University of New Mexico

Doctoral Thesis

Advanced Methods in Stochastic Collocation for Polynomial Chaos in RAVEN

Author:
Paul W. Talbot

Supervisor:
Dr. Anil K. Prinja

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Engineering

in the

Department of Nuclear Engineering

August 2016

Abstract

As experiment complexity in fields such as nuclear engineering continues to increase, so does the demand for robust computational methods to simulate them. In many simulations, input design parameters as well as intrinsic experiment properties are sources of input uncertainty. Often, small perturbations in uncertain parameters have significant impact on the experiment outcome. For instance, when considering nuclear fuel performance, small changes in the fuel thermal conductivity can greatly affect the maximum stress on the surrounding cladding. The difficulty of quantifying input uncertainty impact in such systems has grown with the complexity of the numerical models. Traditionally, uncertainty quantification has been approached using random sampling methods like Monte Carlo. For some models, the input parametric space and corresponding quantity-of-interest output space is sufficiently explored with a few low-cost computational calculations. For other models, it is computationally costly to obtain a good understanding of the output space.

To combat the costliness of random sampling, this research explores the possibilities of advanced methods in stochastic collocation for generalized polynomial chaos (SCgPC) as an alternative to traditional uncertainty quantification techniques such as Monte Carlo (MC) and Latin Hypercube sampling (LHS) methods. In this proposal we explore the behavior of traditional SCgPC construction strategies, as well as truncated polynomial spaces using total degree (TD) and hyperbolic cross (HC) construction strategies. We also consider applying anisotropy to the polynomial space, and analyze methods whereby the level of anisotropy can be approximated. We review and develop potential adaptive polynomial construction strategies. Finally, we add high-dimension model reduction (HDMR) expansions, using SCgPC representations for the constituent terms, and consider adaptive methods to construct them. We analyze these methods on a series of models of increasing complexity. We primarily use analytic methods of various means, and finally demonstrate on an engineering-scale neutron transport problem. For this analysis, we demonstrate the application of the algorithms discussed above in `raven`, a production-level uncertainty quantification framework.

Finally, we propose additional work in enhancing the current implementations of SCgPC and HDMR.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In simulation modeling, we attempt to capture the behavior of a physical system by describing it in a series of equations, often partial differential equations. These equations may be time-dependent, and capture physics of interest for understanding the system. A *solver* is then written that can solve the series of equations and determine quantities of interest (QoI). A traditional solver accepts a set of inputs and produces a set of single-valued outputs. For instance, a solver might solve equations related to the attenuation of a beam of photons through a material, and the QoI might be the strength of the beam exiting the material. A single run of the solver usually results in a single value, or realization, of the quantity of interest.

This single realization might be misleading, however. In most systems there is some degree of uncertainty in the input parameters to the solver. Some of these uncertainties may be epistemic, or systematic uncertainty originating with inexact measurements or measurable unknowns. Other uncertainties might be aleatoric, intrinsic uncertainty in the system itself, such as probabilistic interactions or random motion. Taken together, the input parameter uncertainties exist within a multidimensional probabilistic space. While some points in that space may be more likely than others, the possible range of values for the QoI is only understood when the uncertain input space is considered as a whole. We note here that while it is possible that some of the input parameters are correlated in their probabilistic distribution, it is also possible to decouple them into uncorrelated variables. Throughout this work we will assume the input parameters are uncorrelated.

One traditional method for exploring the uncertain input space is through random sampling, such as in analog Monte Carlo sampling. In this method, a point in the input space is chosen at random based on probability. This point represents values for the input parameters to the solver. The solver is executed with these inputs, and the QoIs

are collected. Then, another point in the input space is chosen at random. This process continues until the properties of the QoIs, or *response*, are well understood.

There are some beneficial properties to random sampling approaches like Monte Carlo. Significantly, they are unintrusive: there is no need to modify the solver in order to use these methods. This allows a framework of algorithms to be developed which know only the input space and QoI of a solver, but need no further knowledge about its operation. Unintrusive methods are desirable because the uncertainty quantification algorithms can be developed and maintained separately from the solver.

Monte Carlo and similar sampling strategies are relatively slow to converge on the response surface. For example, with Monte Carlo sampling, in order to reduce the standard error of the mean of the response by a factor of two, it is necessary to take at least four times as many samples. If a solver is sufficiently computationally inexpensive, running additional solutions is not a large concern; however, for lengthy and expensive solvers, it may not be practical to obtain sufficient realizations to obtain a clear response.

In this work, we will assume solvers are computationally expensive, requiring many hours per solve, and that computational resource availability requires as few solves as possible. As such, we consider several methodologies for quantifying the uncertainty in expensive solver calculations. In order to demonstrate clearly the function of these methods, we apply them first on several simpler problems, such as polynomial evaluations and analytic attenuation. These models have a high degree of regularity, and their analyticity provides for straightforward benchmarking. Through gradual increasing complexity, we investigate the behavior of the UQ methods.

Finally, we apply the methods to an engineering-scale solver that models the neutronics and performance of nuclear fuel. This will demonstrate the practical application of the uncertainty quantification methods, where the regularity and other properties of the model are not well understood.

The first uncertainty quantification method we consider is traditional analog Monte Carlo (MC) analysis, wherein random sampling of the input space generates a view of the response. MC is used as a benchmark methodology; if other methods converge on moments of the quantities of interest more quickly and consistently than MC, we consider them "better" for our purposes.

The second method we consider is stochastic collocation for generalized polynomial chaos (SCgPC)[21, 32–34], whereby deterministic collocation points are used to develop a polynomial-interpolated reduced-order model of the response as a function of the inputs. This method algorithmically expands the solver as the sum of orthogonal multi-dimensional polynomials with scalar coefficients. The scalar coefficients are obtained by

numerical integration using multidimensional collocation (quadrature) points. The chief distinction between SCgPC and Monte Carlo methods is that SCgPC is deterministic, in that the realizations required from the solver are predetermined instead of randomly sampled. There are two major classes of deterministic uncertainty quantification methods: intrusive and unintrusive. Like Monte Carlo, SCgPC is unintrusive and performs well without any need to access the operation of the solver. This behavior is desirable for construction black-box approach algorithms for uncertainty quantification. Other intrusive methods such as stochastic Galerkin exist [15], but require solver modification to operate. This makes them solver-dependent and undesirable for an independent uncertainty quantification framework.

The other methods we present here expand on SCgPC. First, we introduce non-tensor-product methods for determining the set of polynomial bases to use in the expansion. Because a tensor product grows exponentially with increasing cardinality of the input space, we combat this curse of dimensionality using the alternative polynomial set construction methods[23]. These bases will then be used to construct Smolyak-like sparse grids [24] to provide collocation points that in turn calculate the coefficients in the polynomial expansion. Second, we consider anisotropic sparse grids, allowing higher-order polynomials for particular input parameters. We also consider methods for obtaining weights that determine the level of anisotropic preference to give parameters, and explore the effects of a variety of anisotropic choices.

The second method group we consider is high-dimension model representation (HDMR), which correlates with Sobol decomposition [30]. This method is useful both for developing sensitivities of the quantity of interest to subsets of the input space, as well as constructing a reduced-order model itself. We demonstrate the strength of HDMR as a method to inform anisotropic sensitivity weights for SCgPC.

Finally, we consider adaptive algorithms to construct both SCgPC and HDMR expansions using second-moment convergence criteria. We analyze these for potential efficiencies and shortcomings. We also propose future work to further improve the adaptive methods.

We implement all these methods in Idaho National Laboratory's `raven`[19] uncertainty quantification framework. `raven` is a Python-written framework that non-intrusively provides tools for analysts to quantify the uncertainty in their simulations with minimal development. To demonstrate the application of the method developed, we use a complex non-linear multiphysics system solver simulating the operation of a fuel pin within a nuclear reactor core, including both neutronics and fuel performance physics kernals. For this solver, we use the coupled `rattleSNake`[20] and `bison` [16, 18] production

codes. Both of these codes are developed and maintained within the `moose`[17] environment. The multiphysics nonlinear system provides a challenge with unknown response properties for the uncertainty quantification methods discussed in this proposal.

The remainder of this work will proceed as follows:

- Chapter 2: We describe the analytic test problems and engineering-scale problem solved by the simulations we will be running, along with their properties and inferences about the algorithms developed. We discuss potential approaches to model solving and applications of the models.
- Chapter 3: We describe methods for uncertainty quantification, including Monte Carlo (MC), stochastic collocation for generalized Polynomial Chaos (SCgPC), and high-dimension model reduction (HDMR). We additionally describe adaptive methods for SCgPC and HDMR.
- Chapter 4: We analyze results obtained for the various UQ methods on analytic models, and contrast them with traditional Monte Carlo convergence on statistical moments.
- Chapter 5: We perform analysis on the engineering-scale multiphysics coupled problem, and analyze results.
- Chapter 6: We draw conclusions from the evaluations performed, and offer some suggestions for applicability as well as future development for the UQ methods demonstrated.

Chapter 2

Models

2.1 Introduction to Models

In this section we present the models used in demonstrating the uncertainty quantification (UQ) methods in this work. We use the term *model* to describe any code with uncertain inputs and a set of at least one response quantity of interest. We include a variety of models with the intent to demonstrate the strengths and weaknesses of each UQ method.

Firstly, we include several analytic models. These are models who have an exact, derivable value for statistical moments or sensitivities. They are simpler in mechanics than full engineering-scale problems, and offer a way to benchmark the performance of the UQ methods.

Secondly, we include an engineering-scale multiphysics application. There are no analytic response values in this model, only a nominal case with no uncertainty included in the input space. Demonstration of the performance of UQ methods on this model will highlight the practical application of each method.

We describe each model in turn. Throughout the models, we will describe them using the syntax

$$u(Y) = Q, \tag{2.1}$$

where $u(Y)$ is the model as a function of N uncertain inputs $Y = (y_1, \ldots, y_N)$ and Q is a single-valued response quantity of interest.

2.2 Tensor Monomials

The simplest model we make use of is a first-order tensor polynomial (tensor monomial) combination **??**. The mathematical expression is

$$u(Y) = \prod_{n=1}^{N} (y_n + 1). \tag{2.2}$$

For example, for $N = 3$ we have

$$u(Y) = y_1 y_2 y_3 + y_1 y_2 + y_1 y_3 + y_2 y_3 + y_1 + y_2 + y_3 + 1. \tag{2.3}$$

For this model we distribute the uncertain inputs in several ways: uniformly on [-1,1], uniformly on [0,1], and normally on $[\mu, \sigma]$. A summary of analytic statistics is given in Table 2.1.

Distribution	Mean	Variance
$\mathcal{U}[-1,1]$	1	$\left(\frac{4}{3}\right)^N - 1$
$\mathcal{U}[0,1]$	$\left(\frac{3}{4}\right)^N$	$\left(\frac{7}{3}\right)^N - \left(\frac{3}{4}\right)^{2N}$
$\mathcal{N}[\mu,\sigma]$	$\prod_{n=1}^{N}(\mu_{y_n}+1)$	$\prod_{n=1}^{N}[(\mu_{y_n}+1)^2+\sigma_{y_n}^2] - \prod_{n=1}^{N}(\mu_{y_n}+1)^2$

TABLE 2.1: Analytic Expressions for Tensor Monomial Case

2.3 Sudret Polynomial

The polynomial used by Sudret in his work [40] is another tensor-like polynomial, and is a test case traditionally used to identify convergence on sensitivity parameters. The mathematical expression is

$$u(Y) = \frac{1}{2^N} \prod_{n=1}^{N} (3y_n^2 + 1). \tag{2.4}$$

The variables are distributed uniformly on [0,1]. The statistical moments and sensitivities are given in Table 2.2, where $\mathcal{S}_n$ is the global Sobol sensitivity of $u(Y)$ to perturbations in y_n.

Statistic	Expression
Mean	1
Variance	$\left(\frac{6}{5}\right)^N - 1$
$\mathcal{S}_n$	$\frac{5^{-n}}{(6/5)^N - 1}$

TABLE 2.2: Analytic Expressions for Sudret Case

2.4 Attenuation

This model represents an idealized single-dimension system where an beam of particles impinges on a purely-absorbing material with total scaled length of 1. The response of interest is the fraction of particles exiting the opposite side of the material. The material is divided into N segments, each of which has a distinct uncertain absorption cross section y_n. The solution takes the form

$$u(Y) = \prod_{n=1}^{N} \exp(-y_n/N). \tag{2.5}$$

Because negative cross sections have dubious physical meaning, we restrict the distribution cases to uniform on [0,1] as well as normally-distributed on $[\mu, \sigma]$. A summary of analytic statistics is given in Table 2.3.

Distribution	Mean	Variance
$\mathcal{U}[0,1]$	$\left[N\left(1-e^{-1/N}\right)\right]^N$	$\left[\frac{N}{2}\left(1-e^{-2/N}\right)\right]^N - \left[N\left(1-e^{-1/N}\right)\right]^{2N}$
$\mathcal{N}[\mu,\sigma]$	$\prod_{n=1}^{N} \exp\left[\frac{\sigma_{y_n}^2}{2N^2} - \frac{\mu_{y_n}}{N}\right]$	$\prod_{n=1}^{N} \exp\left[\frac{2\sigma_{y_n}^2}{N^2} - \frac{2\mu_{y_n}}{N}\right]$

TABLE 2.3: Analytic Expressions for Attenuation Case

This model has some interesting properties to demonstrate performance of polynomial-based UQ methods. First, because the solution is a product of exponential functions, it cannot be exactly represented by a finite number of polynomials. Second, the Taylor development of the exponential function includes all increasing polynomial orders. The product of several exponential functions is effectively a tensor combination of polynomials for each dimension.

2.5 Gaussian Peak

Similar to the attenuation model, the Gaussian peak [41] instead uses square arguments to the exponential function. A tuning parameter a can be used to change the peakedness of the function. Increased peakedness leads to more difficult polynomial representation. A location parameter μ can be used to change the location of the peak. The mathematical expression is

$$u(Y) = \exp\left(-\sum_{n=1}^{N} a^2(y_n - \mu)^2\right). \tag{2.6}$$

We allow each y_n to vary uniformly on [0,1]. A summary of analytic statistics is given in Table 2.4.

Statistic	Expression
Mean	$\left(\frac{\sqrt{\pi}}{2a}(\operatorname{erf}(a\mu)+\operatorname{erf}(a-a\mu))\right)^N$
Variance	$\left(\frac{\sqrt{\pi/2}}{2a}(\operatorname{erf}(a\mu\sqrt{2})-\operatorname{erf}(a\sqrt{2}(1-\mu)))\right)^N-\left(\frac{\sqrt{\pi}}{2a}(\operatorname{erf}(a\mu)+\operatorname{erf}(a-a\mu))\right)^{2N}$

TABLE 2.4: Analytic Expressions for Gaussian Peak Case

This case offers particular challenge because of its Taylor development, which only includes even powers of the uncertain parameters. This suggests added difficulty in successive representation, especially for an adaptive algorithm.

2.6 Ishigami Function

The Ishigami function [42] is a commonly-used function in performing sensitivity analysis. It is given by

$$u(Y) = \sin y_1 + a\sin^2 y_2 + by_3^4 \sin(y_1). \tag{2.7}$$

In our case, we will use $a = 7$ and $b = 0.1$ as in [43]. In particular interest for this model are its strong nonlinearity and lack of independence for y_3, as it only appears in conjunction with y_1. The analytic statistics of interest for this model are in Table 2.5, where D_n is the partial variance contributed by y_n and Sobol sensitivities $\mathcal{S}_n$ are obtained by dividing D_n by the total variance.

Statistic	Expression	Approx. Value
Mean	$\frac{7}{2}$	3.5
Variance	$\frac{a^2}{8}+\frac{b\pi^4}{5}+\frac{b^2\pi^8}{18}+\frac{1}{2}$	13.84459
D_1	$\frac{b\pi^4}{5}+\frac{b^2\pi^8}{50}+\frac{1}{2}$	4.34589
D_2	$\frac{a^2}{8}$	6.125
$D_{1,3}$	$\frac{8b^2\pi^8}{225}$	3.3737
$D_3, D_{1,2}, D_{2,3}, D_{1,2,3}$	0	0

TABLE 2.5: Analytic Expressions for Ishigami Case

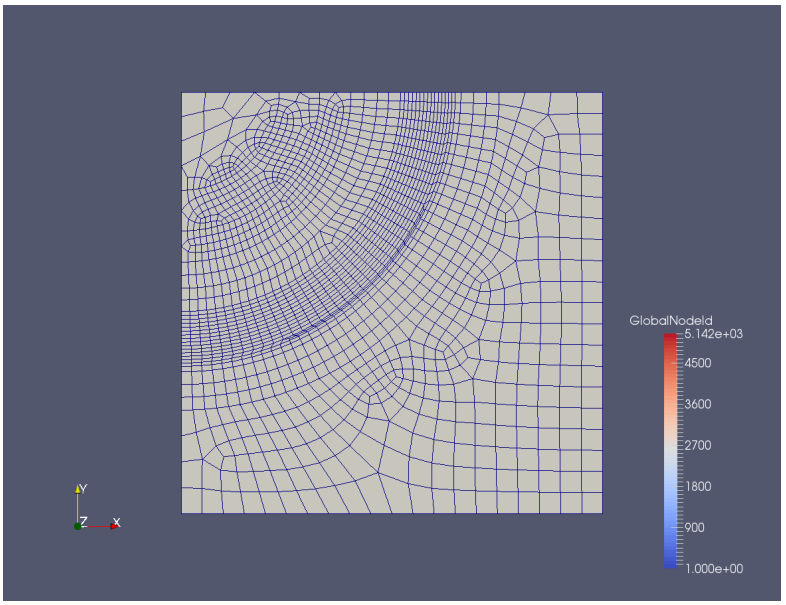
2.7 Sobol G-Function

2.8 Anisotropic Polynomial

2.9 Pin Cell

This model is a coupled multiphysics engineering-scale model. It simulates fuel behavior through the depletion of fissile material in a single two-dimensional slice of a fuel rod. The problem domain contains the fuel, gap, clad, and moderator, and represents a symmetric quarter pin. The depletion steps are carried out through a year-long burn cycle. The coupled multiphysics are neutronics, handled by `rattleSNake`, and fuel performance, handled by `bison`.

The mesh is shown in Fig. **??**. TODO BETTER FIGURE. The mesh contains 20 bands of fuel blocks, the gap, the clad, and the moderator. TODO Use colors to describe locations. TODO dimensions.



The neutronics is calculated using 8 energy groups and takes as uncertain inputs 671 material cross sections, including fission, capture, scattering, and neutron multiplication factor Each cross section is perturbed by 10% of its original value, distributed normally. The scattering cross sections for each material in each group are not perturbed individually; rather, a scattering scaling factor for each group is determined, and the scattering cross sections are all scaled by that factor. The fission and capture cross sections are perturbed individually. The critical output of the neutronics calculation is power shapes for use in the fuel performance code, as well as the k-eigenvalue for the rod.

The fuel performance code models mechanics such as heat conduction in the fuel, clad, and gap, clad stresses, grain radius growth, and fuel expansion through the depletion steps of the fuel. The code takes power shapes from the neutronics code as input and produces several key characteristics, such as peak clad temperature, maximum fuel centerline temperature, and maximum clad stress.

The uncertain input space is highly correlated, so a Karhunen-Loeve (KL) component analysis is performed as the first step in a two-part reduction [38]. The covariance matrix is obtained via cross section construction in `scale`[39] using random sampling. Table 2.6 gives the first several eigenvalues in the KL expansion. Surrogate (or *latent*) dimensions identified by the KL expansion will be used as input variables for demonstration of the various UQ methods.

Index	Eigenvalue
1	0.974489839965
2	0.0183147250746
3	0.00271597405394
4	0.00260939137165
5	0.000486257522596
6	0.000431957645049
7	0.000253683187786
8	0.000228044411204
9	0.000124030638175
10	7.14328494102e-05
11	6.30833696364e-05
12	3.87071149672e-05
13	3.51066363934e-05
14	2.48699823434e-05
15	1.98915286765e-05
16	1.35985387253e-05
17	1.128896325e-05
18	9.59426898684e-06
19	8.11612567548e-06
20	7.16508951777e-06
21	6.53366817241e-06
22	4.50006575957e-06
23	4.19287192651e-06
24	3.7671309151e-06
25	2.61683536224e-06
26	2.22099981728e-06
27	1.6360971709e-06
28	1.13245742809e-06
29	9.92282537141e-07

TABLE 2.6: KL Expansion Eigenvalues for Pin Cell Problem

Chapter 3

Methods

3.1 Introduction

In this chapter we describe various common uncertainty quantification methods and their applications. We begin by discussing the principles of input spaces and responses, and define terminology used in this work. Next we discuss uncertainty quantification at a high level, and describe several common uncertainty quantification tools. Finally, we explore generalized polynomial chaos expansion, stochastic collocation, and high-density model reduction as advanced uncertainty quantification techniques.

Many simulation models are algorithms constructed to solve partial differential equations, often in two or three dimensions and possibly time. The inputs to these models include boundary conditions, material properties, tuning parameters, and so forth. The outputs are quantities of interest, either data fields or scalar values. The outputs are used to inform decision-making processes. In general, we allow $u(Y)$ to be the model u as a function of the input space $Y = (y_1, \ldots, y_n, \ldots, y_N)$ where y_n is a single input parameter to the model, n is an index spanning the number of inputs, and N is the total number of inputs. We signify the output response quantity as Q, and assume it to be a scalar integrated quantity. Our generic model takes the form

$$u(Y) = Q. \tag{3.1}$$

Essential to using simulation models is understanding the possibility of important uncertainties existing in the inputs. These could be aleatoric uncertainties due to intrinsic randomness in the inputs, or epistemic uncertainties due to model imperfections or lack of knowledge. Each of these uncertainties has some distribution defining the likelihood

of an input to have a particular value. These distributions might be assumed or constructed from experiment; for our work, we will assume given distributions are accurate. The input likelihood distribution is the probability distribution function (PDF) $\rho_n(y_n)$. We require

$$\int_a^b \rho_n(y_n) d\ y_n = 1, \tag{3.2}$$

where a and b are the minimum and maximum values y_n can take (possibly infinite).

When there are more than one uncertain input, the combination of distributions for these inputs span an uncertainty space Ω. TODO make sure this terminology is right. The dimensionality of Ω is N, the number of uncertain input variables. The probability of any point in the input space occurring is given by the join-probability distribution $\rho(Y)$, still enforcing

$$\int_{a_1}^{b_1} \cdots \int_{a_N}^{b_N} \rho(Y) dy_1 \cdots dy_N = 1. \tag{3.3}$$

. For clarity, we define multidimensional integral operator

$$\int_\Omega (\cdot) dY \equiv \int_{a_1}^{b_1} \cdots \int_{a_N}^{b_N} (\cdot) dy_1 \cdots dy_N, \tag{3.4}$$

so that Eq. 3.3 can be written

$$\int_\Omega \rho(Y) dY = 1. \tag{3.5}$$

We note the possibility that multiple inputs may be correlated with each other. When inputs are not independent, the joint probability distribution is not the product of each individual probability distribution distribution. Using principle component analysis (sometimes known as Karhunen-Loeve expansion [37]), however, a surrogate orthogonal input space can be constructed. As a result, we only consider independent variables in this work.

3.2 Uncertainty Quantification

The purpose of uncertainty quantification is to propagate the uncertainties present in the input space of a problem through the model and comprehend their effects on the output responses. Often response uncertainty is quantified in terms of moments. Second-order uncertainty quantification seeks for the mean and variance of the perturbed response. In general, the mean of a model is the first moment,

$$\text{mean} = \mathbb{E}[u(Y)] = \int_\Omega \rho(Y) u(Y) dY, \tag{3.6}$$

and the variance is the second moment less the square of the first,

$$\text{variance} = \mathbb{E}[u(Y)^2] - \mathbb{E}[u(Y)]^2 = \int_\Omega \rho(Y)u(Y)^2 dY - \text{mean}^2. \tag{3.7}$$

Another use for uncertainty quantification is understanding the sensitivity of the output responses to the uncertain inputs; that is, determining how responses change as a function of changes in the input space. At the most primitive level, linear sensitivity of a response mean to an input is the derivative of the response with respect to the input. Sensitivities can be both local to a region in the input space as well as global to the entire problem.

There are several common tools used for uncertainty quantification when analytic analysis is not possible. These include stochastic methods such as Monte Carlo sampling, deterministic methods such as Grid sampling, and mixed methods such as Latin Hypercube sampling (LHS).

3.2.1 Monte Carlo

The Monte Carlo method [9] has been used formally since the 1930s as a tool to explore possible outcomes in uncertain models. Nuclear physicist Enrico Fermi used the method in his work with neutron moderation in Rome [10]. In its simplest form, Monte Carlo involves randomly picking realizations from a set of possibilities, then statistically collecting the results. In uncertainty quantification, Monte Carlo can be used to sample points in the input space based on the joint probability distribution. The collection of points is analyzed to determine the moments of the response.

The mean of a response is determined using the unweighted average of samples collected:

$$\mathbb{E}[u(Y)] = \frac{1}{N}\sum_{m=1}^{M}(u(Y_m)) + \epsilon_M^{\text{MC}}, \tag{3.8}$$

where Y_m is a realization randomly chosen based on $rho(Y)$, and M is the total number of samples taken. The error in the approximation diminishes with the root of the number of samples taken,

$$\epsilon_M^{\text{MC}} \propto \frac{1}{\sqrt{M}}. \tag{3.9}$$

The second moment is similarly approximated as

$$\mathbb{E}[u(Y)^2] \approx \frac{1}{N}\sum_{m=1}^{M}\left(u(Y_m)^2\right). \tag{3.10}$$

The standard deviation (root of the variance) converges similarly to the mean for Monte Carlo methods. There are many tools that can be used to improve Monte Carlo sampling [11][12]; we restrict our discussion to traditional analog Monte Carlo sampling.

Monte Carlo has long been a gold standard for uncertainty quantification because of its consistency. Monte Carlo will always resolve the response statistics given a sufficient number of samples. Additionally, the convergence of Monte Carlo is largely agnostic of the input space dimensionality, a feature not shared by the LHS and Grid sampling methods.

The drawback to Monte Carlo sampling also centers on its consistency. The error in analog Monte Carlo can only be consistently reduced by drastically increasing the number of samples calculated. While coarse estimates are inexpensive to obtain, high precision takes a great deal of runs to converge.

TODO 2d, 3d grid example

3.2.2 Grid

One of the drawbacks of Monte Carlo is lack of control over points sampled. While LHS can improve this somewhat, another alternative is using a structured orthogonal grid. In this strategy, the input space is divided into hypervolumes that are equal in volume either in the input space or in uncertainty space. For demonstration, we first consider a one-dimensional case with a single normally-distributed variable y with mean μ and standard deviation σ. If the input space is divided into equal volumes in the input space, a lower and upper bound are determined, then nodes are selected on the ends and equally spaced throughout. See Figure TODO. If the input space is divided into equal probability volumes, nodes are selected to be equidistant along the cumulative distribution function (CDF). This assures that the volume between each set of nodes has equal probability. See Figure TODO. TODO Figure should show both equal spacing and CDF spacing, plus a normal distribution for comparison. In higher dimensions, a grid is constructed as a tensor product of each grid.

Since the grid nodes are user-defined, approximating integrals are slightly more complicated than in the Monte Carlo space. The mean is approximated by

$$\mathbb{E}[u(Y)] = \int_{\Omega} \rho(Y)u(Y)dY \approx \sum_{m=1}^{M} w_m u(Y_m), \tag{3.11}$$

where m iterates over each node in the grid, Y_m is the multidimensional input point as node m, and w_m is a probability weight determined by the volume of probability

represented by the node. In grids constructed by CDF, all w_m are of the same value, while in grids spaced equally by value, w_m can vary significantly. Similarly, the second moment is approximated by

$$\mathbb{E}[u(Y)^2] = \int_\Omega \rho(Y)u(Y)^2 dY \approx \sum_{m=1}^{M} w_m u(Y_m)^2. \tag{3.12}$$

An advantage to grid sampling is its regular construction, which can give more clarity to how a response behaves throughout the input space. However, the grid construction suffers greatly from the curse of dimensionality, which makes it inefficient for input spaces with large dimensionality. TODO references TODO 2d, 3d grid example

3.2.3 LHS

A cross between Monte Carlo and Grid sampling strategies, the Latin Hypercube Sampling (LHS) strategy has long been a sampling tool used to reduce the total samples needed without significantly sacrificing integration quality [13]. In LHS, the input space is also divided into a grid just as in the Grid sampling strategy. However, unlike Grid sampling, only one sample is taken per hyperplane; that is, for any of the input variables, there is only one sample taken between each of the one-dimensional nodes. Once a hypervolume is selected to take a sample, the exact point is selected by random sampling in the probability space within the hypervolume. See for example the sampling in Figure TODO.

As in the Grid method, the weight of each sample is the probability volume of the hypervolume it represents.

3.3 Generalized Polynomial Chaos

Expanding beyond the traditional uncertainty quantification methods of Monte Carlo, Grid, and LHS sampling, there are more advanced methods that are quite efficient in particular applications. Polynomial chaos expansion (PCE) methods, for example, seek to interpolate the simulation code as a combination of polynomials of varying degree in each dimension of the input space. There are several advantage to expanding in polynomials. First, orthonormal polynomials have means and standard deviations that are trivial to calculate analytically, even for computer algorithms. Second, the resulting polynomial expansion is an inexpensive surrogate model that can be used in place of the

original. Third, the unknowns in the expansions are scalar coefficients, which can often be efficiently calculated through numerical integration.

Originally Wiener proposed expanding in Hermite polynomials for Gaussian-normal distributed variables [14]. Askey and Wilson generalized Hermite polynomials to include Jacobi polynomials, including Legendre and Laguerre polynomials [22]. Xiu and Karniadakis combined these concepts to perform PCE for a range of Gaussian-based distributions with corresponding polynomials, including Legendre polynomials for uniform distributions, Laguerre polynomials for Gamma distributions, and Jacobi polynomials for Beta distributions [21].

In each of these cases, a probability-weighted integral over the distribution can be cast in a way that the corresponding polynomials are orthogonal over the same weight and interval. These chaos Wiener-Askey polynomials were used by Xiu and Karniadakis to develop the generalized polynomial chaos expansion method (gPC), including a transformation for applying the same method to arbitrary distributions (as long as they have a known inverse CDF) [21]. Two significant methodologies have grown from gPC application. The first makes use of Lagrange polynomials to expand the original function or simulation code, as Lagrange polynomials can be made orthogonal over the same domain as the distributions [29]; the other uses the Wiener-Askey polynomials [21]. We consider the latter in this work.

We consider a simulation code that produces a quantity of interest u as a function $u(Y)$ whose arguments are the uncertain, distributed input parameters $Y = (Y_1, \ldots, Y_n, \ldots, Y_N)$. A particular realization ω of Y_n is expressed by $Y_n(\omega)$, and a single realization of the entire input space results in a solution to the function as $u(Y(\omega))$. We acknowledge obtaining a realization of $u(Y)$ may take considerable computation time and effort, and may be solved nonlinearly. There also may be other input parameters that contribute to the solution of $u(Y)$; we neglect these, as our interest is in the uncertainty space; all parameters without uncertainty are held at their nominal values. In addition, it is possible that the quantity of interest $u(Y)$ is an integrated quantity or some norm of a value that is temporally or spatially distributed. We restrict $u(Y(\omega))$ to a single scalar output, but the same principles apply to a multidimensional response. Further, a quantity of interest may be time-dependent in a transient simulation. In this case, the PCE can be constructed at several selected points in time throughout the simulation, which can then be interpolated between. In effect, the polynomial coefficients become time-dependent scalar values. For now, we consider a static case with no time dependence.

We expand $u(Y)$ in orthonormal multidimensional polynomials $\Phi_k(Y)$, where k is a multi-index tracking the polynomial order in each axis of the polynomial Hilbert space,

and $\Phi_k(Y)$ is constructed as

$$\Phi_k(Y) = \prod_{n=1}^{N} \phi_{k_n}(Y_n), \tag{3.13}$$

where $\phi_{k_n}(Y_n)$ is a single-dimension Wiener-Askey orthonormal polynomial of order k_n and $k = (k_1, \ldots, k_n, \ldots, k_N)$, $k_n \in \mathbb{N}^0$. For example, given $u(y_1, y_2, y_3)$, $k = (2, 1, 4)$ is the multi-index of the product of a second-order polynomial in y_1, a first-order polynomial in y_2, and a fourth-order polynomial in y_4. The gPC for $u(Y)$ using this notation is

$$u(Y) \approx \sum_{k \in \Lambda(L)} u_k \Phi_k(Y), \tag{3.14}$$

where u_k is a scalar weighting polynomial coefficient. The polynomials used in the expansion are determined by the set of multi-indices Λ, which can be selected in a variety of ways we will discuss in section 3.3.1 and are the essence of this work. In the limit that Λ contains all possible combinations of polynomials of any order, Eq. 3.13 is exact. Practically, however, Λ is truncated to some finite set of combinations, discussed in section 3.3.1.

Using the orthonormal properties of the Wiener-Askey polynomials,

$$\int_{\Omega} \Phi_k(Y)\Phi_{\hat{k}}(Y)\rho(Y)dY = \delta_{k\hat{k}}, \tag{3.15}$$

where $\rho(Y)$ is the combined PDF of Y, Ω is the multidimensional domain of Y, and δ_{nm} is the Dirac delta, we can isolate an expression for the polynomial expansion coefficients. We multiply both sides of Eq. 3.13 by $\Phi_{\hat{k}}(Y)$, integrate both sides over the probability-weighted input domain, and sum over all $\hat{k}$ to obtain the coefficients, sometimes referred to as polynomial expansion moments,

$$u_k = \frac{\langle u(Y)\Phi_k(Y)\rangle}{\langle \Phi_k(Y)^2\rangle}, \tag{3.16}$$

$$= \langle u(Y)\Phi_k(Y)\rangle, \tag{3.17}$$

where we use the angled bracket notation to denote the probability-weighted inner product,

$$\langle f(Y)\rangle \equiv \int_{\Omega} f(Y)\rho(Y)dY. \tag{3.18}$$

When $u(Y)$ has an analytic form, these coefficients can be solved by integration; however, in general other methods must be applied to numerically perform the integral. While tools such as Monte Carlo integration can be used to evaluate the integral, we can harness the properties of Gaussian quadratures because of the probability weights and domain. This stochastic collocation method is discussed in section 3.4.

3.3.1 Polynomial Index Set Construction

The chief concern in expanding a function in interpolating multidimensional polynomials is choosing appropriate polynomials to make up the expansion. There are many generic ways by which a polynomial set can be constructed. Here we present three static approaches: tensor product, total degree, and hyperbolic cross.

In the nominal tensor product case, $\Lambda(L)$ contains all possible combinations of polynomial indices up to truncation order L in each dimension, as

$$\Lambda_{\mathrm{TP}}(L) = \left\{\bar{p} = (p_1, \cdots, p_N) : \max_{1 \le n \le N} p_n \le L\right\}. \tag{3.19}$$

The cardinality of this index set is $|\Lambda_{\mathrm{TP}}(L)| = (L+1)^N$. For example, for a two-dimensional input space (N=2) and truncation limit $L = 3$, the index set $\Lambda_{\mathrm{TP}}(3)$ is given in Table 3.1, where the notation $(1, 2)$ signifies the product of a polynomial that is first order in Y_1 and second order in Y_2.

(3,0)	(3,1)	(3,2)	(3,3)
(2,0)	(2,1)	(2,2)	(2,3)
(1,0)	(1,1)	(1,2)	(1,3)
(0,0)	(0,1)	(0,2)	(0,3)

TABLE 3.1: Tensor Product Index Set, $N = 2, L = 3$

It is evident there is some inefficiencies in this index set. First, it suffers dramatically from the *curse of dimensionality*; that is, the number of polynomials required grows exponentially with increasing dimensions. Second, the total order of polynomials is not considered. Assuming the contribution of each higher-order polynomial is smaller than lower-order polynomials, the (3,3) term is contributing sixth-order corrections that are likely smaller than the error introduced by ignoring fourth-order corrections (4,0) and (0,4). This leads to the development of the *total degree* (TD) and *hyperbolic cross* (HC) polynomial index set construction strategies [23].

In TD, only multidimensional polynomials whose *total* order at most L are permitted,

$$\Lambda_{\mathrm{TD}}(L) = \left\{\bar{p} = (p_1, \cdots, p_N) : \sum_{n=1}^{N} p_n \le L\right\}. \tag{3.20}$$

The cardinality of this index set is $|\Lambda_{\mathrm{TD}}(L)| = \binom{L+N}{N}$, which grows with increasing dimensions much more slowly than TP. For the same $N = 2, L = 3$ case above, the TD index set is given in Table 3.2.

In HC, the *product* of polynomial orders is used to restrict allowed polynomials in the index set. This tends to polarize the expansion, emphasizing higher-order polynomials

(3,0)			
(2,0)	(2,1)		
(1,0)	(1,1)	(1,2)	
(0,0)	(0,1)	(0,2)	(0,3)

TABLE 3.2: Total Degree Index Set, $N = 2, L = 3$

in each dimension but lower-order polynomials in combinations of dimensions, as

$$\Lambda_{\mathrm{HC}}(L) = \left\{\bar{p} = (p_1, \ldots, p_N) : \prod_{n=1}^{N} p_n + 1 \leq L + 1\right\}. \tag{3.21}$$

The cardinality of this index set is bounded by $|\Lambda_{\mathrm{HC}}(L)| \leq (L+1)(1+\log(L+1))^{N-1}$. It grows even more slowly than TD with increasing dimension, as shown in Table 3.3 for $N = 2, L = 3$.

(3,0)			
(2,0)			
(1,0)	(1,1)		
(0,0)	(0,1)	(0,2)	(0,3)

TABLE 3.3: Hyperbolic Cross Index Set, $N = 2, L = 3$

It has been shown that the effectiveness of TD and HC as index set choices depends strongly on the regularity of the responce [23]. TD tends to be most effective for infinitely-continuous response surfaces, while HC is more effective for surfaces with limited smoothness or discontinuities.

3.3.2 Anisotropy

While using TD or HC to construct the polynomial index set combats the curse of dimensionality present in TP, it is not eliminated and continues to be an issue for problems of large dimensionality. Another method that can be applied to mitigate this issue is index set anisotropy, or the unequal treatment of various dimensions. In this strategy, weighting factors $\alpha = (\alpha_1, \ldots, \alpha_n, \ldots, \alpha_N)$ are applied in each dimension to allow additional polynomials in some dimensions and less in others. This change adjusts the TD and HC construction rules as follows, where $|\alpha|_1$ is the one-norm of α.

$$\tilde{\Lambda}_{\mathrm{TD}}(L) = \left\{\bar{p} = (p_1, \cdots, p_N) : \sum_{n=1}^{N} \alpha_n p_n \leq |\alpha|_1 L\right\}, \tag{3.22}$$

$$\tilde{\Lambda}_{\mathrm{HC}}(L) = \left\{\bar{p} = (p_1, \cdots, p_N) : \prod_{n=1}^{N} (p_n + 1)^{\alpha_n} \leq (L+1)^{|\alpha|_1}\right\}. \tag{3.23}$$

As it is desirable to obtain the isotropic case from a reduction of the anisotropic cases, we define the one-norm for the weights as

$$|\alpha|_1 = \frac{\sum_{n=1}^{N} \alpha_n}{N}. \tag{3.24}$$

Considering the same case above ($N = 2, L = 3$), we apply weights $\alpha_1 = 5, \alpha_2 = 3$, and the resulting index sets are Tables 3.4 (TD) and 3.5 (HC).

(2,0)				
(1,0)	(1,1)	(1,2)		
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)

TABLE 3.4: Anisotropic Total Degree Index Set, $N = 2, L = 3$

(1,0)			
(0,0)	(0,1)	(0,2)	(0,3)

TABLE 3.5: Anisotropic Hyperbolic Cross Index Set, $N = 2, L = 3$

There are many methods by which anisotropy weights can be assigned. Often, if a problem is well-known to an analyst, it may be enough to use heuristics to assign importance arbitrarily. Otherwise, a smaller uncertainty quantification solve can be used to roughly determine sensitivity coefficients (such as Pearson coefficients), and the inverse of those can then be applied as anisotropy weights. Sobol coefficients obtained from first- or second-order HDMR, a proposed development for this work, could also serve as a basis for these weights. A good choice of anisotropy weight can greatly speed up convergence; however, a poor choice can slow convergence considerably, as computational resources are used to resolve low-importance dimensions.

3.3.3 Polynomial Expansion Features

As previously mentioned, there are several benefits to the PCE once constructed. First, the PCE is a surrogate model for the original response, and can be used in its place as long as all the inputs are within the same bounds as when the original PCE was constructed. The error in this representation will be of the same order as the truncation error of the expansion.

Second, the first and second moments of the PCE are very easy to obtain. Because the probability-weighted integral of all the orthonormal polynomials is zero with the exception of the zeroth-order polynomial, and using the notation

$$u(Y) \approx \tilde{u}(Y) \equiv \sum_{k \in \lambda} u_k \Phi_k(Y), \tag{3.25}$$

the mean is simply

$$\begin{aligned}\mathbb{E}[\tilde{u}(Y)] &= \int_{\Omega} \rho(Y) \sum_{k\in\Lambda} u_k \Phi_k(Y) dY, \\ &= u_{(0,\cdots,0)}. \end{aligned} \tag{3.26}$$

The second moment is similarly straightforward. The integral of the square of the PCE involves cross-products of all the expansion terms; however, because the integral of the product of any two polynomials is the dirac delta $\delta_{i,j}$, this simplifies to the sum of the squares of the expansion coefficients,

$$\begin{aligned}\mathbb{E}[\tilde{u}(Y)^2] &= \int_{\Omega} \rho(Y) \left[\sum_{k\in\Lambda} u_k \Phi_k(Y)\right]^2 dY, \\ &= \int_{\Omega} \rho(Y) \sum_{k_1\in\Lambda} \sum_{k_2\in\Lambda} u_{k_1}\Phi_{k_1}(Y) \cdot u_{k_2}\Phi_{k_2}(Y) dY, \\ &= \sum_{k_1\in\Lambda} \sum_{k_2\in\Lambda} u_{k_1} \cdot u_{k_2} \delta_{k_1,k_2}, \\ &= \sum_{k\in\Lambda} u_k^2. \end{aligned} \tag{3.27}$$

3.4 Stochastic Collocation

Stochastic collocation is the process of using collocated points to approximate integrals of stochastic space numerically. In particular we consider using Gaussian quadratures (Legendre, Hermite, Laguerre, and Jacobi) corresponding to the polynomial expansion polynomials for numerical integration. Quadrature integration takes the form

$$\int_a^b f(x)\rho(x) = \sum_{\ell=1}^{\infty} w_\ell f(x_\ell), \tag{3.28}$$

$$\approx \sum_{\ell=1}^{\hat{L}} w_\ell f(x_\ell), \tag{3.29}$$

where w_ℓ, x_ℓ are corresponding points and weights belonging to the quadrature set, truncated at order $\hat{L}$. At this point, this $\hat{L}$ should not be confused with the polynomial expansion truncation order L. We can simplify this expression using the operator notation

$$q^{(\hat{L})}[f(x)] \equiv \sum_{\ell=1}^{\hat{L}} w_\ell f(x_\ell). \tag{3.30}$$

A nominal multidimensional quadrature is the tensor product of individual quadrature weights and points, and can be written

$$Q^{(\mathbf{L})} = q_1^{(\hat{L}_1)} \otimes q_2^{(\hat{L}_2)} \otimes \cdots, \tag{3.31}$$

$$= \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}. \tag{3.32}$$

It is worth noting each quadrature may have distinct points and weights; they need not be constructed using the same quadrature rule. In general, one-dimensional Gaussian quadrature excels in exactly integrating polynomials of order $2p-1$ using p points and weights; equivalently, it requires $(p+1)/2$ points to integrate an order p polynomial. For convenience we repeat here the coefficient integral we desire to evaluate, Eq. 3.16.

$$u_k = \langle u(Y)\Phi_k(Y)\rangle. \tag{3.33}$$

We can approximate this integral with the appropriate Gaussian quadrature as

$$u_k \approx Q^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)], \tag{3.34}$$

where we use bold vector notation to note the order of each individual quadrature, $\hat{\mathbf{L}} = [\hat{L}_1, \ldots, \hat{L}_n, \ldots, \hat{L}_N]$. For clarity, we remove the bold notation and assume a one-dimensional problem, which extrapolates as expected into the multidimensional case.

$$u_k \approx q^{(\hat{L})}[u(Y)\Phi_k(Y)], \tag{3.35}$$

$$= \sum_{\ell=1}^{\hat{L}} w_\ell u(Y_\ell)\Phi_k(Y_\ell). \tag{3.36}$$

In order to determine the quadrature order $\hat{L}$ needed to accurately integrate this expression, we consider the gPC formulation for $u(Y)$ in Eq. 3.13 and replace it in the sum,

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell \Phi_k(Y_\ell) \sum_{k\in\Lambda(L)} u_{\hat{k}}\Phi_{\hat{k}}(Y_\ell). \tag{3.37}$$

Using orthogonal properties of the polynomials, this reduces as $\hat{L} \to \infty$ to

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell u_k \Phi_k(Y_\ell)^2. \tag{3.38}$$

Thus, the integral, to the same error introduced by truncating the gPC expansion, the quadrature is approximating an integral of order $2k$. As a result, the quadrature order

should be order

$$p = \frac{2k+1}{2} = k + \frac{1}{2} < k + 1, \tag{3.39}$$

so we can conservatively use $p = k + 1$. In the case of the largest polynomials with order $k = L$, the quadrature size $\hat{L}$ is the same as $L + 1$. It is worth noting that if $u(Y)$ is effectively of much higher-order polynomial than L, this equality for quadrature order does not hold true; however, it also means that gPC of order L will be a poor approximation.

While a tensor product of highest-necessary quadrature orders could serve as a suitable multidimensional quadrature set, we can make use of Smolyak-like sparse quadratures to reduce the number of function evaluations necessary for the TD and HC polynomial index set construction strategies.

3.4.1 Smolyak Sparse Grids

Smolyak sparse grids [24] are an attempt to discover the smallest necessary quadrature set to integrate a multidimensional integral with varying orders of predetermined quadrature sets. In our case, the polynomial index sets determine the quadrature orders each one needs in each dimension to be integrated accurately. For example, the polynomial index set point (2,1,3) requires three points in Y_1, two in Y_2, and four in Y_3,or

$$Q^{(2,1,3)} = q_1^{(3)} \otimes q_2^{(2)} \otimes q_3^{(4)}. \tag{3.40}$$

The full tensor grid of all collocation points would be the tensor product of all quadrature for all points, or

$$Q^{(\Lambda(L))} = \bigotimes_{k \in \Lambda} Q^{(k)}. \tag{3.41}$$

Smolyak sparse grids consolidate this tensor form by adding together the points from tensor products of subset quadrature sets. Returning momentarily to a one-dimensional problem, we introduce the notation

$$\Delta_k^{(\hat{L})}[f(x)] \equiv \left(q_k^{(\hat{L})} - q_{k-1}^{(\hat{L})}\right)[f(x)], \tag{3.42}$$

$$q_0^{(\hat{L})}[f(x)] = 0. \tag{3.43}$$

A Smolyak sparse grid is then defined and applied to the desired integral in Eq. 3.16,

$$S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)] = \sum_{k \in \Lambda(L)} \left(\Delta_{k_1}^{(\hat{L}_1)} \otimes \cdots \otimes \Delta_{k_N}^{(\hat{L}_N)}\right)[u(Y)\Phi_k(Y)]. \tag{3.44}$$

Equivalently, and in a more algorithm-friendly approach,

$$S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)] = \sum_{k\in\Lambda(L)} c(k) \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}[u(Y)\Phi_k(Y)] \tag{3.45}$$

where

$$c(k) = \sum_{\substack{j=\{0,1\}^N, \\ k+j\in\Lambda}} (-1)^{|j|_1}, \tag{3.46}$$

using the traditional 1-norm for $|j|_1$. The values for u_k can then be calculated as

$$u_k = \langle u(Y)\Phi_k(Y)\rangle, \tag{3.47}$$

$$\approx S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)]. \tag{3.48}$$

With this numerical method to determine coefficients, we have a complete method for performing SCgPC analysis in an algorithmic manner.

3.5 Adaptive Sparse Grid

One method for improving SCgPC is to construct the polynomial index set adaptively. This effectively constructs anisotropic index sets based on properties of the expansion as it is constructed, instead of in a predetermined way. This method is presented in [26] and used in [7]. The algorithm proceeds generally as follows:

- Begin with the mean (zeroth-order) polynomial expansion.
- While not converged:
  - Collect a list of the polynomial index set whose predecessors have all been evaluated.
  - Predict the impact of adding each polynomial to the existing polynomial index set.
  - If the total impact of all indices is less than tolerance, convergence is reached.
  - Otherwise, add the predicted highest-impact polynomial and loop back.

This adaptive algorithm has the strength of determining the appropriate anisotropy to apply when generating a polynomial index set. For strongly anisotropic cases, or cases where the static index set construction rules are not ideal, the adaptive index set could potentially provide a method to avoid wasted calculations and emphasize high-impact polynomials in the expansion. TODO visual example

There are, however, some weak points in this algorithm. First, the current algorithm has no predictive method to determine the next polynomial index to include in the set; instead, it evaluates each potential index and selects the one with the most impact [7]. This is somewhat inefficient, because of SCgPC representations created that are not used in the final product. One improvement we make to this algorithm is to predict the impact of un-evaluated polynomials based on the impact of predecessors.

In order to predict the most valuable polynomial to add to the expansion during an adaptive search, we first identify a metric for value. Because our interest is in second-order statistics, and the variance of the polynomial expansions is the sum of the polynomial expansion coefficients, we consider the *impact* η_k of a polynomial to be the square of its polynomial expansion coefficient,

$$\eta_k = u_k^2. \tag{3.49}$$

To estimate the impact of a polynomial whose coefficient is not yet calculated, we consider the average of the preceeding polynomials. That is, for a polynomial $k = (3, 2, 4)$ we average the impacts of $(2, 2, 4)$, $(3, 1, 4)$, and $(3, 1, 3)$,

$$\tilde{\eta}_k = \frac{1}{N - j} \sum_{n=1}^{N} \eta_{k - e_n}, \tag{3.50}$$

where $\tilde{\eta}_k$ is the estimated impact of polynomial k, e_n is a unit vector in dimension n, and for every entry where $k - e_n$ would reduce one index to less than 0, it is skipped and j is incremented by one. In this way any polynomial with some missing predecessors is still averaged appropriately with all available information. While occasionally this prediction algorithm may be misled, in general it saves significantly over the previous algorithm.

Another weakness of the adaptive sparse grid algorithm is that there are certain types of models for which the adaptive algorithm will stall, converge too early, or similarly fail. For instance, if the partial derivative of the model with respect to any of the input dimensions is zero when evaluated at the mean point (but nonzero elsewhere), the algorithm will falsely converge prematurely, as adding additional polynomial orders to the input in question will not change the value of the model at the mean point. For example, consider a model

$$f(a, b) = a^3 b^3, \tag{3.51}$$

with both a and b uniformly distributed on [-1,1]. We note the partial derivatives with respect to either input variable evaluated at the central point (0,0) are zero. The first polynomial index set point to evaluate is zeroth-order in each dimension, [0,0]. We distinguish input domain points from polynomial index set points by using parenthesis

for the former and square brackets for the latter. The quadrature point to evaluate this polynomial coefficient is (0,0), which, when evaluated, gives $f(0,0) = 0$. The next polynomial index set combinations are [1,2] and [2,1]. For [1,2], the quadrature points required are $(0,\pm\sqrt{1/3})$. This evaluates to $f(0,\pm\sqrt{1/3}) = 0$, as well. Because of symmetry, we obtain the same result of [2,1]. According to our algorithm, because our old value was 0, and the sum of the new contributions is 0, we have converged; however, we know this is false convergence. While we expect few applications for SCgPC to exhibit these zero partial derivatives in the input space, it is a limitation to be aware of. An argument can be made that, since lower-order polynomials correspond to lower-energy modes of the modeled physics, it is expected that higher-order polynomials should rarely contribute to an accurate expansion unless lower-order polynomials contribute as well.

3.6 High-Dimension Model Representation (HDMR)

While using SCgPC is one method for creating a reduced-order model for a simulation code $u(Y)$, another useful model reduction is HDMR[30], or Sobol decomposition. In particular, we consider Cut-HDMR in this work. In this methodology, the representation of $u(Y)$ is built up from the contributions of subsets of the input space. We consider a reference point realization of Y (nominally the mean of each distribution) $Y^{(0)} = [Y_1^{(0)}, \ldots, Y_N^{(0)}]$ and introduce the notation

$$\hat{Y}_n \equiv [Y_1, \ldots, Y_{n-1}, Y_{n+1}, \ldots, Y_N], \tag{3.52}$$

and

$$u(Y_n)\Big|_{\hat{Y}_n} \equiv u(\hat{Y}_n^{(0)}, Y_n). \tag{3.53}$$

In words, this notation describes holding all the input variables except Y_n constant and only allowing Y_n to vary. Similarly for higher dimensions,

$$u(Y_m, Y_n)\Big|_{\hat{Y}_{m,n}} \equiv u(\hat{Y}_{m,n}^{(0)}, Y_m, Y_n), \tag{3.54}$$

$$= u(Y_1^{(0)}, \ldots, Y_{m-1}^{(0)}, Y_m, Y_{m+1}^{(0)}, \ldots, Y_{n-1}^{(0)}, Y_n, Y_{n+1}^{(0)}, \ldots, Y_N^{(0)}). \tag{3.55}$$

The HDMR reduced-order model is then constructed as

$$\begin{aligned} u(Y) = u_0 &+ \sum_{n=1}^{N} u_{Y_n} \\ &+ \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} u_{Y_{n_1},Y_{n_2}} \\ &+ \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} \sum_{n_3=1}^{n_2-1} u_{Y_{n_1},Y_{n_2},Y_{n_3}} \\ &\cdots \qquad (3.56) \\ &+ u_{Y_{n_1},\cdots,Y_{n_N}}, \qquad (3.57) \end{aligned}$$

where

$$u_0 = u(Y^{(0)}), \qquad (3.58)$$

$$u_{Y_n} = u(Y_n)\Big|_{Y_n} - u_0, \qquad (3.59)$$

$$u_{Y_m,Y_n} = u(Y_m,Y_n)\Big|_{Y_m,Y_n} - u_{Y_m} - u_{Y_n} - u_0, \qquad (3.60)$$

$$\cdots \qquad (3.61)$$

If not truncated, this wastes significant computational effort; however, truncating at second- or third-order interactions can often capture much of the physics with less computation effort than a full solve. It is important to note that the HDMR is not a value itself, but a reduced-order model. To speed up computation using this model, each component term in the HDMR expansion can be replaced in turn by a reduced-order model. In particular, because SCgPC is most effective in calculations involving a small input space, gPC reduced-order models are excellent for representing components of the HDMR expansion.

In addition, Sobol sensitivity indices s can be obtained by taking the ratio of any one expansion term against the remainder of the terms. For first-order sensitivities,

$$s_n = \frac{\mathrm{var}(u_{Y_n})}{\mathrm{var}(u(Y))}, \qquad (3.62)$$

and for second-order,

$$s_{m,n} = \frac{\mathrm{var}(u_{Y_m,Y_n})}{\mathrm{var}(u(Y))}, \qquad (3.63)$$

etc. These sensitivities can be used to inform adaptive SCgPC calculations on the full model.

3.7 Adaptive HDMR

Chapter 4

Analytic Results

4.1 Introduction

In this chapter we present results obtained using stochastic collocation for generalized polynomial chaos expansions (SCgPC) and high-dimension model reduction (HDMR) uncertainty quantification methods. In each case we also include Monte Carlo as a comparison benchmark.

Our primary objective in expanding the usability of collocation-based methods is to reduce the number of computational model solves necessary to obtain reasonable second-order statistics for the model. For each analytic model described in Chapter 3, we present value figures and convergence figures.

Value figures show the values of the mean or standard deviation obtained, along with the benchmark analytic value as a dotted line. The Monte Carlo samples are taken at a few select points. Error bars are provided for the Monte Carlo method and are estimated using the population variance,

$$\epsilon_{95} = \frac{1.96\bar{\sigma}_N}{\sqrt{N}}, \tag{4.1}$$

$$\bar{\sigma}_N^2 = \frac{N}{N-1}\sigma_N^2 = \frac{N}{N-1}\left(\frac{1}{N}\sum_{i=1}^{M} u(Y_i)^2 - \bar{u}(Y)_N^2\right), \tag{4.2}$$

where Y_i are a set of M independent identically-distributed realizations taken from the input space. These errorbars estimate where the value of the statistic is with a probability near 0.95. The estimate of this error improves as additional samples are taken.

Convergence figures are log-log error graphs with the number of computational solves on the x-axis and error with the analytical solution on the y-axis. The distinct lines

demonstrate series of results obtained for each UQ method. Each series obtains additional values by increasing the refinement of the method. For Monte Carlo, additional samples are added. For static SCgPC, higher-order polynomials are used in the representative expansion. For adaptive methods, additional solves are allowed to adaptively include additional polynomials and/or dimension subsets.

The measure of success for a method is less dependent on the absolute value of the error shown. Instead, the rate of convergence as refinement increases determines the desirability of the method for that model. We expect the rate of convergence to depend on two factors: the dimensionality of the uncertain space for the model, and the continuity of the response measured. We consider the convergence of both the mean and the standard deviation for each model.

The series considered include analog Monte Carlo (mc), Tensor Product polynomial expansion construction method (tp), Total Degree polynomial expansion construction method (td), Hyperbolic Cross polynomial expansion construction method (hc), adaptive sparse grid collocation for generalized polynomial chaos expansion (adaptSC), and adaptive HDMR or adaptive Sobol decomposition (adaptSobol).

We additionally note that results from `raven` computations were written to file using 10 digits of accuracy. As a result, any apparent convergence past this level of accuracy is coincidental or the result of machine-exact values, and we consider a relative difference of 10^{-10} to be converged.

4.2 Tensor Monomials

This model is described in section 2.2. As this polynomial contains only combinations of first-order polynomials, we expect the Tensor Product index set construction method (TP) to be very efficient in absolute error magnitude. Because the model has infinite continuity, we expect all collocation-based methods to be quite efficient. The values and errors of the mean and standard deviation are given in Figures 4.5 through 4.8 for 5 uncertain inputs, and the same for 10 dimensions is given in Figures 4.9 through 4.12. Note that TP exactly reproduces the original model with expansion order 1, so no convergence is observed past the initial sampling point.

4.2.1 3 Inputs

The strength of collocation methods is clear for this small-dimensionality problem of three uncertain inputs. The convergence on the mean and standard deviation is swift

for all the methods. Both adaptive methods converge at nearly identical rates; this is not surprising, as for small dimension problems the adaptive search follows a very similar path.

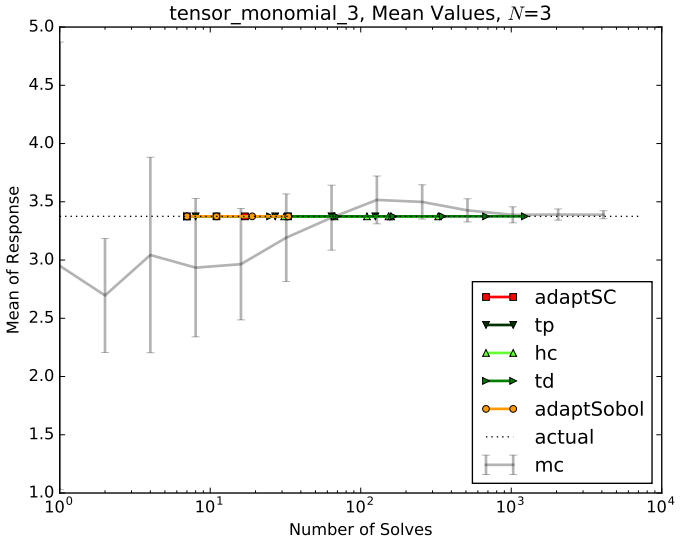


FIGURE 4.1: Tensor Monomial, $N = 3$, Mean Values

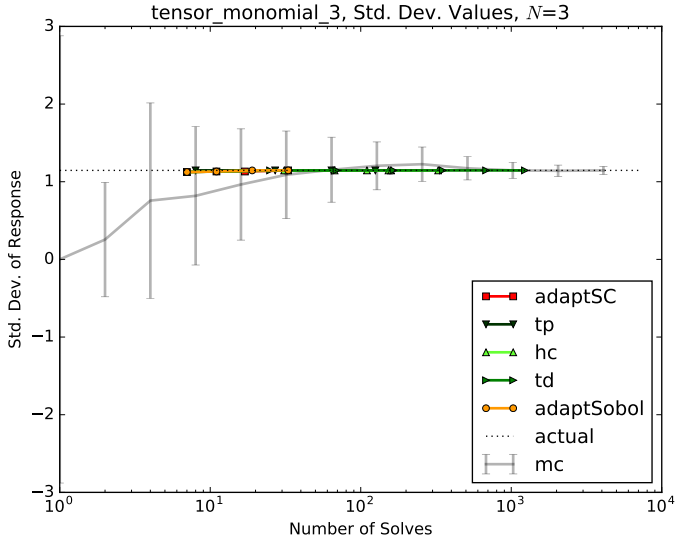


FIGURE 4.2: Tensor Monomial, $N = 3$, Std. Dev. Values

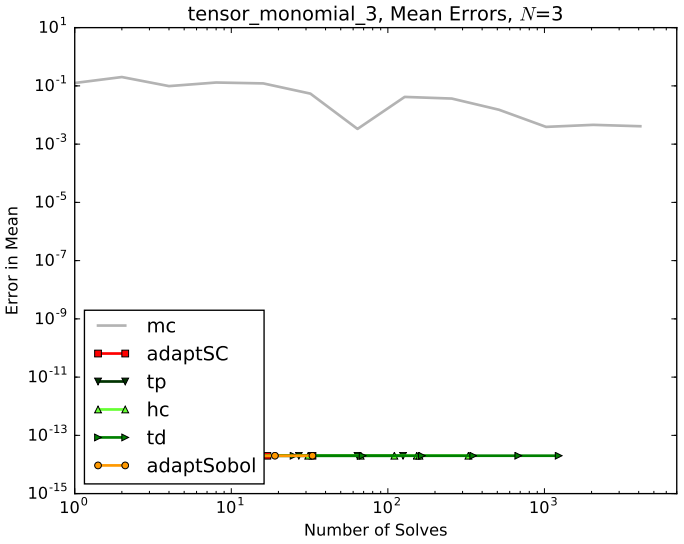


FIGURE 4.3: Tensor Monomial, $N = 3$, Mean Convergence

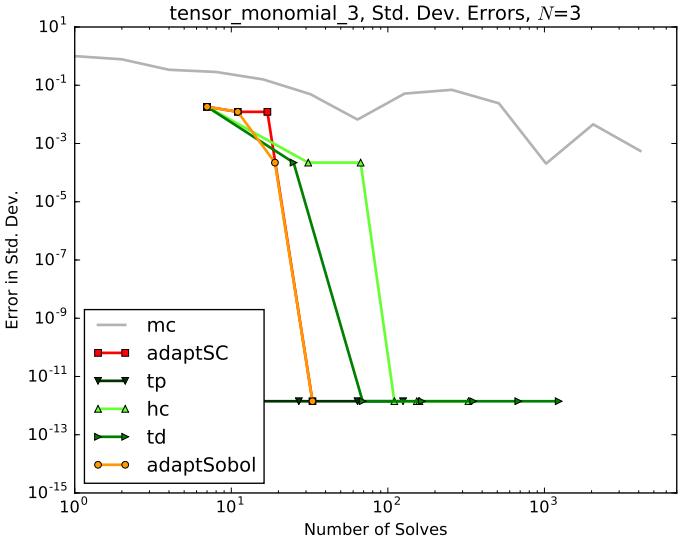


FIGURE 4.4: Tensor Monomial, $N = 3$, Std. Dev. Convergence

4.2.2 5 Inputs

While the convergence on the mean is still direct for the five-dimensional input problem, we begin to see degradation in the convergence of collocation-based methods. Total Degree outperforms adaptive methods, as the search algorithms struggle to find the optimal tensors of low-order polynomials required. Hyperbolic Cross is outperformed by Total Degree, as expected for a problem with this level of regularity.

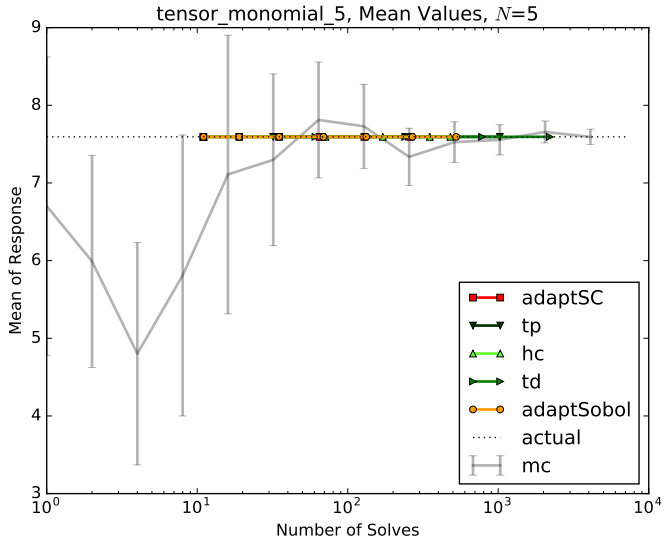


FIGURE 4.5: Tensor Monomial, $N = 5$, Mean Values

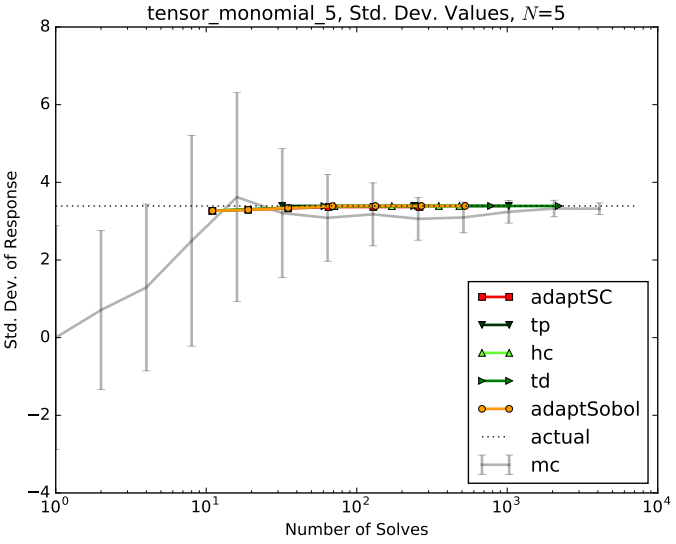


FIGURE 4.6: Tensor Monomial, $N = 5$, Std. Dev. Values

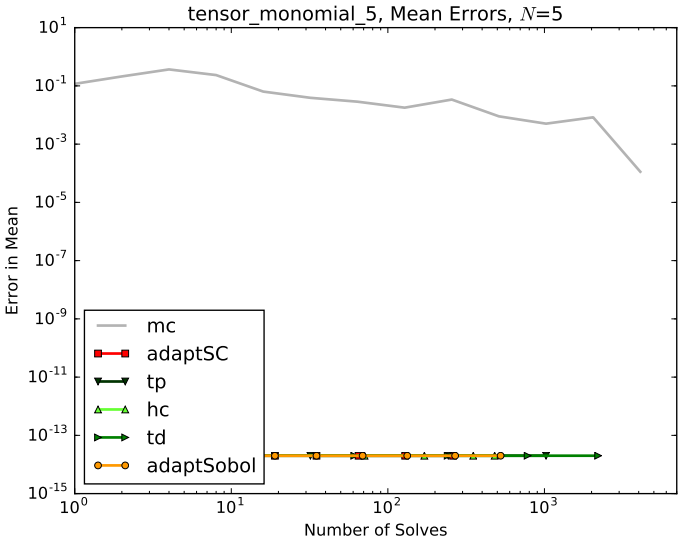


FIGURE 4.7: Tensor Monomial, $N = 5$, Mean Convergence

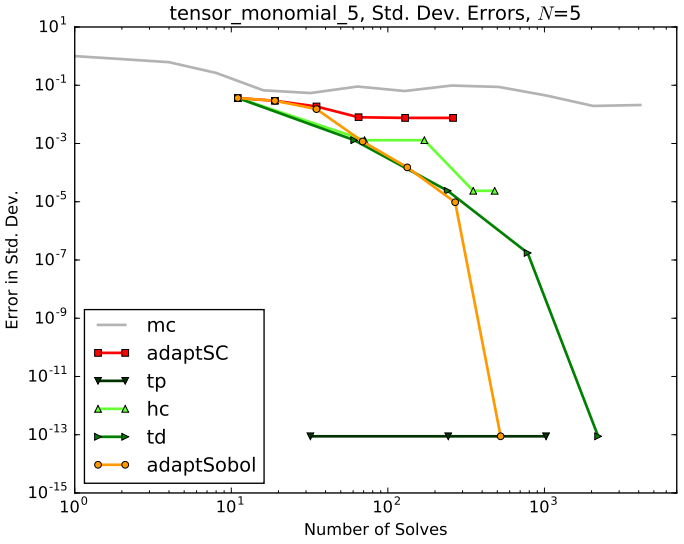


FIGURE 4.8: Tensor Monomial, $N = 5$, Std. Dev. Convergence

4.2.3 10 Inputs

As we increase to ten inputs, we see significant degradation of all the collocation methods in converging on the standard deviation. While it appears there is exponential convergence, the curvature is quite large, and little better than linear convergence is observed for up to 1000 computational solves. One reason the adaptive methods do not perform more admirably for this case is the equal-weight importance of all the input terms as well as the polynomial terms; the high-dimensional space takes considerable numbers of runs

to explore thoroughly, and this model contains some of the most difficult polynomials to find adaptively: those including all of the inputs.

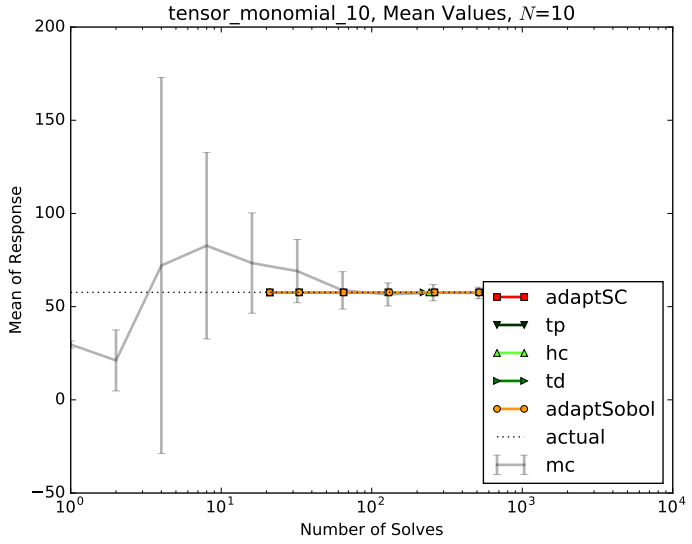


FIGURE 4.9: Tensor Monomial, $N = 10$, Mean Values

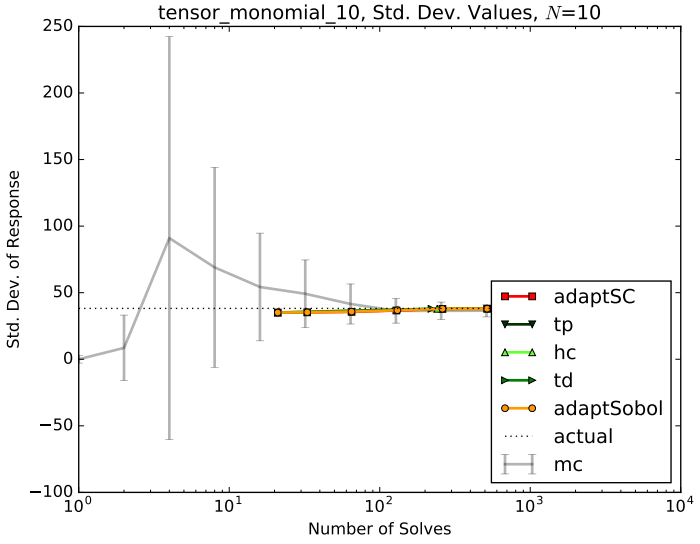


FIGURE 4.10: Tensor Monomial, $N = 10$, Std. Dev. Values

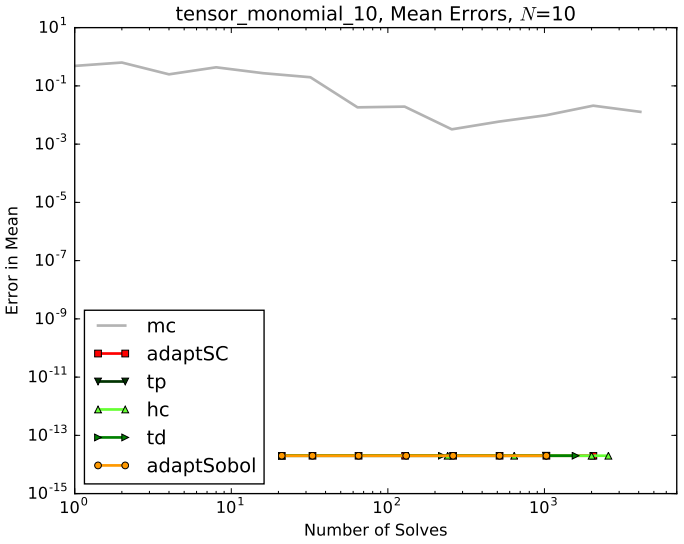


FIGURE 4.11: Tensor Monomial, $N = 10$, Mean Convergence

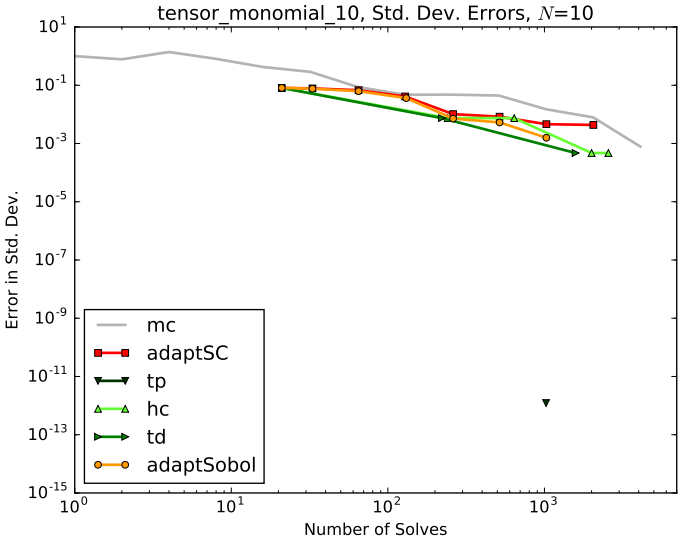


FIGURE 4.12: Tensor Monomial, $N = 10$, Std. Dev. Convergence

4.3 Sudret Polynomial

This model is described in section 2.3.

4.3.1 3 Inputs

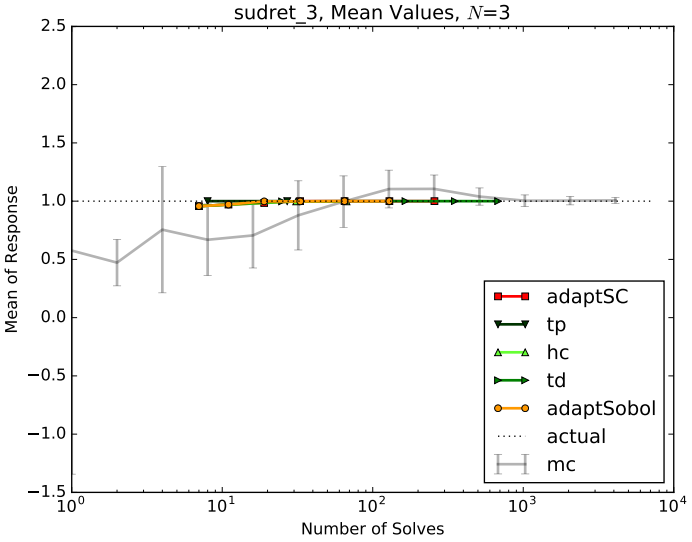


FIGURE 4.13: Sudret Polynomial, $N = 3$, Mean Values

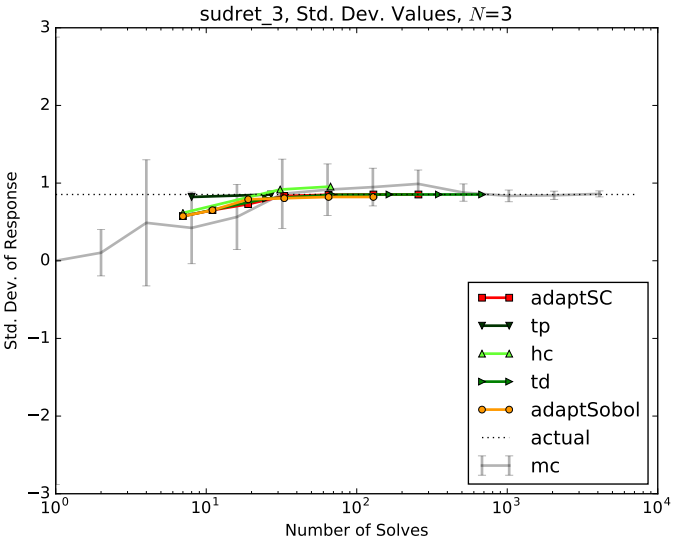


FIGURE 4.14: Sudret Polynomial, $N = 3$, Std. Dev. Values

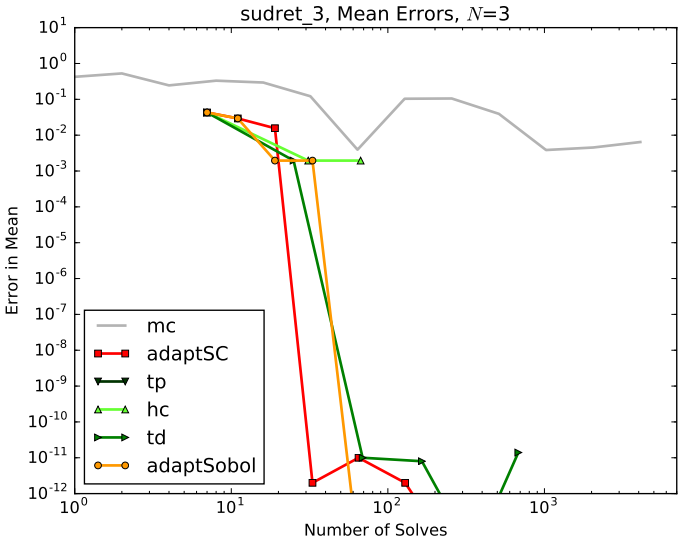


FIGURE 4.15: Sudret Polynomial, $N = 3$, Mean Convergence

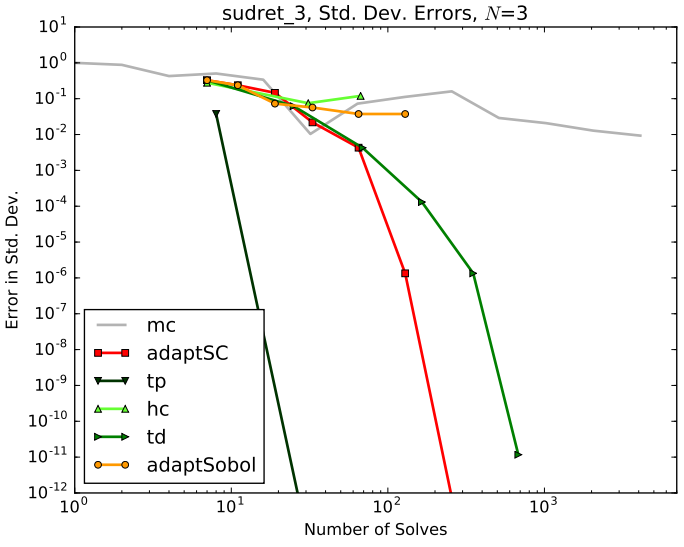


FIGURE 4.16: Sudret Polynomial, $N = 3$, Std. Dev. Convergence

4.3.2 5 Inputs

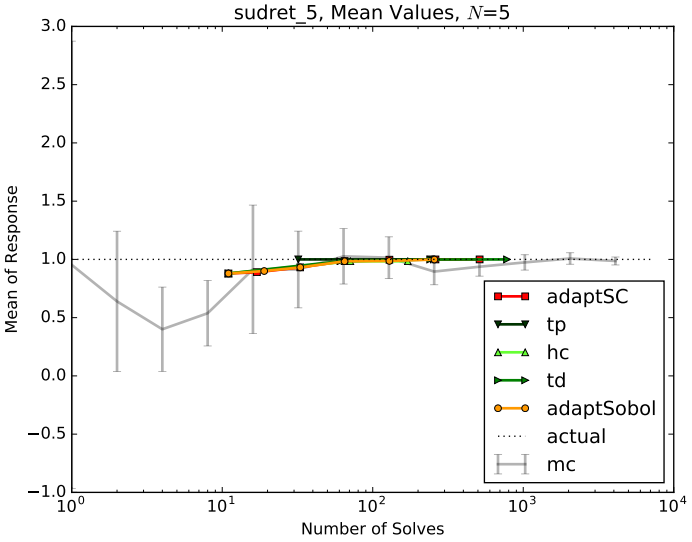


FIGURE 4.17: Sudret Polynomial, $N = 5$, Mean Values

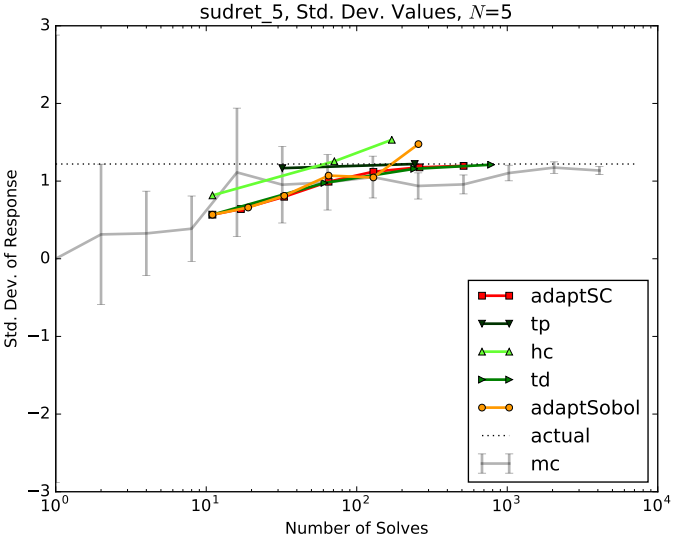


FIGURE 4.18: Sudret Polynomial, $N = 5$, Std. Dev. Values

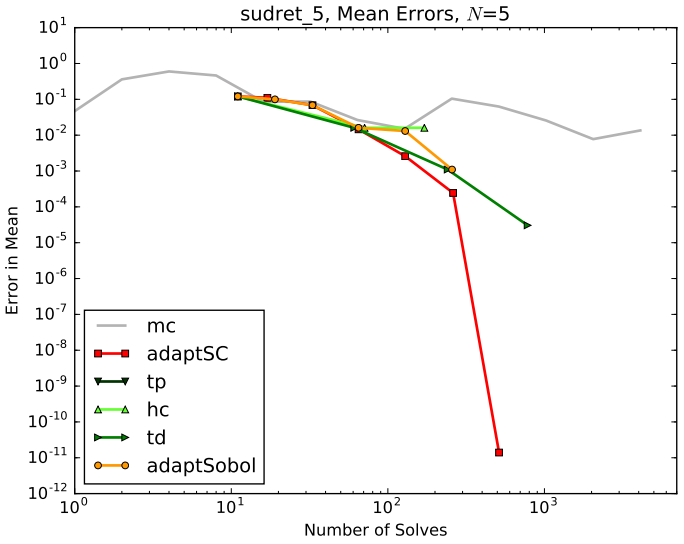


FIGURE 4.19: Sudret Polynomial, $N = 5$, Mean Convergence

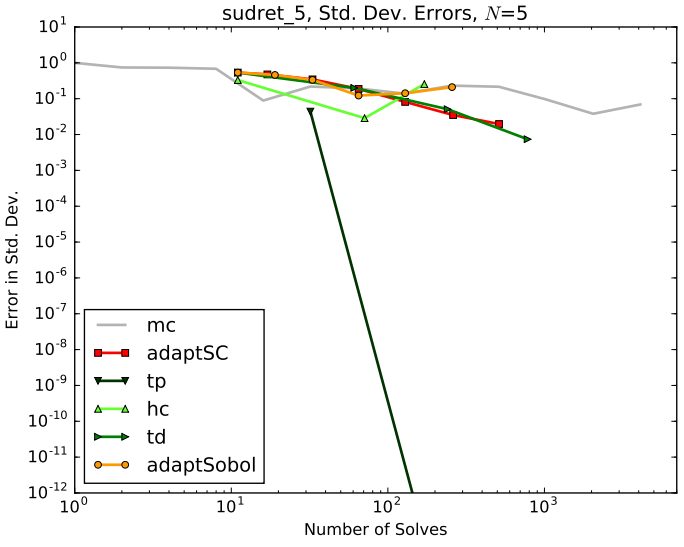


FIGURE 4.20: Sudret Polynomial, $N = 5$, Std. Dev. Convergence

4.4 Attenuation

This model is described in section 2.4.

4.4.1 2 Inputs

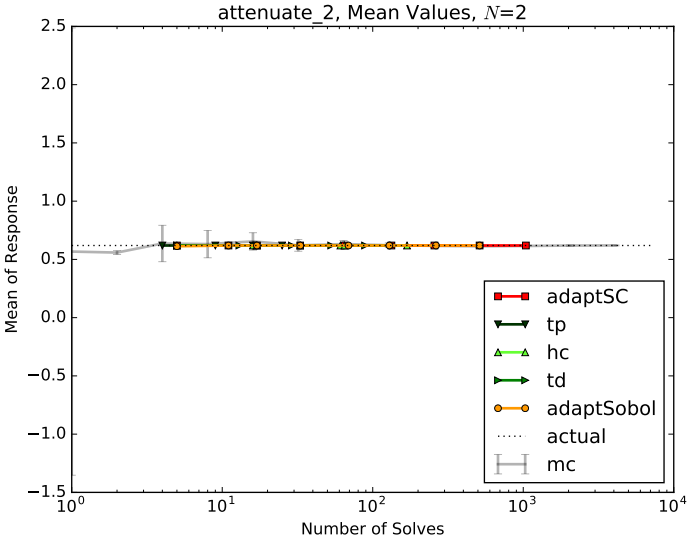


FIGURE 4.21: Attenuation, $N = 2$, Mean Values

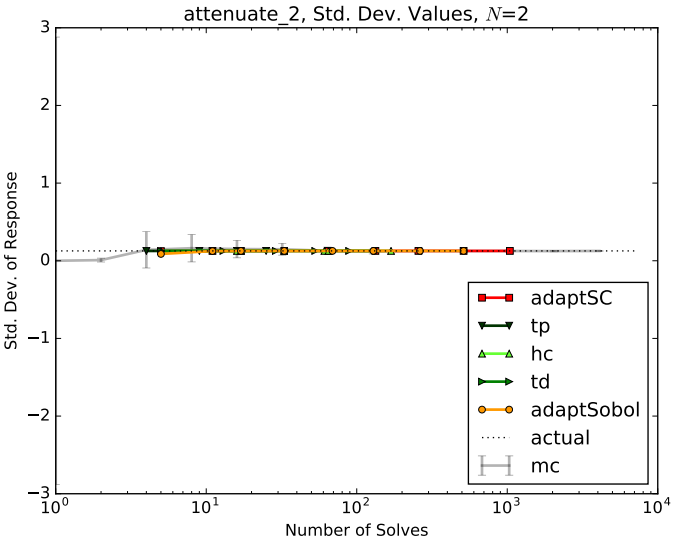


FIGURE 4.22: Attenuation, $N = 2$, Std. Dev. Values

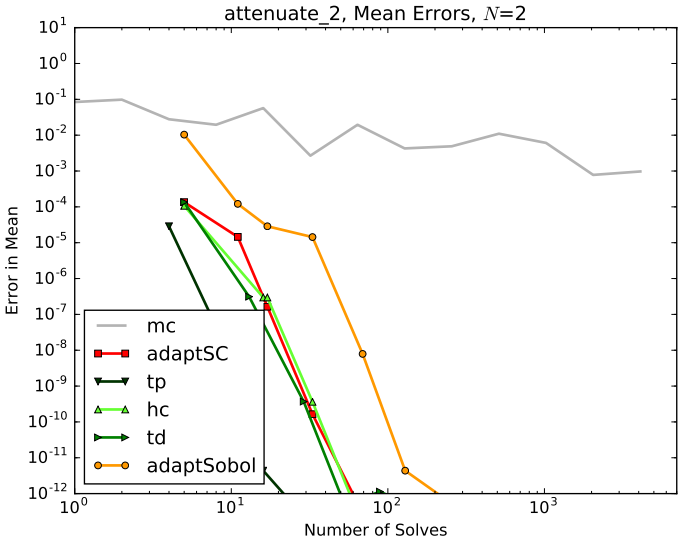


FIGURE 4.23: Attenuation, $N = 2$, Mean Convergence

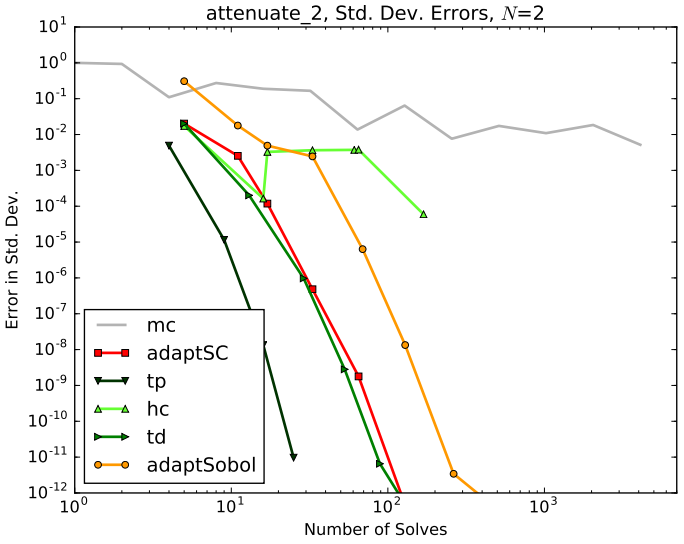


FIGURE 4.24: Attenuation, $N = 2$, Std. Dev. Convergence

4.4.2 4 Inputs

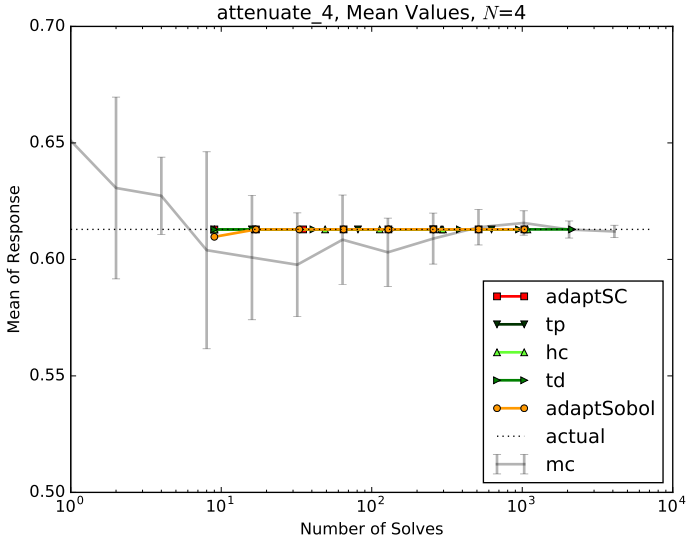


FIGURE 4.25: Attenuation, $N = 4$, Mean Values

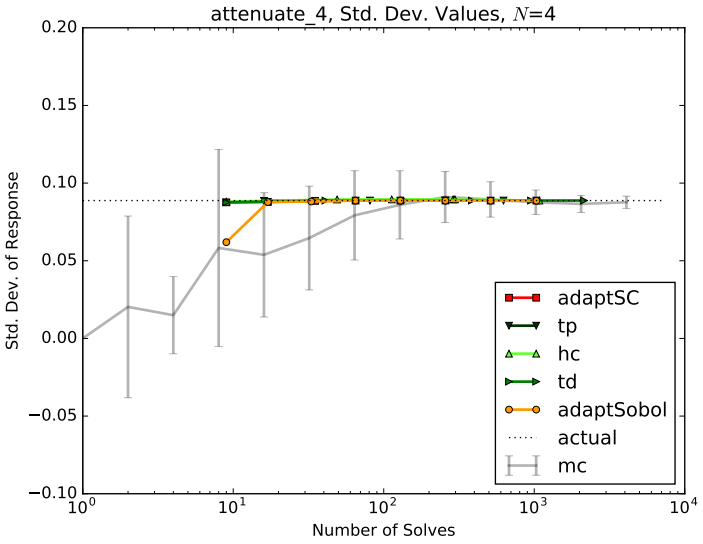


FIGURE 4.26: Attenuation, $N = 4$, Std. Dev. Values

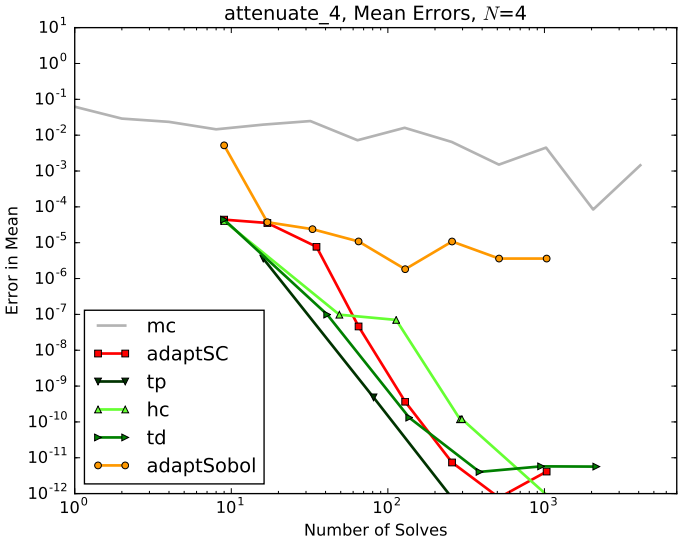


FIGURE 4.27: Attenuation, $N = 4$, Mean Convergence

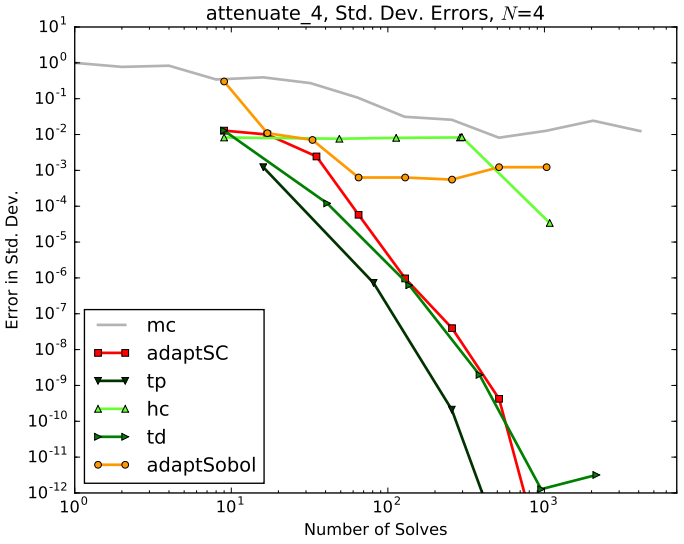


FIGURE 4.28: Attenuation, $N = 4$, Std. Dev. Convergence

4.4.3 6 Inputs

4.5 Gauss Peak

This model is described in section 2.5.

4.6 Ishigami

This model is described in section 2.6.

4.7 Sobol G-Function

This model is described in section 2.7.

4.8 Anisotropic Polynomial

This model is described in section 2.8.

Chapter 5

Engineering Demonstration

5.1 Todo

todo.

Notes:

Run times

For Picard 6 and SN (3 azimuthal, 3 polar Gauss Chebyshev):

MPI 24: 11m 29.452s = 689.452 sec = 16546.848 single equivalent (0.424 efficient) MPI 12: 19m 41.703s = 1181.703 sec = 14180.436 single equivalent (0.495 efficient) MPI 6: 27m 50.373s = 1670.373 sec = 10022.238 single equivalent (0.701 efficient) MPI 1: 117m 0.756s = 7020.756 sec = 7020.756 single equivalent (1.000 efficient)

Chapter 6

Conclusions

6.1 Todo

todo.

Chapter 7

Future Work

7.1 Introduction

The results in this work lead to several interesting areas of improvement. We discuss some of these briefly here.

7.2 Impact Inertia in Adaptive Samplers

One weakness demonstrated in the adaptive sampling techniques is the phenomenon of purely-even or purely-odd polynomial representations. This is seen clearly in the Ishigami (2.6) and Gauss Peak (2.5) models. Consider the Taylor development of a sine function,

$$\sin x = x - \frac{x^3}{6} + \frac{x^5}{120} + \mathcal{O}(x^7), \tag{7.1}$$

and for a square exponential,

$$e^{-x^2} = 1 - x^2 + \frac{x^4}{2} - \frac{x^6}{6} + \frac{x^8}{24} + \mathcal{O}(x^10). \tag{7.2}$$

Unique to both of these functions is "skipping" certain polynomial orders (evens for sine, odds for square exponential).

In a single-dimension example, the current impact estimation expression is

$$\tilde{\eta_k} = \eta_{k-1}. \tag{7.3}$$

Because adaptive sampling currently relies on the previous-order polynomial to estimate the importance of the current polynomial, it can be misled into thinking there is no additional information to gather if certain polynomials are not present in the expansion.

For example, if the adaptive sampler finds the impacts of a one-dimensional problem to be 0.4 for x, it will try x^2. If the model is an odd function, it will find the impact for x^2 is actually zero. As a result, it will be very unlikely to try x^3, despite the fact that x^3 has significant real impact.

One resolution to this method is to apply some sort of *impact inertia* to the estimation of impact values; that is, in addition to considering the previous polynomial impact when estimating current polynomial impact, several previous polynomials might be considered. This kind of inertia is likely problem-dependent in its effectiveness, and would be best controlled through an optional user input. Some research would be required to determine what default level of inertia is recommended. The new impact estimation expression would be something like the following:

$$\tilde{\eta_k} = \frac{1}{\alpha} \sum_{n=1}^{k} \frac{1}{g(n)} \eta_{k-n}, \tag{7.4}$$

where α is a balancing parameter and $g(n)$ is a penalty function that grows as $k - n$ increases.

7.3 Cross-Communication in Adaptive HDMR

Todo, basically if I want the next step in (x,y), that next step shouldn't be a polynomial containing 0 in either x or y.

Bibliography

[1] Michael Heroux et. al. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[2] E. D. Fichtl and A. K. Prinja. The Stochastic Collocation Method for Radiation Transport in Random Media. *J. Quantitative Spectroscopy & Radiative Transfer*, 12, 2011.

[3] M.E. Rising, A.K. Prinja, and P. Talou. Prompt Fission Neutron Spectrum Uncertainty Propagation Using Polynomial Chaos Expansion. *Nucl. Sci. Eng.*, 175, 2013.

[4] Talbot, Prinja, and Rabiti. Adaptive sparse-grid stochastic collocation uncertainty quantification convergence for multigroup diffusion. *2016 ANS summer conference transactions*, 114:738–740, June 2016.

[5] Talbot and Prinja. Sparse-grid stochastic collocation uncertainty quantification convergence for multigroup diffusion. *2014 ANS winter conference transactions*, 111:747–750, November 2014.

[6] Cooling, Ayres, Prinja, and Eaton. Uncertainty and global sensitivity analysis of neutron survival and extinction probabilities using polynomial chaos. *Annals of Nuclear Energy*, 88:158–173, November 2016.

[7] Ayres and Eaton. Uncertainty quantification in nuclear criticality modelling using a high dimensional model representation. *Annals of Nuclear Energy*, 80:379–402, May 2015.

[8] Rabiti, Cogliati, Pastore, Gardner, and Alfonsi. Fuel reliability analysis using bison and raven. In *PSA 2015 Probabilistic Safety Assessment and Analysis*, Sun Valley, Idaho, April 2015.

[9] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[10] Nicholas Metropolis. The beginning of the monte carlo method. *Los Alamos Science*, 15(584):125–130, 1987.

[11] Herman Kahn and Andy W Marshall. Methods of reducing sample size in monte carlo computations. *Journal of the Operations Research Society of America*, 1(5): 263–278, 1953.

[12] Thomas E Booth. Mcnp variance reduction examples. *Los Alamos National Laboratory, Diagnostic Applications Group X-5. Mail Stop F*, 663, 2004.

[13] McKay, Beckman, and Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[14] Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60:897–936, 1938.

[15] Babuska, Tempone, and Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825, 2004.

[16] Gleicher, Williamson, Ortensi, Wang, Spencer, Novascone, Hales, and Martineau. The coupling of the neutron transport application rattlesnake to the nuclear fuels performance application bison under the moose framework. Technical report, Idaho National Laboratory (INL), 2015.

[17] Gaston, Newman, Hansen, and Lebrun-Grandié. Moose: a parallel computational framework for coupled systems of nonlinear equations. *Nuclear Engineering and Design*, 239(10):1768 – 1778, 2009.

[18] Newman, Hansen, and Gaston. Three dimensional coupled simulation of thermomechanics, heat, and oxygen diffusion in nuclear fuel rods. *Journal of Nuclear Materials*, 392(1):6 – 15, 2009.

[19] Rabiti, Alfonsi, Mandelli, Cogliati, and Kinoshita. Raven, a new software for dynamic risk analysis. In *PSAM 12 Probabilistic Safety Assessment and Management*, Honolulu, Hawaii, June 2014.

[20] Wang. Nonlinear diffusion acceleration for the multigroup transport equation discretized with sn and continuous fem with rattlesnake. In *Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering (M&C 2013)*, Sun Valley, Idaho, May 2013.

[21] Xiu and Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, 2002.

[22] Askey and Wilson. Some basic hypergeometric orthogonal polynomials that generalize jacobi polynomials. *Memoirs of the American Mathematical Society*, 54:1–55, 1985.

[23] Novak and Ritter. The curse of dimension and a universal method for numerical integration. In Günther Nürnberger, JochenW. Schmidt, and Guido Walz, editors, *Multivariate approximation and splines*, volume 125 of *ISNM International Series of Numerical Mathematics*, pages 177–187. Birkhäuser Basel, 1997.

[24] Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Dokl. Akad. Nauk SSSR*, volume 4, page 123, 1963.

[25] Trefethen. Is guass quadrature better than clenshaw-curtis? *SIAM Review*, 50(1): 67–87, 2008.

[26] Gerstner and Griebel. Dimension-adaptive tensor-product quadrature. *Computing*, 71, 2003.

[27] Boulore, Struzik, and Gaudier. Uncertainty and sensitivity analysis of the nuclear fuel thermal behavior. *Nuclear Engineering and Design*, 253:200–210, 2012.

[28] Argonne National Laboratory. Argonne code center: benchmark problem book. *ANL-7416 M&C Division of ANS*, 1968.

[29] Babuska, Nobile, and Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45, 2007.

[30] Li, Rosenthal, and Rabitz. High dimensional model representations. *J. Phys. Chem. A*, 105, 2001.

[31] Hu, Smith, Willert, and Kelley. High dimensional model representations for the neutron transport equation. *NS&E*, 177, 2014.

[32] Nobile, Tempone, and Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46, 2008.

[33] Barthelmann, Novak, and Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12, 2000.

[34] Bungartz and Griebel. Sparse grids. *Acta Numerica*, 13, 2004.

[35] Le Maître and Knio. *Spectral methods for uncertainty quantification with applications to computational fluid dynamics*. Springer, 1st edition, 2010.

[36] Blyth, Porter, Avramova, Ivanov, Royer, Sartori, Cabellos, Feroukhi, and Ivanov. Benchmark for uncertainty analysis in modelling (uam) for design, operation, and safety analysis of lwrs. volume ii: specification and support data for the core cases (phase ii). *Nuclear Energy Agency/Nuclear Science Committee of the Organization for Economic Cooperation and Development*, Version 2, 2014.

[37] Satosi Watanabe. Karhunen-loeve expansion and factor analysis, theoretical remarks and applications. In *Proc. 4th Prague Conf. Inform. Theory*, 1965.

[38] Talbot, Wang, Rabiti, and Prinja. Multistep input reduction for high dimensional uncertainty quantification in raven code. *Proceedings of PHYSOR*, 2016.

[39] SCALE Manual Scale ORNL. Scale: A comprehensive modeling and simulation suite for nuclear safety analysis and design. *ORNL/TM-2005/39, Version*, 6.

[40] Sudret. Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety*, 93(7):964–979, 2008.

[41] Genz. A package for testing multiple integration subroutines. In *Numerical Integration*, pages 337–340. Springer, 1987.

[42] T Ishigami and Toshimitsu Homma. An importance quantification technique in uncertainty analysis for computer models. In *Uncertainty Modeling and Analysis, 1990. Proceedings., First International Symposium on*, pages 398–403. IEEE, 1990.

[43] Amandine Marrel, Bertrand Iooss, Beatrice Laurent, and Olivier Roustant. Calculations of sobol indices for the gaussian process metamodel. *Reliability Engineering & System Safety*, 94(3):742–751, 2009.