \frac{1}{M}\sum_{i=1}^{M} \nabla_{\mathbf{x}} f_i \nabla_{\mathbf{x}} f_i^T = \hat{\mathbf{W}}\hat{\mathbf{\Lambda}}\hat{\mathbf{W}}^T,$$
where $\hat{\mathbf{W}}$ has eigenvectors as columns, $\hat{\mathbf{\Lambda}} = \text{diag}(\hat{\lambda}_1, \ \ldots \ , \hat{\lambda}_N)$ contains eigenvalues, and N is the total number of parameters.

3. Using a truncation method or specifying a dimension to estimate the active subspace size, split the eigenvectors into active and inactive directions,
$$\hat{\mathbf{W}} = \begin{bmatrix} \hat{\mathbf{W}}_1 & \hat{\mathbf{W}}_2 \end{bmatrix}.$$
These eigenvectors are used to rotate the input variables.

4. Next the input variables, $\mathbf{x}$, are expanded in terms of active and inactive variables,

$$\mathbf{x} = \hat{\mathbf{W}}_1\mathbf{y} + \hat{\mathbf{W}}_2\mathbf{z}.$$

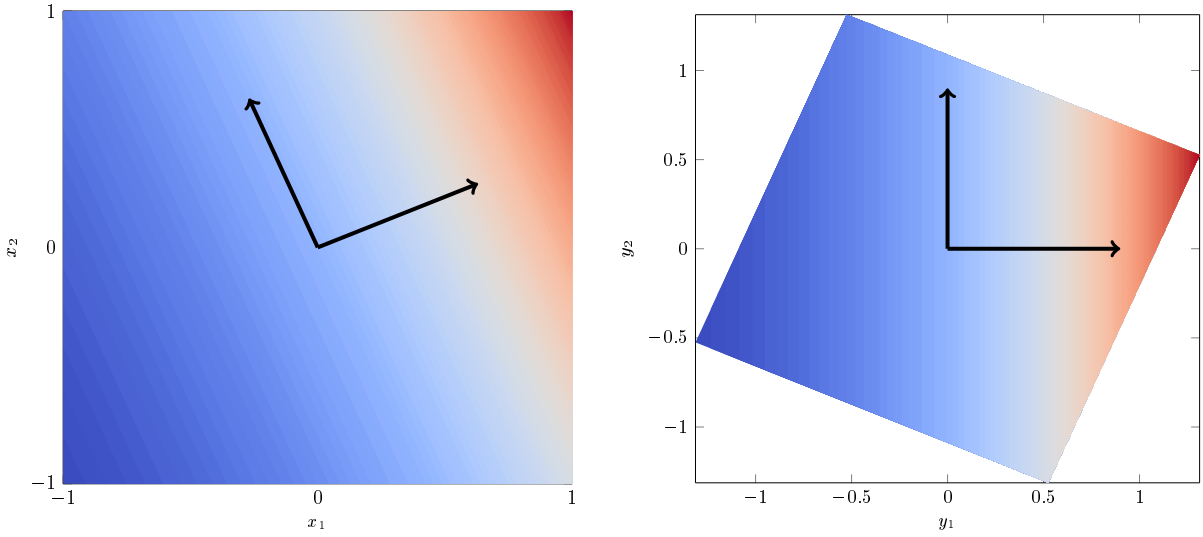
5. A surrogate is then built as a function of the active variables,

$$g(\mathbf{y}) \approx f(\mathbf{x})$$

As a concrete example, consider the function: [25]

$$f(x) = \exp\left(0.7x_1 + 0.3x_2\right).$$

Figure 9.1(a) is a contour plot of $f(x)$. The black arrows indicate the eigenvectors of the matrix $\hat{\mathbf{C}}$. Figure 9.1(b) is the same function but rotated so that the axes are aligned with the eigenvectors. We arbitrarily give these rotated axes the labels y_1 and y_2. From fig. 9.1(b) it is clear that all of the variation is along y_1 and the dimension of the rotated input space can be reduced to 1.



(a) Original function

(b) Rotated function

Figure 9.1: Example of a 2D function with a 1D active subspace.

For additional information, see references [28, 26, 25].

9.1.1 Truncation Methods

Once the eigenvectors of $\hat{\mathbf{C}}$ are obtained we must decide how many directions to keep. If the exact subspace size is known *a priori* it can be specified. Otherwise there are three automatic active subspace detection and truncation methods implemented:

- Constantine metric (default),
- Bing Li metric,
- and Energy metric.

9.1.1.1 Constantine metric

The Constantine metric uses a criterion based on the variability of the subspace estimate. Eigenvectors are computed for bootstrap samples of the gradient matrix. The subspace size associated with the minimum distance between bootstrap eigenvectors and the nominal eigenvectors is the estimated active subspace size.

Below is a brief outline of the Constantine method of active subspace identification. The first two steps are common to all active subspace truncation methods.

1. Compute the gradient of the quantity of interest, $q = f(\mathbf{x})$, at several locations sampled from the input space,
$$\nabla_{\mathbf{x}} f_i = \nabla f(\mathbf{x}_i).$$

2. Compute the eigendecomposition of the matrix $\hat{\mathbf{C}}$,
$$\hat{\mathbf{C}} = \frac{1}{M}\sum_{i=1}^{M} \nabla_{\mathbf{x}} f_i \nabla_{\mathbf{x}} f_i^T = \hat{\mathbf{W}}\hat{\mathbf{\Lambda}}\hat{\mathbf{W}}^T,$$
where $\hat{\mathbf{W}}$ has eigenvectors as columns, $\hat{\mathbf{\Lambda}} = \text{diag}(\hat{\lambda}_1, \ \ldots \ , \hat{\lambda}_N)$ contains eigenvalues, and N is the total number of parameters.

3. Use bootstrap sampling of the gradients found in step 1 to compute replicate eigendecompositions,
$$\hat{\mathbf{C}}_j^* = \hat{\mathbf{W}}_j^* \hat{\mathbf{\Lambda}}_j^* \left(\hat{\mathbf{W}}_j^*\right)^T.$$

4. Compute the average distance between nominal and bootstrap subspaces,
$$e_n^* = \frac{1}{M_{boot}} \sum_{j}^{M_{boot}} \text{dist}(\text{ran}(\hat{\mathbf{W}}_n), \text{ran}(\hat{\mathbf{W}}_{j,n}^*)) = \frac{1}{M_{boot}} \sum_{j}^{M_{boot}} \left\| \hat{\mathbf{W}}_n \hat{\mathbf{W}}_n^T - \hat{\mathbf{W}}_{j,n}^* \left(\hat{\mathbf{W}}_{j,n}^*\right)^T \right\|,$$
where M_{boot} is the number of bootstrap samples, $\hat{\mathbf{W}}_n$ and $\hat{\mathbf{W}}_{j,n}^*$ both contain only the first n eigenvectors, and $n < N$.

5. The estimated subspace rank, r, is then,
$$r = \arg\min_n \ e_n^*.$$

For additional information, see Ref. [25].

9.1.1.2 Bing Li metric

The Bing Li metric uses a trade-off criterion to determine where to truncate the active subspace. The criterion is a function of the eigenvalues and eigenvectors of the active subspace gradient matrix. This function compares the decrease in eigenvalue amplitude with the increase in eigenvector variability under bootstrap sampling of the gradient matrix. The active subspace size is taken to be the index of the first minimum of this quantity.

Below is a brief outline of the Bing Li method of active subspace identification. The first two steps are common to all active subspace truncation methods.

1. Compute the gradient of the quantity of interest, $q = f(\mathbf{x})$, at several locations sampled from the input space,
$$\nabla_{\mathbf{x}} f_i = \nabla f(\mathbf{x}_i).$$
2. Compute the eigendecomposition of the matrix $\hat{\mathbf{C}}$,
$$\hat{\mathbf{C}} = \frac{1}{M}\sum_{i=1}^{M} \nabla_{\mathbf{x}} f_i \nabla_{\mathbf{x}} f_i^T = \hat{\mathbf{W}}\hat{\mathbf{\Lambda}}\hat{\mathbf{W}}^T,$$
where $\hat{\mathbf{W}}$ has eigenvectors as columns, $\hat{\mathbf{\Lambda}} = \text{diag}(\hat{\lambda}_1, \ \ldots \ , \hat{\lambda}_N)$ contains eigenvalues, and N is the total number of parameters.
3. Normalize the eigenvalues,
$$\lambda_i = \frac{\hat{\lambda}_i}{\sum_j^N \hat{\lambda}_j}.$$
4. Use bootstrap sampling of the gradients found in step 1 to compute replicate eigendecompositions,
$$\hat{\mathbf{C}}_j^* = \hat{\mathbf{W}}_j^* \hat{\mathbf{\Lambda}}_j^* \left(\hat{\mathbf{W}}_j^*\right)^T.$$
5. Compute variability of eigenvectors,
$$f_i^0 = \frac{1}{M_{boot}} \sum_{j}^{M_{boot}} \left\{1 - \left|\det\left(\hat{\mathbf{W}}_i^T \hat{\mathbf{W}}_{j,i}^*\right)\right|\right\},$$
where $\hat{\mathbf{W}}_i$ and $\hat{\mathbf{W}}_{j,i}^*$ both contain only the first i eigenvectors and M_{boot} is the number of bootstrap samples. The value of the variability at the first index, f_1^0, is defined as zero.
6. Normalize the eigenvector variability,
$$f_i = \frac{f_i^0}{\sum_j^N f_j^0}.$$
7. The criterion, g_i, is defined as,
$$g_i = \lambda_i + f_i.$$
8. The index of first minimum of g_i is then the estimated active subspace rank.

For additional information, see Ref. [91].

9.1.1.3 Energy metric

The energy metric truncation method uses a criterion based on the derivative matrix eigenvalue energy. The user can specify the maximum percentage (as a decimal) of the eigenvalue energy that is not captured by the active subspace represenation.

Using the eigenvalue energy truncation metric, the subspace size is determined using the following equation:

$$n = \inf\left\{ d \in \mathbb{Z} \quad \middle| \quad 1 \leq d \leq N \quad \wedge \quad 1 - \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{N} \lambda_i} < \epsilon \right\}$$

where ϵ is the `truncation_tolerance`, n is the estimated subspace size, N is the size of the full space, and λ_i are the eigenvalues of the derivative matrix.

9.2 Basis Adaptation Models

The idea behind the basis adaptation is similar to the one employed in the active subspaces that is to find the directions in the input space where the variations of the QoI are negligible or they can be safely discarded, *i.e.* without significantly affecting the QoI's statistics, according to a truncation criterion. One of the main differences between the basis adaptation and the active subspaces strategy is that the basis adaptation approach relies on the construction of a Polynomial Chaos Expansion (PCE) that is subsequently rotated to decrease the dimensionality of the problem.

As in the case of PCE, let's be $\mathcal{H}$ the Hilbert space formed by the closed linear span of $\boldsymbol{\xi}$ and let $\mathcal{F}(\mathcal{H})$ be the σ-algebra generated by $\boldsymbol{\xi}$. A generic QoI Q can be approximated by the PCE up to order p as

$$Q(\boldsymbol{\xi}) = \sum_{\boldsymbol{\alpha} \in \mathcal{J}_{d,p}} Q_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \,, \tag{9.1}$$

where $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_d) \in \mathcal{J}_{d,p} := (\mathbb{N}_0)^d$ with $|\boldsymbol{\alpha}| = \sum_{i=1}^{d} \alpha_i <= d$ is multi-index of dimension d and order up to p. In this chapter, for simplicity of exposure, we assume the expansion with respect to a basis of (normalized) Hermite polynomials and $\boldsymbol{\xi}$ is assumed to have standard multivariate Gaussian distribution. The general case of arbitrary distribution can be handled, at least from a theoretical standpoint, by resorting to input parameter transformations as the inverse of cumulative distribution function or other more sophisticated transformations like the Rosenblatt transformation. The $P = \binom{n+p}{p}$ PCE coefficients can be computed by projecting Q to the space spanned by $\{\psi_{\boldsymbol{\alpha}}, \boldsymbol{\alpha} \in \mathcal{J}_{d,p}\}$ (or other methods like Monte Carlo and regression) as

$$Q_{\boldsymbol{\alpha}} = \frac{\langle Q, \psi_{\boldsymbol{\alpha}} \rangle}{\langle \psi_{\boldsymbol{\alpha}}^2 \rangle} = \langle Q, \psi_{\boldsymbol{\alpha}} \rangle, \quad \boldsymbol{\alpha} \in \mathcal{J}_{d,p} \,. \tag{9.2}$$

The basis adaptation method tries to rotate the input Gaussian variables by an isometry such that the QoI can be well approximated by PCE of the first several dimensions of the new orthogonal basis. Let $\boldsymbol{A}$ be an isometry on $\mathbb{R}^{d \times d}$ such that $\boldsymbol{A}\boldsymbol{A}^{\boldsymbol{T}} = \boldsymbol{I}$, and $\boldsymbol{\eta}$ be defined as

$$\boldsymbol{\eta} = \boldsymbol{A}\boldsymbol{\xi}, \qquad \boldsymbol{\eta} = \begin{Bmatrix} \boldsymbol{\eta}_r \\ \boldsymbol{\eta}_{\neg r} \end{Bmatrix} , \tag{9.3}$$

It follows that $\boldsymbol{\eta}$ also has multivariate Gaussian distribution. Then the expansion $Q^{\boldsymbol{A}}$ in terms of $\boldsymbol{\eta}$ can be obtained as

$$Q^{\boldsymbol{A}}(\boldsymbol{\eta}) = \sum_{\boldsymbol{\beta} \in \mathcal{J}_{d,p}} Q_{\boldsymbol{\beta}}^{\boldsymbol{A}} \psi_{\boldsymbol{\beta}}(\boldsymbol{\eta}) \,. \tag{9.4}$$

Since $\{\psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\}$ and $\{\psi_{\boldsymbol{\beta}}(\boldsymbol{\eta})\}$ span the same space, $Q^{\boldsymbol{A}}(\boldsymbol{\eta}(\boldsymbol{\xi})) \triangleq Q(\boldsymbol{\xi})$, and thus

$$Q_{\boldsymbol{\alpha}} = \sum_{\boldsymbol{\beta} \in \mathcal{J}_{d,p}} Q_{\boldsymbol{\beta}}^{\boldsymbol{A}} \langle \psi_{\boldsymbol{\beta}}^{\boldsymbol{A}}, \psi_{\boldsymbol{\alpha}} \rangle, \ \boldsymbol{\alpha} \in \mathcal{J}_{d,p} \,. \tag{9.5}$$

This latter equation provides foundation to transform PCE from the original space spanned by $\boldsymbol{\xi}$ to the new space spanned by $\boldsymbol{\eta}$. In the classical Gaussian adaptation, also called linear adaptation, the rotation matrix $\boldsymbol{A}$ is constructed such that

$$\eta_1 = \sum_{\boldsymbol{\alpha} \in \mathcal{J}_{d,1}} Q_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \sum_{i=1}^{d} Q_{\boldsymbol{e}_i} \xi_i \tag{9.6}$$

where $\boldsymbol{e}_i$ is d-dimensional multi-index with 1 at i-th location and zeros elsewhere, *i.e.* the first order PCE coefficients in the original space are placed in the first row of the initial construction of $\boldsymbol{A}$. The benefit of this approach

is that the complete Gaussian components of Q are contained in the variable η_1. Note that the first order PC coefficients also represent the sensitivities of the input parameters because the derivative of the first order PCE expansion with respect to each variable is always equal to its coefficient. Once the first the row of $\boldsymbol{A}$ is defined, the first order PC coefficient with largest absolute value are placed on each subsequent row of $\boldsymbol{A}$ in the same columns as they appear in the first row of $\boldsymbol{A}$. All other elements are equal to zero. For instance, if we consider the following PCE expansion

$$Q(\boldsymbol{\xi}) = \beta_0 + 2\xi_1 + 5\xi_2 + 1\xi_3,$$

the corresponding $\boldsymbol{A}$ would be

$$\begin{bmatrix} 2.0 & 5.0 & 1.0 \\ 0.0 & 5.0 & 0.0 \\ 2.0 & 0.0 & 0.0 \end{bmatrix}.$$

The procedure described above reflects the relative importance/sensitivities with respect to the original input parameters. A Gram-Schmidt procedure is then applied to make $\boldsymbol{A}$ an isometry. The transformed variables has descending importance in the probabilistic space which is the foundation that we could achieve accurate representation of QoI by only the first several dimensions.

Suppose the dimension after reduction is $r < d$, we can project Q to the space spanned by Hermite polynomials $\{\psi_{\boldsymbol{\beta}}^{\boldsymbol{A}_r}, \boldsymbol{\beta} \in \mathcal{J}_{r,p}\}$,

$$Q^{\boldsymbol{A}_r}(\boldsymbol{\eta}_r) = Q^{\boldsymbol{A}}\left(\begin{Bmatrix} \boldsymbol{\eta}_r \\ \boldsymbol{0} \end{Bmatrix}\right) = \sum_{\boldsymbol{\beta} \in \mathcal{J}_{r,p}} Q_{\boldsymbol{\beta}}^{\boldsymbol{A}_r} \psi_{\boldsymbol{\beta}}(\boldsymbol{\eta}_r) \tag{9.7}$$

where $\mathcal{J}_{r,p} \subset \mathcal{J}_{d,p}$ is the set of multi-indices that only have non-zero entries regarding $\boldsymbol{\eta}_r$; $\boldsymbol{A}_r$ are the first r rows of the rotation matrix $\boldsymbol{A}$; and the superscript $\boldsymbol{A}_r$ stresses that the expansion is in terms of $\boldsymbol{\eta}_r$. PC coefficients of the above expansion are obtained by projecting Q to the space spanned by $\{\psi_{\boldsymbol{\beta}}^{\boldsymbol{A}_r}, \boldsymbol{\beta} \in \mathcal{J}_{r,p}\}$

$$Q_{\boldsymbol{\beta}}^{\boldsymbol{A}_r} = \langle Q, \psi_{\boldsymbol{\beta}}^{\boldsymbol{A}_r} \rangle . \tag{9.8}$$

The PC coefficient in η space can be transformed to ξ space by eq. (9.5) as

$$\tilde{Q}_{\boldsymbol{\alpha}} = \sum_{\boldsymbol{\beta} \in \mathcal{J}_{r,p}} Q_{\boldsymbol{\beta}}^{\boldsymbol{A}_r} \langle \psi_{\boldsymbol{\beta}}^{\boldsymbol{A}_r}, \psi_{\boldsymbol{\alpha}} \rangle . \tag{9.9}$$

If we define the vectors of the PCE coefficients $\tilde{\boldsymbol{Q}}_{coeff} := \{\tilde{Q}_{\boldsymbol{\alpha}}, \boldsymbol{\alpha} \in \mathcal{J}_{d,p}\}$ and $\boldsymbol{Q}_{coeff} := \{Q_{\boldsymbol{\alpha}}, \boldsymbol{\alpha} \in \mathcal{J}_{d,p}\}$, the relative 2-norm error of PCE in ξ space can be measured by

$$\boldsymbol{\epsilon}_D = \frac{\left\| \boldsymbol{Q}_{coeff} - \tilde{\boldsymbol{Q}}_{coeff} \right\|_2}{\left\| \boldsymbol{Q}_{coeff} \right\|_2} . \tag{9.10}$$

Note that although (9.10) provides a way to compare the r-d adaptation with the full dimensional PCE, in practical, it is more convenient to compare two adaptations with successive dimensions, say, r-d and $(r+1)$-d, to check the convergence. The accuracy of basis adaptation increases with increase of r and will recover full dimensional expansion with $r = d$. At this stage the implementation in Dakota of this strategy will use the full dimension of the original model, *i.e.* the rotation matrix is not truncated and the model is expressed in a different variable set. Although the rotation matrix is explicitly reported and it can be used to obtain information about the significant directions of the model.

Chapter 10

Optimization Under Uncertainty (OUU)

10.1 Reliability-Based Design Optimization (RBDO)

Reliability-based design optimization (RBDO) methods are used to perform design optimization accounting for reliability metrics. The reliability analysis capabilities described in Section 2.1 provide a substantial foundation for exploring a variety of gradient-based RBDO formulations. [45] investigated bi-level, fully-analytic bi-level, and first-order sequential RBDO approaches employing underlying first-order reliability assessments. [46] investigated fully-analytic bi-level and second-order sequential RBDO approaches employing underlying second-order reliability assessments. These methods are overviewed in the following sections.

10.1.1 Bi-level RBDO

The simplest and most direct RBDO approach is the bi-level approach in which a full reliability analysis is performed for every optimization function evaluation. This involves a nesting of two distinct levels of optimization within each other, one at the design level and one at the MPP search level.

Since an RBDO problem will typically specify both the $\bar{z}$ level and the $\bar{p}/\bar{\beta}$ level, one can use either the RIA or the PMA formulation for the UQ portion and then constrain the result in the design optimization portion. In particular, RIA reliability analysis maps $\bar{z}$ to p/β, so RIA RBDO constrains p/β:

$$\begin{aligned} \text{minimize} \quad & f \\ \text{subject to} \quad & \beta \geq \bar{\beta} \\ \text{or} \quad & p \leq \bar{p} \end{aligned} \tag{10.1}$$

And PMA reliability analysis maps $\bar{p}/\bar{\beta}$ to z, so PMA RBDO constrains z:

$$\begin{aligned} \text{minimize} \quad & f \\ \text{subject to} \quad & z \geq \bar{z} \end{aligned} \tag{10.2}$$

where $z \geq \bar{z}$ is used as the RBDO constraint for a cumulative failure probability (failure defined as $z \leq \bar{z}$) but $z \leq \bar{z}$ would be used as the RBDO constraint for a complementary cumulative failure probability (failure defined as $z \geq \bar{z}$). It is worth noting that Dakota is not limited to these types of inequality-constrained RBDO formulations;

rather, they are convenient examples. Dakota supports general optimization under uncertainty mappings [50] which allow flexible use of statistics within multiple objectives, inequality constraints, and equality constraints.

An important performance enhancement for bi-level methods is the use of sensitivity analysis to analytically compute the design gradients of probability, reliability, and response levels. When design variables are separate from the uncertain variables (i.e., they are not distribution parameters), then the following first-order expressions may be used [73, 80, 6]:

$$\nabla_{\mathbf{d}} z = \nabla_{\mathbf{d}} g \tag{10.3}$$

$$\nabla_{\mathbf{d}} \beta_{cdf} = \frac{1}{\| \nabla_{\mathbf{u}} G \|} \nabla_{\mathbf{d}} g \tag{10.4}$$

$$\nabla_{\mathbf{d}} p_{cdf} = -\phi(-\beta_{cdf}) \nabla_{\mathbf{d}} \beta_{cdf} \tag{10.5}$$

where it is evident from Eqs. 2.10-2.11 that $\nabla_{\mathbf{d}} \beta_{ccdf} = -\nabla_{\mathbf{d}} \beta_{cdf}$ and $\nabla_{\mathbf{d}} p_{ccdf} = -\nabla_{\mathbf{d}} p_{cdf}$. In the case of second-order integrations, Eq. 10.5 must be expanded to include the curvature correction. For Breitung's correction (Eq. 2.39),

$$\nabla_{\mathbf{d}} p_{cdf} = \left[\Phi(-\beta_p) \sum_{i=1}^{n-1} \left(\frac{-\kappa_i}{2(1+\beta_p \kappa_i)^{\frac{3}{2}}} \prod_{\substack{j=1 \\ j \neq i}}^{n-1} \frac{1}{\sqrt{1+\beta_p \kappa_j}} \right) - \phi(-\beta_p) \prod_{i=1}^{n-1} \frac{1}{\sqrt{1+\beta_p \kappa_i}} \right] \nabla_{\mathbf{d}} \beta_{cdf} \tag{10.6}$$

where $\nabla_{\mathbf{d}} \kappa_i$ has been neglected and $\beta_p \geq 0$ (see Section 2.1.2.2). Other approaches assume the curvature correction is nearly independent of the design variables [110], which is equivalent to neglecting the first term in Eq. 10.6.

To capture second-order probability estimates within an RIA RBDO formulation using well-behaved β constraints, a generalized reliability index can be introduced where, similar to Eq. 2.8,

$$\beta^*_{cdf} = -\Phi^{-1}(p_{cdf}) \tag{10.7}$$

for second-order p_{cdf}. This reliability index is no longer equivalent to the magnitude of $\mathbf{u}$, but rather is a convenience metric for capturing the effect of more accurate probability estimates. The corresponding generalized reliability index sensitivity, similar to Eq. 10.5, is

$$\nabla_{\mathbf{d}} \beta^*_{cdf} = -\frac{1}{\phi(-\beta^*_{cdf})} \nabla_{\mathbf{d}} p_{cdf} \tag{10.8}$$

where $\nabla_{\mathbf{d}} p_{cdf}$ is defined from Eq. 10.6. Even when $\nabla_{\mathbf{d}} g$ is estimated numerically, Eqs. 10.3-10.8 can be used to avoid numerical differencing across full reliability analyses.

When the design variables are distribution parameters of the uncertain variables, $\nabla_{\mathbf{d}} g$ is expanded with the chain rule and Eqs. 10.3 and 10.4 become

$$\nabla_{\mathbf{d}} z = \nabla_{\mathbf{d}} \mathbf{x} \nabla_{\mathbf{x}} g \tag{10.9}$$

$$\nabla_{\mathbf{d}} \beta_{cdf} = \frac{1}{\| \nabla_{\mathbf{u}} G \|} \nabla_{\mathbf{d}} \mathbf{x} \nabla_{\mathbf{x}} g \tag{10.10}$$

where the design Jacobian of the transformation ($\nabla_{\mathbf{d}} \mathbf{x}$) may be obtained analytically for uncorrelated $\mathbf{x}$ or semi-analytically for correlated $\mathbf{x}$ ($\nabla_{\mathbf{d}} \mathbf{L}$ is evaluated numerically) by differentiating Eqs. 2.14 and 2.15 with respect to the distribution parameters. Eqs. 10.5-10.8 remain the same as before. For this design variable case, all required information for the sensitivities is available from the MPP search.

Since Eqs. 10.3-10.10 are derived using the Karush-Kuhn-Tucker optimality conditions for a converged MPP, they are appropriate for RBDO using AMV+, AMV^2+, TANA, FORM, and SORM, but not for RBDO using MVFOSM, MVSOSM, AMV, or AMV^2.

10.1.2 Sequential/Surrogate-based RBDO

An alternative RBDO approach is the sequential approach, in which additional efficiency is sought through breaking the nested relationship of the MPP and design searches. The general concept is to iterate between optimization and uncertainty quantification, updating the optimization goals based on the most recent probabilistic assessment results. This update may be based on safety factors [138] or other approximations [41].

A particularly effective approach for updating the optimization goals is to use the $p/\beta/z$ sensitivity analysis of Eqs. 10.3-10.10 in combination with local surrogate models [146]. In [45] and [46], first-order and second-order Taylor series approximations were employed within a trust-region model management framework [67] in order to adaptively manage the extent of the approximations and ensure convergence of the RBDO process. Surrogate models were used for both the objective function and the constraints, although the use of constraint surrogates alone is sufficient to remove the nesting.

In particular, RIA trust-region surrogate-based RBDO employs surrogate models of f and p/β within a trust region Δ^k centered at $\mathbf{d}_c$. For first-order surrogates:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{d}_c) + \nabla_d f(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) \\ \text{subject to} \quad & \beta(\mathbf{d}_c) + \nabla_d \beta(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) \geq \bar{\beta} \\ \text{or} \quad & p(\mathbf{d}_c) + \nabla_d p(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) \leq \bar{p} \\ & \| \mathbf{d}-\mathbf{d}_c \|_\infty \leq \Delta^k \end{aligned} \tag{10.11}$$

and for second-order surrogates:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{d}_c) + \nabla_{\mathbf{d}} f(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) + \tfrac{1}{2}(\mathbf{d}-\mathbf{d}_c)^T\nabla^2_{\mathbf{d}} f(\mathbf{d}_c)(\mathbf{d}-\mathbf{d}_c) \\ \text{subject to} \quad & \beta(\mathbf{d}_c) + \nabla_{\mathbf{d}} \beta(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) + \tfrac{1}{2}(\mathbf{d}-\mathbf{d}_c)^T\nabla^2_{\mathbf{d}} \beta(\mathbf{d}_c)(\mathbf{d}-\mathbf{d}_c) \geq \bar{\beta} \\ \text{or} \quad & p(\mathbf{d}_c) + \nabla_{\mathbf{d}} p(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) + \tfrac{1}{2}(\mathbf{d}-\mathbf{d}_c)^T\nabla^2_{\mathbf{d}} p(\mathbf{d}_c)(\mathbf{d}-\mathbf{d}_c) \leq \bar{p} \\ & \| \mathbf{d}-\mathbf{d}_c \|_\infty \leq \Delta^k \end{aligned} \tag{10.12}$$

For PMA trust-region surrogate-based RBDO, surrogate models of f and z are employed within a trust region Δ^k centered at $\mathbf{d}_c$. For first-order surrogates:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{d}_c) + \nabla_d f(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) \\ \text{subject to} \quad & z(\mathbf{d}_c) + \nabla_d z(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) \geq \bar{z} \\ & \| \mathbf{d}-\mathbf{d}_c \|_\infty \leq \Delta^k \end{aligned} \tag{10.13}$$

and for second-order surrogates:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{d}_c) + \nabla_{\mathbf{d}} f(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) + \tfrac{1}{2}(\mathbf{d}-\mathbf{d}_c)^T\nabla^2_{\mathbf{d}} f(\mathbf{d}_c)(\mathbf{d}-\mathbf{d}_c) \\ \text{subject to} \quad & z(\mathbf{d}_c) + \nabla_{\mathbf{d}} z(\mathbf{d}_c)^T(\mathbf{d}-\mathbf{d}_c) + \tfrac{1}{2}(\mathbf{d}-\mathbf{d}_c)^T\nabla^2_{\mathbf{d}} z(\mathbf{d}_c)(\mathbf{d}-\mathbf{d}_c) \geq \bar{z} \\ & \| \mathbf{d}-\mathbf{d}_c \|_\infty \leq \Delta^k \end{aligned} \tag{10.14}$$

where the sense of the z constraint may vary as described previously. The second-order information in Eqs. 10.12 and 10.14 will typically be approximated with quasi-Newton updates.

10.2 Stochastic Expansion-Based Design Optimization (SEBDO)

10.2.1 Stochastic Sensitivity Analysis

Section 3.9 describes sensitivity analysis of the polynomial chaos expansion with respect to the expansion variables. Here we extend this analysis to include sensitivity analysis of probabilistic moments with respect to non-

probabilistic (i.e., design or epistemic uncertain) variables.

10.2.1.1 Local sensitivity analysis: first-order probabilistic expansions

With the introduction of nonprobabilistic variables $\boldsymbol{s}$ (for example, design variables or epistemic uncertain variables), a polynomial chaos expansion only over the probabilistic variables $\boldsymbol{\xi}$ has the functional relationship:

$$R(\boldsymbol{\xi}, \boldsymbol{s}) \cong \sum_{j=0}^{P} \alpha_j(\boldsymbol{s}) \Psi_j(\boldsymbol{\xi}) \tag{10.15}$$

For computing sensitivities of response mean and variance, the ij indices may be dropped from Eqs. 3.54 and 3.55, simplifying to

$$\mu(s) \ = \ \alpha_0(s), \quad \sigma^2(s) = \sum_{k=1}^{P} \alpha_k^2(s) \langle \Psi_k^2 \rangle \tag{10.16}$$

Sensitivities of Eq. 10.16 with respect to the nonprobabilistic variables are as follows, where independence of $\boldsymbol{s}$ and $\boldsymbol{\xi}$ is assumed:

$$\frac{d\mu}{ds} \ = \ \frac{d\alpha_0}{ds} \ = \ \langle \frac{dR}{ds} \rangle \tag{10.17}$$

$$\frac{d\sigma^2}{ds} \ = \ \sum_{k=1}^{P} \langle \Psi_k^2 \rangle \frac{d\alpha_k^2}{ds} \ = \ 2 \sum_{k=1}^{P} \alpha_k \langle \frac{dR}{ds}, \Psi_k \rangle \tag{10.18}$$

where

$$\frac{d\alpha_k}{ds} = \frac{\langle \frac{dR}{ds}, \Psi_k \rangle}{\langle \Psi_k^2 \rangle} \tag{10.19}$$

has been used. Due to independence, the coefficients calculated in Eq. 10.19 may be interpreted as either the derivatives of the expectations or the expectations of the derivatives, or more precisely, the nonprobabilistic sensitivities of the chaos coefficients for the response expansion or the chaos coefficients of an expansion for the nonprobabilistic sensitivities of the response. The evaluation of integrals involving $\frac{dR}{ds}$ extends the data requirements for the PCE approach to include response sensitivities at each of the sampled points.The resulting expansions are valid only for a particular set of nonprobabilistic variables and must be recalculated each time the nonprobabilistic variables are modified.

Similarly for stochastic collocation,

$$R(\boldsymbol{\xi}, \boldsymbol{s}) \cong \sum_{k=1}^{N_p} r_k(\boldsymbol{s}) \boldsymbol{L}_k(\boldsymbol{\xi}) \tag{10.20}$$

leads to

$$\mu(s) \ = \ \sum_{k=1}^{N_p} r_k(s) w_k, \quad \sigma^2(s) \ = \ \sum_{k=1}^{N_p} r_k^2(s) w_k - \mu^2(s) \tag{10.21}$$

$$\frac{d\mu}{ds} \ = \ \sum_{k=1}^{N_p} w_k \frac{dr_k}{ds} \tag{10.22}$$

$$\frac{d\sigma^2}{ds} \ = \ \sum_{k=1}^{N_p} 2 w_k r_k \frac{dr_k}{ds} - 2\mu \frac{d\mu}{ds} \ = \ \sum_{k=1}^{N_p} 2 w_k (r_k - \mu) \frac{dr_k}{ds} \tag{10.23}$$

10.2.1.2 Local sensitivity analysis: zeroth-order combined expansions

Alternatively, a stochastic expansion can be formed over both $\boldsymbol{\xi}$ and $\boldsymbol{s}$. Assuming a bounded design domain $\boldsymbol{s}_L \leq \boldsymbol{s} \leq \boldsymbol{s}_U$ (with no implied probability content), a Legendre chaos basis would be appropriate for each of the dimensions in $\boldsymbol{s}$ within a polynomial chaos expansion.

$$R(\boldsymbol{\xi}, \boldsymbol{s}) \cong \sum_{j=0}^{P} \alpha_j \Psi_j(\boldsymbol{\xi}, \boldsymbol{s}) \tag{10.24}$$

In this case, design sensitivities for the mean and variance do not require response sensitivity data, but this comes at the cost of forming the PCE over additional dimensions. For this combined variable expansion, the mean and variance are evaluated by performing the expectations over only the probabilistic expansion variables, which eliminates the polynomial dependence on $\boldsymbol{\xi}$, leaving behind the desired polynomial dependence of the moments on $\boldsymbol{s}$:

$$\mu_R(\boldsymbol{s}) = \sum_{j=0}^{P} \alpha_j \langle \Psi_j(\boldsymbol{\xi}, \boldsymbol{s}) \rangle_{\boldsymbol{\xi}} \tag{10.25}$$

$$\sigma_R^2(\boldsymbol{s}) = \sum_{j=0}^{P} \sum_{k=0}^{P} \alpha_j \alpha_k \langle \Psi_j(\boldsymbol{\xi}, \boldsymbol{s}) \Psi_k(\boldsymbol{\xi}, \boldsymbol{s}) \rangle_{\boldsymbol{\xi}} - \mu_R^2(\boldsymbol{s}) \tag{10.26}$$

The remaining polynomials may then be differentiated with respect to $\boldsymbol{s}$. In this approach, the combined PCE is valid for the full design variable range ($\boldsymbol{s}_L \leq \boldsymbol{s} \leq \boldsymbol{s}_U$) and does not need to be updated for each change in nonprobabilistic variables, although adaptive localization techniques (i.e., trust region model management approaches) can be employed when improved local accuracy of the sensitivities is required.

Similarly for stochastic collocation,

$$R(\boldsymbol{\xi}, \boldsymbol{s}) \cong \sum_{j=1}^{N_p} r_j \boldsymbol{L}_j(\boldsymbol{\xi}, \boldsymbol{s}) \tag{10.27}$$

leads to

$$\mu_R(\boldsymbol{s}) = \sum_{j=1}^{N_p} r_j \langle \boldsymbol{L}_j(\boldsymbol{\xi}, \boldsymbol{s}) \rangle_{\boldsymbol{\xi}} \tag{10.28}$$

$$\sigma_R^2(\boldsymbol{s}) = \sum_{j=1}^{N_p} \sum_{k=1}^{N_p} r_j r_k \langle \boldsymbol{L}_j(\boldsymbol{\xi}, \boldsymbol{s}) \boldsymbol{L}_k(\boldsymbol{\xi}, \boldsymbol{s}) \rangle_{\boldsymbol{\xi}} - \mu_R^2(\boldsymbol{s}) \tag{10.29}$$

where the remaining polynomials not eliminated by the expectation over $\boldsymbol{\xi}$ are again differentiated with respect to $\boldsymbol{s}$.

10.2.1.3 Inputs and outputs

There are two types of nonprobabilistic variables for which sensitivities must be calculated: "augmented," where the nonprobabilistic variables are separate from and augment the probabilistic variables, and "inserted," where the nonprobabilistic variables define distribution parameters for the probabilistic variables. Any inserted nonprobabilistic variable sensitivities must be handled using Eqs. 10.17-10.18 and Eqs. 10.22-10.23 where $\frac{dR}{ds}$ is calculated as $\frac{dR}{dx}\frac{dx}{ds}$ and $\frac{dx}{ds}$ is the Jacobian of the variable transformation $\mathbf{x} = T^{-1}(\boldsymbol{\xi})$ with respect to the inserted nonprobabilistic variables. In addition, parameterized polynomials (generalized Gauss-Laguerre, Jacobi,

and numerically-generated polynomials) may introduce a $\frac{d\Psi}{ds}$ or $\frac{d\boldsymbol{L}}{ds}$ dependence for inserted s that will introduce additional terms in the sensitivity expressions.

While moment sensitivities directly enable robust design optimization and interval estimation formulations which seek to control or bound response variance, control or bounding of reliability requires sensitivities of tail statistics. In this work, the sensitivity of simple moment-based approximations to cumulative distribution function (CDF) and complementary cumulative distribution function (CCDF) mappings (Eqs. 2.3–2.4) are employed for this purpose, such that it is straightforward to form approximate design sensitivities of reliability index β (forward reliability mapping $\bar{z} \rightarrow \beta$) or response level z (inverse reliability mapping $\bar{\beta} \rightarrow z$) from the moment design sensitivities and the specified levels $\bar{\beta}$ or $\bar{z}$.

10.2.2 Optimization Formulations

Given the capability to compute analytic statistics of the response along with design sensitivities of these statistics, Dakota supports bi-level, sequential, and multifidelity approaches for optimization under uncertainty (OUU). The latter two approaches apply surrogate modeling approaches (data fits and multifidelity modeling) to the uncertainty analysis and then apply trust region model management to the optimization process.

10.2.2.1 Bi-level SEBDO

The simplest and most direct approach is to employ these analytic statistics and their design derivatives from Section 10.2.1 directly within an optimization loop. This approach is known as bi-level OUU, since there is an inner level uncertainty analysis nested within an outer level optimization.

Consider the common reliability-based design example of a deterministic objective function with a reliability constraint:

$$\begin{aligned} \text{minimize} \quad & f \\ \text{subject to} \quad & \beta \geq \bar{\beta} \end{aligned} \tag{10.30}$$

where β is computed relative to a prescribed threshold response value $\bar{z}$ (e.g., a failure threshold) and is constrained by a prescribed reliability level $\bar{\beta}$ (minimum allowable reliability in the design), and is either a CDF or CCDF index depending on the definition of the failure domain (i.e., defined from whether the associated failure probability is cumulative, $p(g \leq \bar{z})$, or complementary cumulative, $p(g > \bar{z})$).

Another common example is robust design in which the constraint enforcing a reliability lower-bound has been replaced with a constraint enforcing a variance upper-bound $\bar{\sigma}^2$ (maximum allowable variance in the design):

$$\begin{aligned} \text{minimize} \quad & f \\ \text{subject to} \quad & \sigma^2 \leq \bar{\sigma}^2 \end{aligned} \tag{10.31}$$

Solving these problems using a bi-level approach involves computing β and $\frac{d\beta}{d\boldsymbol{s}}$ for Eq. 10.30 or σ^2 and $\frac{d\sigma^2}{d\boldsymbol{s}}$ for Eq. 10.31 for each set of design variables $\boldsymbol{s}$ passed from the optimizer. This approach is supported for both probabilistic and combined expansions using PCE and SC.

10.2.2.2 Sequential/Surrogate-Based SEBDO

An alternative OUU approach is the sequential approach, in which additional efficiency is sought through breaking the nested relationship of the UQ and optimization loops. The general concept is to iterate between optimization

and uncertainty quantification, updating the optimization goals based on the most recent uncertainty assessment results. This approach is common with the reliability methods community, for which the updating strategy may be based on safety factors [138] or other approximations [41].

A particularly effective approach for updating the optimization goals is to use data fit surrogate models, and in particular, local Taylor series models allow direct insertion of stochastic sensitivity analysis capabilities. In Ref. [45], first-order Taylor series approximations were explored, and in Ref. [46], second-order Taylor series approximations are investigated. In both cases, a trust-region model management framework [48] is used to adaptively manage the extent of the approximations and ensure convergence of the OUU process. Surrogate models are used for both the objective and the constraint functions, although the use of surrogates is only required for the functions containing statistical results; deterministic functions may remain explicit is desired.

In particular, trust-region surrogate-based optimization for reliability-based design employs surrogate models of f and β within a trust region Δ^k centered at $\mathbf{s}_c$:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{s}_c) + \nabla_s f(\mathbf{s}_c)^T(\mathbf{s} - \mathbf{s}_c) \\ \text{subject to} \quad & \beta(\mathbf{s}_c) + \nabla_s \beta(\mathbf{s}_c)^T(\mathbf{s} - \mathbf{s}_c) \geq \bar{\beta} \\ & \| \mathbf{s} - \mathbf{s}_c \|_\infty \leq \Delta^k \end{aligned} \tag{10.32}$$

and trust-region surrogate-based optimization for robust design employs surrogate models of f and σ^2 within a trust region Δ^k centered at $\mathbf{s}_c$:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{s}_c) + \nabla_s f(\mathbf{s}_c)^T(\mathbf{s} - \mathbf{s}_c) \\ \text{subject to} \quad & \sigma^2(\mathbf{s}_c) + \nabla_s \sigma^2(\mathbf{s}_c)^T(\mathbf{s} - \mathbf{s}_c) \leq \bar{\sigma}^2 \\ & \| \mathbf{s} - \mathbf{s}_c \|_\infty \leq \Delta^k \end{aligned} \tag{10.33}$$

Second-order local surrogates may also be employed, where the Hessians are typically approximated from an accumulation of curvature information using quasi-Newton updates [101] such as Broyden-Fletcher-Goldfarb-Shanno (BFGS, Eq. 2.43) or symmetric rank one (SR1, Eq. 2.44). The sequential approach is available for probabilistic expansions using PCE and SC.

10.2.2.3 Multifidelity SEBDO

The multifidelity OUU approach is another trust-region surrogate-based approach. Instead of the surrogate UQ model being a simple data fit (in particular, first-/second-order Taylor series model) of the truth UQ model results, distinct UQ models of differing fidelity are now employed. This differing UQ fidelity could stem from the fidelity of the underlying simulation model, the fidelity of the UQ algorithm, or both. In this section, we focus on the fidelity of the UQ algorithm. For reliability methods, this could entail varying fidelity in approximating assumptions (e.g., Mean Value for low fidelity, SORM for high fidelity), and for stochastic expansion methods, it could involve differences in selected levels of p and h refinement.

Here, we define UQ fidelity as point-wise accuracy in the design space and take the high fidelity truth model to be the probabilistic expansion PCE/SC model, with validity only at a single design point. The low fidelity model, whose validity over the design space will be adaptively controlled, will be either the combined expansion PCE/SC model, with validity over a range of design parameters, or the MVFOSM reliability method, with validity only at a single design point. The combined expansion low fidelity approach will span the current trust region of the design space and will be reconstructed for each new trust region. Trust region adaptation will ensure that the combined expansion approach remains sufficiently accurate for design purposes. By taking advantage of the design space spanning, one can eliminate the cost of multiple low fidelity UQ analyses within the trust region, with fallback to the greater accuracy and higher expense of the probabilistic expansion approach when needed. The

MVFOSM low fidelity approximation must be reformed for each change in design variables, but it only requires a single evaluation of a response function and its derivative to approximate the response mean and variance from the input mean and covariance (Eqs. 2.1–2.2) from which forward/inverse CDF/CCDF reliability mappings can be generated using Eqs. 2.3–2.4. This is the least expensive UQ option, but its limited accuracy[1] may dictate the use of small trust regions, resulting in greater iterations to convergence. The expense of optimizing a combined expansion, on the other hand, is not significantly less than that of optimizing the high fidelity UQ model, but its representation of global trends should allow the use of larger trust regions, resulting in reduced iterations to convergence. The design derivatives of each of the PCE/SC expansion models provide the necessary data to correct the low fidelity model to first-order consistency with the high fidelity model at the center of each trust region, ensuring convergence of the multifidelity optimization process to the high fidelity optimum. Design derivatives of the MVFOSM statistics are currently evaluated numerically using forward finite differences.

Multifidelity optimization for reliability-based design can be formulated as:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{s}) \\ \text{subject to} \quad & \hat{\beta_{hi}}(\mathbf{s}) \geq \bar{\beta} \\ & \| \mathbf{s} - \mathbf{s}_c \|_\infty \leq \Delta^k \end{aligned} \tag{10.34}$$

and multifidelity optimization for robust design can be formulated as:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{s}) \\ \text{subject to} \quad & \hat{\sigma_{hi}}^2(\mathbf{s}) \leq \bar{\sigma}^2 \\ & \| \mathbf{s} - \mathbf{s}_c \|_\infty \leq \Delta^k \end{aligned} \tag{10.35}$$

where the deterministic objective function is not approximated and $\hat{\beta_{hi}}$ and $\hat{\sigma_{hi}}^2$ are the approximated high-fidelity UQ results resulting from correction of the low-fidelity UQ results. In the case of an additive correction function:

$$\hat{\beta_{hi}}(\mathbf{s}) = \beta_{lo}(\mathbf{s}) + \alpha_\beta(\mathbf{s}) \tag{10.36}$$

$$\hat{\sigma_{hi}}^2(\mathbf{s}) = \sigma^2_{lo}(\mathbf{s}) + \alpha_{\sigma^2}(\mathbf{s}) \tag{10.37}$$

where correction functions $\alpha(\mathbf{s})$ enforcing first-order consistency [49] are typically employed. Quasi-second-order correction functions [49] can also be employed, but care must be taken due to the different rates of curvature accumulation between the low and high fidelity models. In particular, since the low fidelity model is evaluated more frequently than the high fidelity model, it accumulates curvature information more quickly, such that enforcing quasi-second-order consistency with the high fidelity model can be detrimental in the initial iterations of the algorithm[2]. Instead, this consistency should only be enforced when sufficient high fidelity curvature information has been accumulated (e.g., after n rank one updates).

10.3 Sampling-based OUU

Gradient-based OUU can also be performed using random sampling methods. In this case, the sample-average approximation to the design derivative of the mean and standard deviation are:

$$\frac{d\mu}{ds} = \frac{1}{N}\sum_{i=1}^{N}\frac{dQ}{ds} \tag{10.38}$$

$$\frac{d\sigma}{ds} = \left[\sum_{i=1}^{N}(Q\frac{dQ}{ds}) - N\mu\frac{d\mu}{ds}\right]/(\sigma(N-1)) \tag{10.39}$$

[1] MVFOSM is exact for linear functions with Gaussian inputs, but quickly degrades for nonlinear and/or non-Gaussian.
[2] Analytic and numerical Hessians, when available, are instantaneous with no accumulation rate concerns.

This enables design sensitivities for mean, standard deviation or variance (based on `final_moments` type), and forward/inverse reliability index mappings ($\bar{z} \rightarrow \beta$, $\bar{\beta} \rightarrow z$).

Bibliography

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 1965. 28, 36, 47

[2] B. M. Adams, W. J. Bohnhoff, R. A. Canfield, K. R. Dalbey, M. S. Ebeida, J. P. Eddy, M. S. Eldred, G. Geraci, R. W. Hooper, P. D. Hough, K. T. Hu, J. D. Jakeman, K. Carson, M. Khalil, K. A. Maupin, J. A. Monschke, E. M. Ridgway, A. A. Rushdi, D. T. Seidl, J. A. Stephens, L. P. Swiler, A. Tran, D. M. Vigil, T. M. Wildey, J. G. Winokur, and (with Menhorn, F. and Zeng, X.). *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.12 Reference Manual*. Sandia National Laboratories, Albuquerque, NM, May 2020. Available online from http://dakota.sandia.gov/documentation.html. 40, 49, 69, 89

[3] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, M. S. Ebeida, J. P. Eddy, M. S. Eldred, R. W. Hooper, P. D. Hough, K. T. Hu, J. D. Jakeman, M. Khalil, K. A. Maupin, J. A. Monschke, E. M. Ridgway, A. A. Rushdi, D. T. Seidl, J. A. Stephens, L. P. Swiler, and J. G. Winokur. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.12 users manual. Technical Report SAND2020-5001, Sandia National Laboratories, Albuquerque, NM, May 2020. Available online from http://dakota.sandia.gov/documentation.html. 48, 51, 85

[4] N. Agarwal and N.R. Aluru. A domain adaptive stochastic collocation approach for analysis of MEMS under uncertainties. *Journal of Computational Physics*, 228:7662–7688, 2009. 30

[5] N. M. Alexandrov, R. M. Lewis, C. R. Gumbert, L. L. Green, and P. A. Newman. Optimization with variable-fidelity models applied to wing design. In *Proceedings of the 38th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2000. AIAA Paper 2000-0841. 87, 88, 89

[6] M. Allen and K. Maute. Reliability-based design optimization of aeroelastic structures. *Struct. Multidiscip. O.*, 27:228–242, 2004. 108

[7] R. Askey and J. Wilson. Some basic hypergeometric polynomials that generalize jacobi polynomials. In *Mem. Amer. Math. Soc. 319*, Providence, RI, 1985. AMS. 27

[8] R.G. Baraniuk, V. Cevher, M.F. Duarte, and C. Hegde. Model-based compressive sensing. *Information Theory, IEEE Transactions on*, 56(4):1982–2001, 2010. 42

[9] V. Barthelmann, E. Novak, and K. Ritter. High dimensional polynomial interpolation on sparse grids. *Adv. Comput. Math.*, 12(4):273–288, 2000. Multivariate polynomial interpolation. 38

[10] C. Berg, J. Mateu, and E. Porcu. The dagum family of isotropic correlation functions. *Bernoulli*, 14(4):1134–1149, 2008. 75

[11] J.-P. Berrut and L. N. Trefethen. Barycentric lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004. 29

[12] B. J. Bichon, M. S. Eldred, L. P. Swiler, S. Mahadevan, and J. M. McFarland. Multimodal reliability assessment for complex engineering applications using efficient global optimization. In *Proceedings of the 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (9th AIAA Non-Deterministic Approaches Conference)*, number AIAA-2007-1946, Honolulu, HI, April 2007. 24

[13] Graud Blatman and Bruno Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics*, 230(6):2345 – 2367, 2011. 40

[14] G. E. P. Box and D. R. Cox. An analysis of transformations. *J. Royal Stat. Soc., Series B*, 26:211–252, 1964. 19, 36

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. 40

[16] K. Breitung. Asymptotic approximation for multinormal integrals. *J. Eng. Mech., ASCE*, 110(3):357–366, 1984. 22

[17] J. Burkardt. The "combining coefficient" for anisotropic sparse grids. Technical report, Virginia Tech, Blacksburg, VA, 2009. http://people.sc.fsu.edu/~jburkardt/presentations/sgmga_coefficient.pdf. 39

[18] J. Burkardt. Computing the hermite interpolation polynomial. Technical report, Florida State University, Gainesville, FL, 2009. http://people.sc.fsu.edu/~jburkardt/presentations/hermite_interpolant.pdf. 29

[19] T. Butler, J. D. Jakeman, and T. M. Wildey. A consistent bayesian formulation for stochastic inverse problems based on push-forward measures. *arXiv*, math.NA:1704.00680, 2017. 71, 72

[20] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995. 84

[21] M. R. Celis, J. .E. Dennis, and R. .A Tapia. A trust region strategy for nonlinear equality constrained optimization. In P. .T. Boggs, R. H. Byrd, and R. B. Schnabel, editors, *Numerical Optimization 1984*, pages 71–82. SIAM, Philadelphia, USA, 1985. 89

[22] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):129–159, January 2001. 40

[23] X. Chen and N.C. Lind. Fast probability integration by three-parameter normal tail approximation. *Struct. Saf.*, 1:269–276, 1983. 19

[24] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization, SIAM-MPS, Philadelphia, 2000. 90

[25] P. G. Constantine. *Active subspaces: Emerging ideas for dimension reduction in parameter studies*, volume 2. SIAM, 2015. 101, 102, 103

[26] P. G. Constantine, E. Dow, and Q. Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014. 102

[27] P. G. Constantine, M. S. Eldred, and E. T. Phipps. Sparse pseudospectral approximation method. *Computer Methods in Applied Mechanics and Engineering*, 229–232:1–12, July 2012. 39

[28] P. G. Constantine and D. F. Gleich. Computing active subspaces. *arXiv*, math.NA:1408.0545, 2014. 102

[29] P. G. Constantine, D. F. Gleich, and G. Iaccarino. Spectral methods for parameterized matrix equations. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2681–2699, 2010. 38

[30] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013. 98, 99

[31] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. A Wiley-Interscience publication. Wiley, 2006. 65, 70

[32] N. Cressie. *Statistics of Spatial Data*. John Wiley and Sons, New York, 1991. 73, 96

[33] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. Active reward learning. In *Robotics: Science and Systems*, 2014. 98

[34] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constructive Approximation*, 13:57–98, 1997. 10.1007/BF02678430. 40

[35] G.J. Davis and M.D. Morris. Six factors which affect the condition number of matrices associated with kriging. *Mathematical geology*, 29(5):669–683, 1997. 75

[36] A. Der Kiureghian and P. L. Liu. Structural reliability under incomplete information. *J. Eng. Mech., ASCE*, 112(EM-1):85–104, 1986. 19, 28, 36

[37] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, 2014. 98, 99

[38] A. Dey and S. Mahadevan. Ductile structural system reliability analysis using adaptive importance sampling. *Structural Safety*, 20:137–154, 1998. 24

[39] D.L. Donoho and Y. Tsaig. Fast solution of ℓ_1-norm minimization problems when the solution may be sparse. *Information Theory, IEEE Transactions on*, 54(11):4789 –4812, nov. 2008. 41

[40] A . Doostan and H. Owhadi. A non-adapted sparse approximation of PDEs with stochastic inputs. *Journal of Computational Physics*, 230(8):3015 – 3034, 2011. 40

[41] X. Du and W. Chen. Sequential optimization and reliability assessment method for efficient probabilistic design. *J. Mech. Design*, 126:225–233, 2004. 109, 113

[42] Marco F. Duarte, Michael B. Wakin, and Richard G. Baraniuk. Fast reconstruction of piecewise smooth signals from random projections. In *Online Proceedings of the Workshop on Signal Processing with Adaptative Sparse Structured Representations (SPARS)*, Rennes, France, 2005. 42

[43] R.P. Dwight and Z. Han. Efficient uncertainty quantification using gradient-enhanced kriging. In *11th AIAA Non-Deterministic Approaches Conference*, Reston, VA, USA, May 2009. American Institute of Aeronautics and Astronautics. 80

[44] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004. 41

[45] M. S. Eldred, H. Agarwal, V. M. Perez, S. F. Wojtkiewicz, Jr., and J. E. Renaud. Investigation of reliability method formulations in DAKOTA/UQ. *Structure & Infrastructure Engineering: Maintenance, Management, Life-Cycle Design & Performance*, 3(3):199–213, 2007. 20, 23, 24, 107, 109, 113

[46] M. S. Eldred and B. J. Bichon. Second-order reliability formulations in DAKOTA/UQ. In *Proceedings of the 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, number AIAA-2006-1828, Newport, RI, May 1–4 2006. 20, 23, 107, 109, 113

[47] M. S. Eldred and J. Burkardt. Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA-2009-0976, Orlando, FL, January 5–8, 2009. 37, 38, 39

[48] M. S. Eldred and D. M. Dunlavy. Formulations for surrogate-based optimization with data fit, multifidelity, and reduced-order models. In *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, number AIAA-2006-7117, Portsmouth, VA, September 6–8 2006. 113

[49] M. S. Eldred, A. A. Giunta, and S. S. Collis. Second-order corrections for surrogate-based optimization with model hierarchies. In *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY,, Aug. 30–Sept. 1, 2004. AIAA Paper 2004-4457. 114

[50] M. S. Eldred, A. A. Giunta, S. F. Wojtkiewicz, Jr., and T. G. Trucano. Formulations for surrogate-based optimization under uncertainty. In *Proc. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, number AIAA-2002-5585, Atlanta, GA, September 4–6, 2002. 108

[51] M. S. Eldred, C. G. Webster, and P. Constantine. Evaluation of non-intrusive approaches for wiener-askey generalized polynomial chaos. In *Proceedings of the 10th AIAA Non-Deterministic Approaches Conference*, number AIAA-2008-1892, Schaumburg, IL, April 7–10 2008. 28

[52] M.S. Eldred, B.M. Adams, D.M. Gay, L.P. Swiler, K. Haskell, W.J. Bohnhoff, J.P. Eddy, W.E. Hart, J.P Watson, J.D. Griffin, P.D. Hough, T.G. Kolda, P.J. Williams, and M.L. Martinez-Canales. Dakota version 4.1 users manual. Sandia Technical Report SAND2006-6337, Sandia National Laboratories, Albuquerque, NM, 2007. 75

[53] G. M. Fadel, M. F. Riley, and J.-F. M. Barthelemy. Two point exponential approximation method for structural optimization. *Structural Optimization*, 2(2):117–124, 1990. 21

[54] J. M. Flegal and G. L. Jones. Batch means and spectral variance estimators in Markov chain Monte Carlo. *The Annals of Statistics*, 38(2):1034–1070, 2010. 59, 60

[55] R. Fletcher, S. Leyffer, and P. L. Toint. On the global convergence of a filter-SQP algorithm. *SIAM J. Optim.*, 13(1):44–59, 2002. 89, 90

[56] P. Frauenfelder, C. Schwab, and R. A. Todor. Finite elements for elliptic problems with stochastic coefficients. *Comput. Methods Appl. Mech. Engrg.*, 194(2-5):205–228, 2005. 38

[57] J. Gablonsky. Direct version 2.0 userguide technical report. Technical Report CRSC-TR01-08, North Carolina State University, Center for Research in Scientific Computation, Raleigh, NC, 2001. 97

[58] S. Gao, G. Ver Steeg, and A Galstyan. Efficient estimation of mutual information for strongly dependent variables. *CoRR*, abs/1411.2003, 2014. 68

[59] W. Gautschi. *Orthogonal Polynomials: Computation and Approximation*. Oxford University Press, New York, 2004. 28

[60] G. Geraci, G. Iaccarino, and Michael S. Eldred. A multi fidelity control variate approach for the multilevel monte carlo technique. *CTR Annual Research Briefs 2015*, pages 169–181, 2015. 15

[61] T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numer. Algorithms*, 18(3-4):209–232, 1998. 38

[62] T. Gerstner and M. Griebel. Dimension-adaptive tensor-product quadrature. *Computing*, 71(1):65–87, 2003. 50

[63] Michael B. Giles. Multilevel Monte Carlo Path Simulation. *Operations Research*, 56(3):607–617, June 2008. 12

[64] Michael B. Giles. Multilevel monte carlo methods. *Acta Numerica*, 24:259328, 2015. 12

[65] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. User's guide for NPSOL (Version 4.0): A Fortran package for nonlinear programming. Technical Report TR SOL-86-2, System Optimization Laboratory, Stanford University, Stanford, CA, 1986. 23

[66] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, San Diego, CA, 1981. 88

[67] A. A. Giunta and M. S. Eldred. Implementation of a trust region model management strategy in the DAKOTA optimization toolkit. In *Proc. 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, number AIAA-2000-4935, Long Beach, CA, September 6–8, 2000. 109

[68] T. Gneiting, M.G. Genton, and P. Guttorp. Geostatistical space-time models, stationarity, separability and full symmetry. In B. Finkenstadt, L. Held, and V. Isham, editors, *Statistical Methods for Spatio-Temporal Systems*, chapter 4, pages 151–172. Boca Raton: Chapman & Hall/CRC, 2007. 75

[69] G. H.. Golub and J. H. Welsch. Caclulation of gauss quadrature rules. *Mathematics of Computation*, 23(106):221–230, 1969. 28

[70] A. Haldar and S. Mahadevan. *Probability, Reliability, and Statistical Methods in Engineering Design*. Wiley, New York, 2000. 17, 18, 23

[71] T. Hastie, R. Tibshirani, and J.H. Friedman. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. New York: Springer-Verlag, 2001. 42

[72] N. J. Higham. The numerical stability of barycentric lagrange interpolation. *IMA Journal of Numerical Analysis*, 24(4):547–556, 2004. 29

[73] M. Hohenbichler and R. Rackwitz. Sensitivity and importance measures in structural reliability. *Civil Eng. Syst.*, 3:203–209, 1986. 108

[74] M. Hohenbichler and R. Rackwitz. Improvement of second-order reliability estimates by importance sampling. *J. Eng. Mech., ASCE*, 114(12):2195–2199, 1988. 22

[75] H.P. Hong. Simple approximations for improving second-order reliability estimates. *J. Eng. Mech., ASCE*, 125(5):592–595, 1999. 22

[76] S. Hosder, R. W. Walters, and M. Balch. Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables. In *Proceedings of the 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, number AIAA-2007-1939, Honolulu, HI, April 23–26, 2007. 40

[77] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34:441–466, 2006. 95, 97

[78] E. T. Jaynes and G. Larry. Bretthorst. *Probability theory : the logic of science*. Cambridge University Press, Cambridge, UK; New York, NY, 2003. 55

[79] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998. 24, 95, 97

[80] A. Karamchandani and C. A. Cornell. Sensitivity estimation within first and second order reliability methods. *Struct. Saf.*, 11:95–107, 1992. 108

[81] M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society*, 63:425–464, 2001. 61

[82] A. Klimke and B. Wohlmuth. Algorithm 847: spinterp: Piecewise multilinear hierarchical sparse grid interpolation in matlab. *ACM Transactions on Mathematical Software*, 31(4):561–579, 2005. 30

[83] W. A. Klimke. *Uncertainty Modeling using Fuzzy Arithmetic and Sparse Grids*. PhD thesis, Universität Stuttgart, Stuttgart, Germany, 2005. 34

[84] D. P. Kouri, D. Ridzal, B. G. van Bloeman Waanders, and G. von Winckel. Rapid optimization library. Technical Report SAND2014-19572, Sandia National Laboratories, Albuquerque, NM, 2014. 84

[85] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004. 68

[86] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964. 98

[87] C. La and M.N. Do. Tree-based orthogonal matching pursuit algorithm for signal reconstruction. In *Image Processing, 2006 IEEE International Conference on*, pages 1277–1280, 2006. 42

[88] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice–Hall, 1974. 91

[89] Allison Lewis, Ralph Smith, Brian Williams, and Victor Figueroa. An information theoretic approach to use high-fidelity codes to calibrate low-fidelity codes. *Journal of Computational Physics*, 324:24 – 43, 2016. 68

[90] C. Lucas. Lapack-style codes for level 2 and 3 pivoted cholesky factorizations. Numerical Analysis Reports 442, Manchester Centre for Computational Mathematics, Manchester, England, February 2004. LAPACK Working Note 161. 80

[91] W. Luo and B. Li. Combining eigenvalues and variation of eigenvectors for order determination. 2015. 104

[92] J.D. Martin. Computational improvements to estimating kriging metamodel parameters. *Journal of Mechanical Design*, 131:084501, 2009. 75

[93] 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. 74

[94] J. C. Meza, R. A. Oliva, P. D. Hough, and P. J. Williams. OPT++: an object oriented toolkit for nonlinear optimization. *ACM Transactions on Mathematical Software*, 33(2), 2007. 23

[95] A. Narayan and D. Xiu. Stochastic collocation methods on unstructured grids in high dimensions via interpolation. *SIAM J. Scientific Computing*, 34(3), 2012. 46

[96] J. Neter, W. Wasserman, and M. H. Kutner. *Applied Linear Statistical Models*. Irwin Professional Publishing, Burr Ridge, IL, 2nd edition, 1985. 85

[97] L. W. T. Ng and M. S. Eldred. Multifidelity uncertainty quantification using nonintrusive polynomial chaos and stochastic collocation. In *Proceedings of the 14th AIAA Non-Deterministic Approaches Conference*, number AIAA-2012-1852, Honolulu, HI, April 23-26, 2012. 51

[98] L. W. T. Ng. and K. E. Willcox. Multifidelity approaches for optimization under uncertainty. *International Journal for numerical methods in Engineering*, 100(10):746–772, 2014. 10

[99] F. Nobile, R. Tempone, and C. G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. Technical Report Technical report TRITA-NA 2007:7, Royal Institute of Technology, Stockholm, Sweden, 2007. 38

[100] F. Nobile, R. Tempone, and C. G. Webster. An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. on Num. Anal.*, 46(5):2411–2442, 2008. 38

[101] J. Nocedal and Wright S. J. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, 1999. 88, 113

[102] E. O. Omojokun. *Trust Region Algorithms for Optimization with Nonlinear Equality and Inequality Constraints*. PhD thesis, University of Colorado, Boulder, Colorado, 1989. 89

[103] M.R. Osborne, B. Presnell, and B.A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000. 41

[104] R. Pasupathy, M. Taaffe, B. W. Schmeiser, and W. Wang. Control-variate estimation using estimated control means. *IIE Transactions*, 44(5):381–385, 2014. 10

[105] V. M. Pérez, M. S. Eldred, , and J. E. Renaud. Solving the infeasible trust-region problem using approximations. In *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, NY, Aug. 30–Sept. 1, 2004. AIAA Paper 2004-4312. 89, 91

[106] V. M. Pérez, J. E. Renaud, and L. T. Watson. An interior-point sequential approximation optimization methodology. *Structural and Multidisciplinary Optimization*, 27(5):360–370, July 2004. 88, 89, 91, 93

[107] Fernando Pérez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *2008 IEEE international symposium on information theory*, pages 1666–1670. IEEE, 2008. 70

[108] N. Petra, J. Martin, G. Stadler, and O. Ghattas. A computational framework for infinite-dimensional bayesian inverse problems; part II: Stochastic newton mcmc with application to ice sheet flow inverse problems. *SIAM J. Sci. Comput.*, 36(4):A1525–A1555, 2014. 57

[109] Michele Pisaroni, Sebastian Krumscheid, and Fabio Nobile. Mathicse technical report : Quantifying uncertain system outputs via the multilevel monte carlo method - part 1: Central moment estimation. 2017. MATHICSE Technical Report Nr. 23.2017 October 2017. 13

[110] R. Rackwitz. Optimization and risk acceptability based on the Life Quality Index. *Struct. Saf*, 24:297–331, 2002. 108

[111] R. Rackwitz and B. Fiessler. Structural reliability under combined random load sequences. *Comput. Struct.*, 9:489–494, 1978. 19

[112] P. Ranjan, Bingham D., and Michailidis G. Sequential experiment design for contour estimation from complex computer codes. *Technometrics*, 50(4):527–541, 2008. 25

[113] M. T. Reagan, H. N. Najm, P. P. Pebay, O. M. Knio, and R. G. Ghanem. Quantifying uncertainty in chemical systems modeling. *Int. J. Chem. Kinet.*, 37:368–382, 2005. 47

[114] G. Rennen. Subset selection from large datasets for kriging modeling. *Structural and Multidisciplinary Optimization*, 38(6):545–569, 2009. 75

[115] N. L. Robertson, M. Khalil, and B. M. Adams. Comparing MCMC diagnostics and stopping rules. In A. Cangi and M. L. Parks, editors, *Center for Computing Research Summer Proceedings 2018*, pages 132–144. Sandia National Laboratories, 2018. Technical Report SAND2019-5093R. 60

[116] J. F. Rodriguez, J. E. Renaud, and L. T. Watson. Convergence of trust region augmented lagrangian methods using variable fidelity approximation data. *Structural Optimization*, 15:1–7, 1998. 87

[117] M. Rosenblatt. Remarks on a multivariate transformation. *Annals of Mathematical Statistics*, 23(3):470–472, 1952. 19, 36

[118] J. Sacks, S. B. Schiller, and W. Welch. Design for computer experiments. *Technometrics*, 31:41–47, 1989. 96

[119] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. 70

[120] I.C. Simpson. Numerical integration over a semi-infinite interval, using the lognormal distibution. *Numerische Mathematik*, 31:71–76, 1978. 28

[121] S.A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk SSSR*, 4:240–243, 1963. 38

[122] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009. 98

[123] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012. 98

[124] A. Stroud. *Approximate Calculation of Multiple Integrals*. Prentice Hall, 1971. 39

[125] G. Tang, G. Iaccarino, and M. S Eldred. Global sensitivity analysis for stochastic collocation expansion. In *Proceedings of the 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (12th AIAA Non-Deterministic Approaches conference)*, Orlando, FL, April 12-15, 2010. AIAA Paper 2010-XXXX. 48

[126] M.A. Tatang. *Direct incorporation of uncertainty in chemical and environmental engineering systems*. PhD thesis, MIT, 1995. 40

[127] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996. 41

[128] Ramakrishna Tipireddy and Roger Ghanem. Basis adaptation in homogeneous chaos spaces. *Journal of Computational Physics*, 259:304 – 317, 2014. 101

[129] Anh Tran, Jing Sun, John M Furlan, Krishnan V Pagalthivarthi, Robert J Visintainer, and Yan Wang. pBO-2GP-3B: A batch parallel known/unknown constrained Bayesian optimization with feasibility classification and its applications in computational fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 347:827–852, 2019. 98, 99

[130] J. Tu, K. K. Choi, and Y. H. Park. A new study on reliability-based design optimization. *J. Mech. Design*, 121:557–564, 1999. 19

[131] A. van der Sluis. Condition numbers and equilibration of matrices. *Numerische Mathematik*, 14(1):14–23, 1969. 80

[132] G. N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design: With Applications*. McGraw-Hill, New York, 1984. 88, 90

[133] S. N. Walsh, T. M. Wildey, and J. D. Jakeman. Optimal experimental design using a consistent bayesian approach. *arXiv*, stat.CO:1705.09395, 2017. 71

[134] R. W. Walters. Towards stochastic fluid mechanics via polynomial chaos. In *Proceedings of the 41st AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA-2003-0413, Reno, NV, January 6–9, 2003. 40

[135] G. W. Wasilkowski and H. Woźniakowski. Explicit cost bounds of algorithms for multivariate tensor product problems. *Journal of Complexity*, 11:1–56, 1995. 38

[136] N. Wiener. The homogoeneous chaos. *Amer. J. Math.*, 60:897–936, 1938. 27

[137] J. A. S. Witteveen and H. Bijl. Modeling arbitrary uncertainties using gram-schmidt polynomial chaos. In *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA-2006-0896, Reno, NV, January 9–12 2006. 28

[138] 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. 109, 113

[139] Y.-T. Wu and P.H. Wirsching. A new algorithm for structural reliability estimation. *J. Eng. Mech., ASCE*, 113:1319–1336, 1987. 19

[140] B. A. Wujek and J. E. Renaud. New adaptive move-limit management strategy for approximate optimization, part 1. *AIAA Journal*, 36(10):1911–1921, 1998. 89

[141] B. A. Wujek and J. E. Renaud. New adaptive move-limit management strategy for approximate optimization, part 2. *AIAA Journal*, 36(10):1922–1934, 1998. 89

[142] D. Xiu. Numerical integration formulas of degree two. *Applied Numerical Mathematics*, 58:1515–1520, 2008. 39

[143] D. Xiu and J.S. Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM J. Sci. Comput.*, 27(3):1118–1139 (electronic), 2005. 38

[144] S. Xu and R. V. Grandhi. Effective two-point function approximation for design optimization. *AIAA J.*, 36(12):2269–2275, 1998. 21

[145] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. 41

[146] T. Zou, S. Mahadevan, and R. Rebba. Computational efficiency in reliability-based optimization. In *Proceedings of the 9th ASCE Specialty Conference on Probabilistic Mechanics and Structural Reliability*, Albuquerque, NM, July 26–28, 2004. 109

[147] T. Zou, Z. Mourelatos, S. Mahadevan, and P. Meernik. Reliability analysis of automotive body-door subsystem. *Rel. Eng. and Sys. Safety*, 78:315–324, 2002. 24