University of New Mexico

Doctoral Thesis

Advanced Stochastic Collocation Methods 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

October 2016

Declaration of Authorship

TODO find the UNM version of this. I, Paul W. Talbot, declare that this thesis titled, 'Advanced Stochastic Collocation Methods for Polynomial Chaos in RAVEN' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

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 calculations. For other models, it is computationally costly to obtain a good understanding of the output space.

To combat the expense 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 thesis 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. While an infinite number of polynomials can exactly represent any mathematical model, of necessity we limit (or truncate) the number of polynomials for practical application. We also consider applying anisotropy (unequal treatment of different dimensions) to the polynomial space, and analyze methods whereby the optimal level of anisotropy can be approximated. We review and develop potential adaptive polynomial construction strategies. Finally, we add High-Dimensional Model Reduction (HDMR) expansions, using SCgPC representations for the subspace terms, and consider adaptive methods to construct them. We analyze these methods on a series of models of increasing complexity. We use a series of analytic methods of various levels of complexity, then demonstrate performance on two engineering-scale problems: a single-physics nuclear reactor neutronics problem, and a multiphysics fuel cell problem coupling fuels performance and neutronics. Lastly, we demonstrate sensitivity analysis for a time-dependent fuels performance problem. For this analysis, we demonstrate the application of the algorithms in `RAVEN`, a production-level uncertainty quantification framework.

Acknowledgements

TODO finish acknowledge Adviser Anil Prinja, along with other UNM committee members etc

The `RAVEN` team: Cristian Rabiti, Andrea Alfonsi, Joshua Cogliati, Diego Mandelli, CongJian Wang, and Dan Maljovec. These remarkable gentlemen have provided no end of assistance as I learned, used, and contributed to their project. It has been an honor and a phantasmagoric pleasure working on the team.

University of New Mexico colleagues Aaron Olsen, David Dixon, Nick Myers, and Matthew Gonzalez, whose collaboration dried out many whiteboard markers but led to many discoveries and much better understanding.

Idaho National Laboratory researchers Sebastian Schunert, for his efforts in helping me use both `MAMMOTH` and `RATTLESNAKE`, and Kyle Gamble, for his contributions in using `BISON`.

Contents

List of Figures

List of Tables

Abbreviations

ANOVA	ANalysis Of VAriance
CDF	Cumulative Distribution Function
gPC	generalize Polynomial Chaos
HDMR	High-Dimensional Model Representation
KL	Karhunen-Loevre Expansion
LHS	Latin Hypercube Sampling
MC	Monte Carlo (sampling strategy)
PDF	Probability Distribution Function
SCgPC	Stochastic Collocation for generalized Polynomial Chaos expansions
UQ	Uncertainty Quantification

Symbols

Symbol	Name	Space
N	Dimensionality of input space	$\mathbb{N}^1$
Y	Multidimensional uncertain input vector	$\mathbb{R}^N$
y_n	Single uncertain input	$\mathbb{R}^1$
Ω	Multidimensional uncertainty input space	$\mathcal{R}^N$
Ω_{y_n}	Single variable uncertainty input space	$\mathcal{R}^1$
$u(Y)$	Response functional	$\mathbb{R}^1$
k	Multidimensional index for polynomials	$\mathbb{N}^N$
k_n	Single index for polynomials	$\mathbb{N}^1$
L	Limiting order for SCgPC expansion	$\mathbb{N}^1$
$\lvert\Lambda\rvert$	Number of polynomials in SCgPC expansion	$\mathbb{N}^1$
Λ	Collection of k corresponding to polynomial orders	$\mathbb{N}^{\lvert\Lambda\rvert\times N}$
$\mathbb{E}[u(Y)]$	Expectation value of $u(Y)$	$\mathbb{R}^1$
μ	Mean of a distribution	$\mathbb{R}^1$
σ	Standard Deviation of a distribution	$\mathbb{R}^1$
γ_1	Skewness of a distribution	$\mathbb{R}^1$
γ_2	Excess Kurtosis of a distribution	$\mathbb{R}^1$
$\mathcal{S}_\setminus$	Sobol sensitivity coefficient for input n	$\mathbb{R}^1$
η_k	Impact of polynomial k on response	$\mathbb{R}^1$
ξ	Impact of HDMR subset on response	$\mathbb{R}^1$
$\phi_{k_n}(y_n)$	Orthonormal polynomial of order k_n	
$\Phi_k(Y)$	Multidimensional orthonormal polynomial of orders k	
$S[u(Y)]$	Smolyak sparse grid numerical integral of $u(Y)$	
$G(Y)$	SCgPC expansion of $u(Y)$	
$H(Y)$	ANOVA HDMR expansion of $u(Y)$	

$T(Y)$	Cut-HDMR expansion of $u(Y)$

TODO Dedication

Chapter 1

Introduction

In simulation modeling, we seek to capture the behavior of a physical system by describing it in a *model*, a series of mathematical equations. These often take the form of partial differential equations. These models may be time- and spacially-dependent, and capture physics of interest for understanding the system. A *solver* is then written that can solve the series of equations and determine *responses*, or quantities of interest, often through numerical evaluations on computation devices. 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 response might be the strength of the beam exiting the material. A single evaluation of the solver usually results in a single value, or realization, of the response. Figure 1.1 shows these relationships, and provides an example for nuclear reactor criticality calculations.

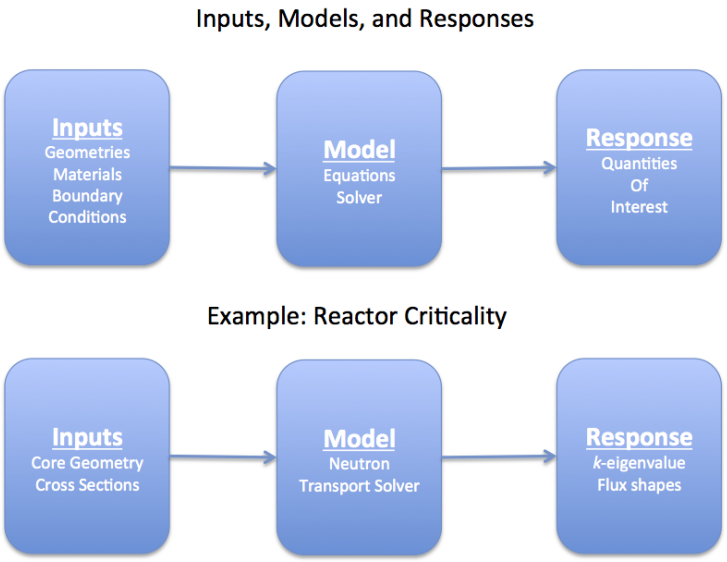

FIGURE 1.1: Example Models

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. This multidimensional probabilistic space is weighted by the probability of realizations occurring in hypervolumes within the space. While some points in that space may be more likely than others, the possible range of values for the response is only understood when the uncertain input space is considered as a whole and propagated through the model. 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 independent variables (see 2.1.3). Throughout this work we will assume the input parameters are independent.

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 responses are collected. Then, another point in the input space is chosen at random. This process continues until the properties of the 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 response 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, such as frameworks like `RAVEN` [2].

Monte Carlo (MC) and similar sampling strategies are relatively slow to converge the response surface. The response surface is the space of all possible outcomes for the model given the uncertainty in the inputs, weighted by the probability of that outcome occurring. This surface is difficult to converge accurately in MC. For example, in order to reduce the standard error of the mean of the response by a factor of two using MC, it is necessary to evaluate the model four times more often. If a solver is sufficiently computationally inexpensive, obtaining additional evaluations is not a large concern; however, for lengthy and expensive solvers such as those found commonly in nuclear engineering applications, it may not be practical to obtain sufficient realizations to obtain a clear response surface. In this work MC is used as a benchmark methodology; if other methods converge on moments of the responses more quickly and consistently than MC, we consider them "better" for our purposes.

The first advanced uncertainty quantification method we consider is Stochastic Collocation for generalized Polynomial Chaos (SCgPC) [25, 36–38], wherein deterministic collocation points are used to develop a polynomial-interpolated surrogate model of the response as a function of the inputs. This method algorithmically expands the solver as the sum of orthogonal multidimensional polynomials with scalar coefficients. The scalar coefficients are obtained by numerical integration using multidimensional collocation (quadrature) points. The chief distinction between SCgPC and MC methods is that SCgPC is deterministic, in that the realizations required from the solver are predetermined instead of randomly sampled. Like Monte Carlo, SCgPC is unintrusive and performs well without any need to access the internal operations of the solver. This behavior is desirable for construction of black-box approach algorithms for uncertainty quantification. Other intrusive methods such as Stochastic Galerkin exist [20], 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 standard SCgPC. First, we explore non-tensor-product methods for determining the set of polynomial bases to use in the expansion. Because a tensor product grows exponentially with increasing dimensionality of the input space, we combat this curse of dimensionality using alternative polynomial set construction methods [27]. These polynomial bases will then be used to construct Smolyak-like sparse grids [28] to provide collocation points that are used to calculate the

coefficients in the polynomial expansion. Further, 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-Dimensional Model Representation (HDMR), sometimes also referred to as high-density model reduction, which is based on Sobol decomposition [34]. This method is useful both for developing sensitivities of the quantity of interest with respect to subsets of the input space, as well as constructing a reduced-order representation of the 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.

This work is premised on the idea that solvers are often computationally expensive, requiring many hours per evaluation, and that computational resource availability requires an analyst perform as few evaluations as possible. As such, we consider several methodologies for quantifying the uncertainty in expensive solver calculations. In order to demonstrate the range of operation for these methods, we apply them first on several analytic problems, such as polynomial evaluations. 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 advanced UQ methods (SCgPC and HDMR) in comparison to MC.

However, the problems facing nuclear engineering analysts today are rarely analytic and have simple forms. As a result, we also demonstrate three applications of SCgPC and HDMR to engineering-scale applications. The first is a single-physics neutronics benchmark problem that models a small reactor core. While the responses of this model have no simple analytic form, the underlying physics are well-understood and provide for good analysis. The second engineering-scale application is a multiphysics problem modeling nuclear fuel through burnup depletion. This model couples neutronics and fuels performance nonlinearly, where neutronics provides flux shapes that determine power shapes for fuels performance, and fuels performance provides temperature feedbacks to the nuclear cross sections. Finally, we demonstrate how HDMR and SCgPC can be used to analyze a time-dependent fuels performance problem in which a fuel rod undergoes several changes in power level over a long period of irradiation. Instead of converging moments of responses, this last analysis concerns using changing sensitivity parameters to detect changes in dominant physics during a transient problem.

We implement all the advanced UQ methods in Idaho National Laboratory's `RAVEN` [2] 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 methods developed, we apply `RAVEN` to complex non-linear solvers. To simulate neutronics and fuels performance, we use `RATTLESNAKE` [24] and `BISON` [21, 23] production codes respectively. Both of these codes are developed in and based on the `MOOSE` [22] environment.

The remainder of this work will proceed as follows: FIXME TODO

- Chapter 2: We begin by defining concepts and ideas used in uncertainty quantification. We discuss uncertainty spaces, correlated inputs, statistical moments, the purposes of uncertainty quantification, and some common existing uncertainty quantification techniques, including Monte Carlo, Grid, and Latin Hypercube sampling strategies.

- Chapter 3: We consider SCgPC as an uncertainty quantification method in comparison to Monte Carlo. We discuss implementation of several polynomial expansion types as well as both isotropic and anisotropic approaches. We additionally consider an adaptive scheme for the polynomial expansion, and offer some improvements on existing efforts in adaptivity.

- Chapter 4: We present the performance of SCgPC expansion methods as applied to a variety of analytic models. We analyze a set of increasingly-complex models and contrast collocation methods on models of varying dimensionality. We draw some conclusions based on performance to determine when SCgPC is a viable choice over Monte Carlo sampling.

- Chapter 5: Expanding on SCgPC, we further consider HDMR, achieved through Sobol decomposition. We apply generalized polynomial chaos expansions to the subspace terms in the HDMR expansion, and demonstrate synergies with the two expansions. Further, we present a combined adaptive scheme for the two expansion methods and offer improvements on existing adaptive methods.

- Chapter 6: We add the performance of HDMR to the previous analyses of analytic models, and consider the merits and shortcomings of several truncation orders of HDMR eduction. We also consider the adaptive HDMR method when compared with the SCgPC methods.

- Chapter 7: We apply the uncertainty quantification techniques described in this work to a single-physics neutronics benchmark problem. We consider the efficiencies and limitations of both SCgPC and HDMR as uncertainty quantification tools for this model, and discuss some findings due to this non-analytic model.

- Chapter 8: We further apply the advanced uncertainty quantification techniques to a multiphysics engineering-scale problem coupling fuels performance to neutronics. We consider the efficiency of select methods discussed, and limitations discovered from these coupled codes.

- Chapter 9: We consider application of low-order HDMR to a time-dependent fuels performance benchmark. We present the development of Sobol sensitivity coefficients between responses of interest and uncertain input parameters over a time-dependent problem. Further, we demonstrate how these sensitivities can yield additional comprehension of the underlying physical models. We discuss briefly the limitations discovered as a result of performing this analysis.

- Chapter 10: We summarize the efforts of this paper and offer conclusions, including the types of models for which the uncertainty quantification techniques are efficient, when they can be expected to be inefficient, and when they will not work altogether. We also suggest some future work efforts that are logical progressive steps from the work performed here.

Chapter 2

Traditional Uncertainty Quantification Methods

2.1 Introduction

In this chapter we describe traditional uncertainty quantification concepts as well as several common existing 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 finally describe several common uncertainty quantification tools.

Many simulation models are algorithms constructed to solve partial differential equations, often in two or three spatial dimensions and possibly time. The inputs to these models include boundary conditions, material properties, tuning parameters, and so forth. The outputs are responses, either data fields or scalar values. The responses are used to inform decision-making processes. For example, a neutronics simulation in nuclear engineering takes materials, geometries, and boundary conditions as inputs, and yields neutron flux and the neutron multiplication factor k as responses. Similarly, a fuels performance code takes materials, geometries, and power shapes as inputs and yields stresses, strains, and temperature profiles as outputs. Figure 2.1 is an example of this workflow [1].

In general, we define $u(Y)$ to be a response as a function of the input space $Y = (y_1, \ldots, y_n, \ldots, y_N)$ where y_n is a single uncertain input parameter to the model, n is an index spanning the number of inputs, and N is the total number of inputs. Uncertain input parameters can include any of the inputs to the simulation. We assume each response to be a scalar, integrated quantity. In the event the output is a vector or

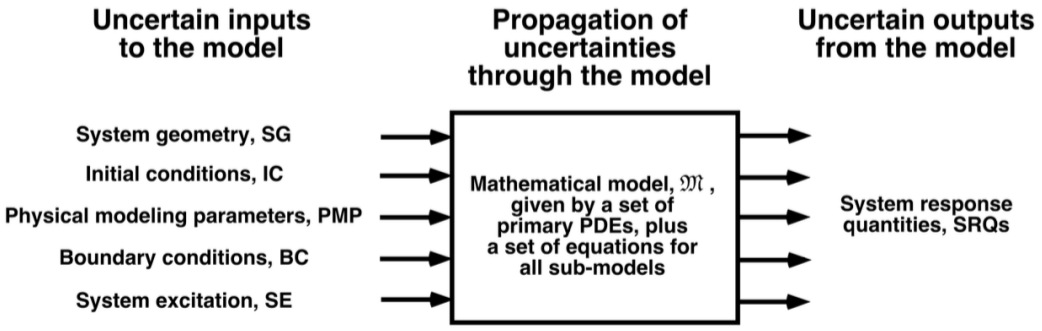

FIGURE 2.1: Uncertainty Quantification [1]

field quantity, each element can be considered as a distinct scalar response. *Models* are mathematical equations used to obtain the response $u(Y)$, and *solvers* or *simulations* are numerical algorithms used to solve models.

Using our examples above, for neutronics calculations Y might include nuclear cross sections, geometry parameters, and sources, while $u(Y)$ could be k-effective or the neutron flux at a particular location of interest. For fuels performance calculations, Y might entail thermal conductivity of various parameters, geometric construction parameters, moderator inlet temperatures, and so forth. $u(Y)$ could be peak clad temperature, maximum fuel centerline temperature, clad elongation, percent fission gas released, and so on.

2.1.1 Uncertain Inputs

Essential to using simulation models is understanding the possibility that significant 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. For example, quantum behaviors or Brownian motion often provide non-deterministic sources of aleatoric uncertainty. Further, the simulation itself might be solved through non-deterministic methods such as Monte Carlo sampling, in which case the random seed acts as an uncertain input. Examples of epistemic uncertainties include initial or boundary conditions that can only be controlled to some finite level, such as manufacturing tolerances, temperature and pressure, and so forth. Each of these aleatoric and epistemic uncertainties has some distribution defining the likelihood of an input to have a particular value. Sometimes these distributions are known; often, they can only be approximated. These distributions might be assumed or constructed from experiment; for our work, we will assume distributions are given, and that the given distributions are accurate. The input likelihood distribution is the probability distribution function (PDF) $\rho_n(y_n)$. An integral over any portion of the input space of the PDF

provides the probability that the input's value is within that portion. We require

$$\int_a^b \rho_n(y_n) d\ y_n = 1, \tag{2.1}$$

where a and b are the minimum and maximum values y_n can take (possibly infinite). In other words, the probability of finding the input between a and b is 100%; similarly, we can say the value of the input lies between a and b almost surely.

2.1.2 Multidimensional Input Spaces

When there are more than one uncertain input, the combination of distributions for these inputs span an uncertainty space Ω. Ω is a part of the probability space (Ω, σ, ρ), where Ω is the set of all possible outcomes, σ is the set of events ω, and ρ is the probability function for the space. The dimensionality of Ω is N, the number of uncertain input variables. The probability of any event in the input space occurring is given by an integral of the joint-probability distribution $\rho(Y)$, still enforcing

$$\int_{a_1}^{b_1} \cdots \int_{a_N}^{b_N} \rho(Y) dy_1 \cdots dy_N = 1. \tag{2.2}$$

For clarity, we define multidimensional integral operator

$$\int_\Omega (\cdot) dY \equiv \int_{a_1}^{b_1} \cdots \int_{a_N}^{b_N} (\cdot) dy_1 \cdots dy_N, \tag{2.3}$$

so that Eq. 2.2 can be written

$$\int_\Omega \rho(Y) dY = 1. \tag{2.4}$$

The function $u(Y)$ maps realizations (ω) from the input space Ω to a real-valued response. That is, for each input variable y_n, a realization is taken by selecting a single value from the distribution of y_n, which gives a single input value $y_n(\omega)$. Taking a single realization of each of the distributed input parameters yields a full input realization $Y(\omega) = (y_1(\omega), \cdots, y_N(\omega))$, which can be used as inputs for $u(Y)$ to obtain a realization of the response $u(Y(\omega))$. To simplify notation, in general the dependency of a realization on ω will be omitted and referred to as *sampling* or *taking a realization.*

2.1.3 Correlation and the Karhunen-Loevre expansion

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. When this is the case, each distribution cannot be sampled independently, and this creates complications for many of the sampling strategies presented in this work.

Using input space mapping, however, a surrogate orthogonal input space can be constructed. This surrogate space is functionally identical to the original for our purposes. There are mathematical approaches to decoupling input parameters through surrogate spaces. In particular, using principle component analysis (or Karhunen-Loeve expansion [41] for discrete inputs), the covariance matrix for the distributed input parameters can be used to construct a multidimensional standard Gaussian normal distribution, whose components are all orthogonal. As a result, we only consider independent variables in this work, as dependent variables can be decoupled through this surrogate mapping process.

2.2 Uncertainty Quantification

The purpose of uncertainty quantification is to propagate the uncertainties present in the input space of a model through that model and comprehend their effects on the output responses. This is desirable because single-realization simulations give a very limited view of real-life operation. In traditional simulations, a single value for each input variable results in a single value for the response. When performing uncertainty quantification, a range of values for each input results in a range of response values. To quantify the distribution of the output response, often statistical moments are used, including the mean, variance, skewness, and kurtosis.

2.2.1 Statistical Moments

The four most basic statistical moments used in describing probability distributions are the mean, variance, skewness, and (excess) kurtosis. The mean (μ) provides the expected value of the response, or generally the most probable value for the response. The variance (σ^2) establishes the spread of the response, or the distance response values have from the mean on average. The standard deviation of the response is given by the square root of the variance, and provides a useful metric to determine the probability of finding a response value within a range. For instance, Chebyshev's inequality [47] says $1-1/k^2$ of a distribution's values are within k standard deviations from the mean. This is true whenever the mean and variance can be defined. Table 2.1 shows the minimum percent of the response covered by including multiples of the standard deviation from the mean. Some distributions are much more restrictive than Chebyshev's inequality

requires. For instance, we show a similar table for a normal Gaussin distribution in Table 2.2. Fig. 2.2 shows the same information graphically.

Number of Std. Dev.	Percent of Values
1	0
$\sqrt{2}$	50
2	75
3	88.89
4	93.75
5	96
10	99

TABLE 2.1: Percentage of Values within k standard deviations for general distributions

Number of Std. Dev.	Percent of Values
1	68.3
2	95.45
3	99.73
4	99.994

TABLE 2.2: Percentage of Values within k standard deviations for Gaussian normal

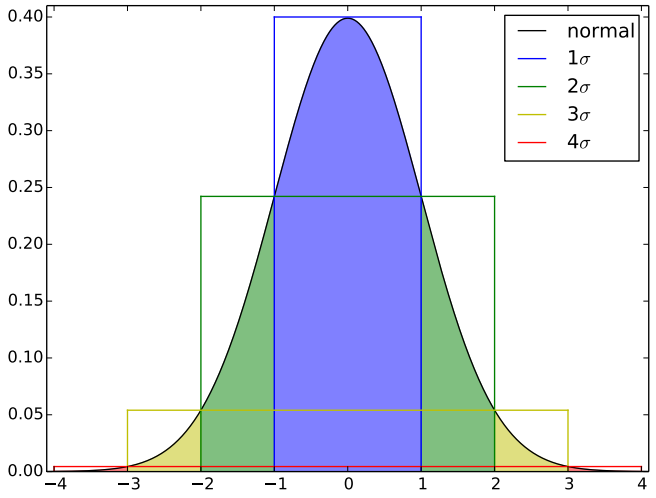

FIGURE 2.2: Standard Deviations of Normal Gaussian Distribution

Higher order moments, skewness (γ_1) and kurtosis (γ_2), describe the asymmetry and "tailedness" of the response distribution respectively. The more asymmetric the distribution, the higher the skewness is. For example, a Gaussian normal distribution has zero skewness, and skewness is introduced to a Beta distribution by allowing $\alpha \neq \beta$. Kurtosis is more complicated in its interpretation, but in general kurtosis provides an idea of how much of the variance is contributed by extreme deviations from the mean.

The kurtosis of a Guassian normal distribution is 3. This leads to the definition of excess kurtosis, which is 3 less than the traditional kurtosis.

An example of similar distributions with different moments is given in Figure 2.3. The mean shifts the entire distribution, the variance spreads the distribution, the skewness measures asymmetry, and the kurtosis measures tailedness. In each case, the blue is a "standard" distribution, and the red demonstrates increasing the indicated statistical moment.

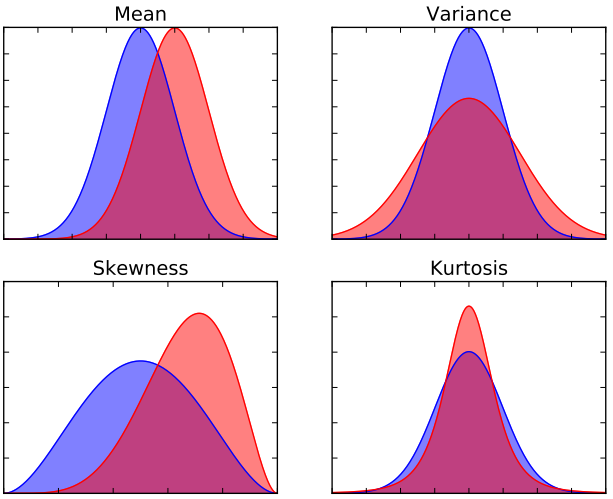

FIGURE 2.3: Visual Representation of Statistical Moments

While both skewness and kurtosis provide insight to the distribution of responses, most uncertainty quantification is centered on second-order metrics. Second-order uncertainty quantification seeks for the mean and variance of the perturbed response. Mathematically, the mean of a model is the first moment,

$$\text{mean} = \mathbb{E}[u(Y)] = \int_{\Omega} \rho(Y)u(Y)dY, \tag{2.5}$$

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

$$\text{variance} = \mathbb{E}[u(Y)^2] - \mathbb{E}[u(Y)]^2 = \int_{\Omega} \rho(Y)u(Y)^2dY - \text{mean}^2. \tag{2.6}$$

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 the mean of a response to an input is the derivative of the response with respect to the

input. Sensitivities can be either local to a region in the input space or global to the entire space.

There are two chief methods to define sensitivity. One of the most typical sensitivities is mean to mean; that is, the rate of change in the value of the response as a function of changes in the input values. This metric is most useful what attempting to maximize or minimize a response value by changing input parameters. The second method is variance to variance, or the rate of change in the variance of the response as a function of changes in the variance of an input. This is useful when trying to mitigate the spread of possible response values. If there is a possibility of a response having an undesirable value, knowing the variance-variance sensitivity helps in identifying which inputs need to have their variance reduced to prevent the undesirable value from occurring.

2.2.2 After Uncertainty Quantification

Once the response distribution is well-understood through statistical moments and sensitivities, further analysis and decisions can be made. For example, one post-uncertainty quantification analysis is *limit surface* definition and *failure probability*. In this type of analysis, a criteria is given that determines a "success" and "failure" condition for a response. For instance, in the simulation of a material undergoing stress during heating, a failure condition could be whether the material buckles during the simulation. The limit surface search seeks to determine what portion of the input space results in response failures, and what portion to successes, and define the hypersurfaces dividing successes and failures. Figure 2.4 shows a sample limit surface search sampling, and Figure 2.5 shows the surface between success and failure regions [2].

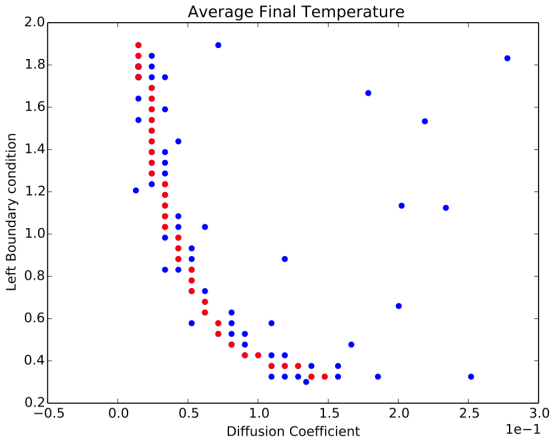

FIGURE 2.4: Limit Surface Sampling [2]

After a limit surface search, optimization can be performed, which gives some criteria for ideal operation and searches the success space for optimal inputs. For example, if

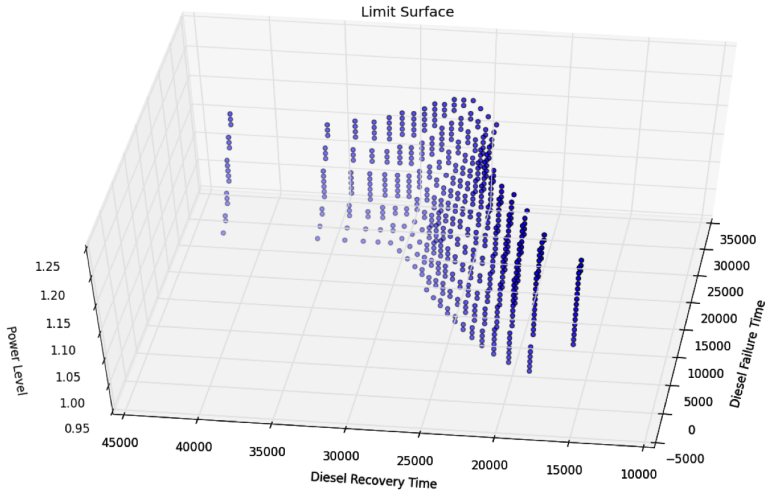

FIGURE 2.5: Limit Surface and Failure Regions [2]

alloy compositions are the inputs for the stress and heat material mentioned earlier, optimization can help find the least expensive alloy that won't buckle in the conditions given by the simulation.

Another post-uncertainty quantification calculation is to make use of sensitivity information to determine the inputs that could benefit from reduced variance to reduce the variance of the response in turn. If some inputs have minimal impact on variance in the output, they don't need the same level of care in manufacturing as other inputs. For example, consider the construction of a commercial nuclear power reactor. If the material properties and geometry of the reflector have a much smaller impact on the operation variance than the fuel content and geometry of the fuel pellets, the most naive cost-effective way to control variance is to decrease margins in fuel manufacturing instead of reflector construction. While this example seems readily evident, often engineering intuition can be informed by uncertainty quantification and sensitivity analysis.

2.2.3 Analytic Uncertainty Quantification

For some models, there exists analytic techniques for propagation of uncertainty. One of these is the so-called *sandwich formula*, often referred to as *standard propagation of error* [62]. Assuming independent input parameters (see section 2.1.3), the standard deviation σ_u of $u(Y)$ is given as

$$\sigma_u^2 = \sum_{n=1}^{N} \left(\frac{\partial u(Y)}{\partial y_n} \right)^2 \sigma_{y_n}^2, \tag{2.7}$$

where σ_{y_n} is the standard deviation of uncertain input y_n. This approximation is limited to the linear characteristics of the gradient of $u(Y)$, and so is useful especially when the standard deviation of the inputs are small compared to the partial derivatives [63].

For models with gradients that are simple to calculate accurately (and sufficiently small input uncertainties), this formula is very effective at propagating error. However, computing or estimating such gradients accurately for complex models is prohibitive, and leads to numerical approaches to uncertainty quantification, as we discuss in this thesis.

2.2.4 Uncertainty Quantification Techniques

There are several common tools used for uncertainty quantification when analytic analysis is not possible or not practical. These include stochastic methods such as Monte Carlo sampling, deterministic methods such as Grid sampling, and mixed methods such as Latin Hypercube sampling (LHS). We discuss each here and show examples of the sampling strategies.

2.2.5 Monte Carlo

The Monte Carlo method (MC) [12] 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 [13]. In its simplest form, MC involves randomly picking realizations from a set of possibilities, then statistically collecting the results. In uncertainty quantification, Monte Carlo can be used to sample realizations in the input space based on the joint probability distribution. These realizations are then run through the model solver, and the collection of response realizations is analyzed to determine its moments.

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

$$\mathbb{E}[u(Y)] = \frac{1}{M}\sum_{m=1}^{M}\left(u(Y(\omega_m))\right) + \epsilon_M^{\mathrm{MC}}, \tag{2.8}$$

where $Y(\omega_m)$ 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,

$$\epsilon_M^{\mathrm{MC}} \propto \frac{1}{\sqrt{M}}. \tag{2.9}$$

The second moment is similarly approximated as

$$\mathbb{E}[u(Y)^2] \approx \frac{1}{M} \sum_{m=1}^{M} \left(u(Y(\omega_m))^2\right). \tag{2.10}$$

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 [14][15]; 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 Grid sampling method.

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 evaluations solved. While coarse estimates are inexpensive to obtain, high precision takes a great deal of effort to converge. Figure 2.6 shows Monte Carlo sampling of a two-dimensional input space, and Figure 2.7 shows the probability weight of the sampled points [2].

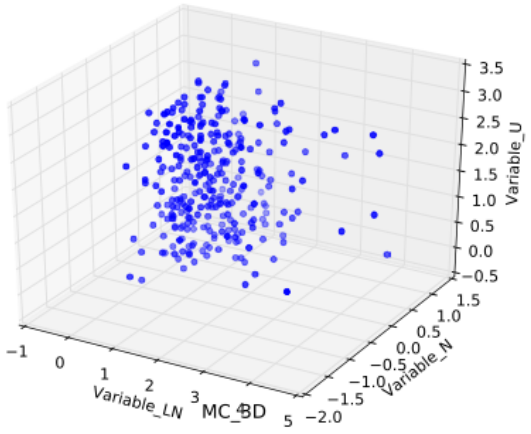

FIGURE 2.6: Example Monte Carlo Samples [2]

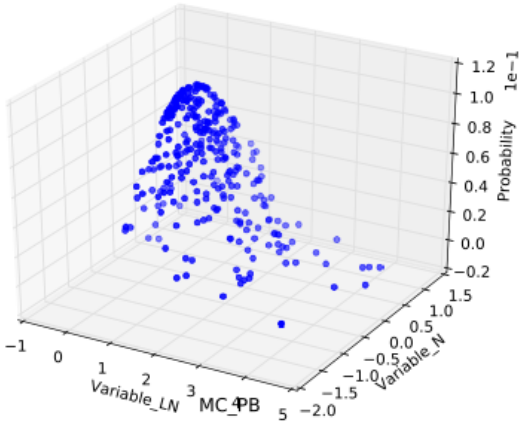

FIGURE 2.7: Example Monte Carlo Probabilities [2]

2.2.6 Grid

One of the drawbacks of Monte Carlo is lack of control over points sampled. An 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. 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 2.8, in which both equal in value and equal in CDF grid spacing is applied to a standard Gaussian normal distribution. In multidimensional input spaces, the tensor product of each grid is taken to result in the full 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

$$\mathbb{E}[u(Y)] = \int_{\Omega} \rho(Y)u(Y)dY \approx \sum_{m=1}^{M} w_m u(Y(\omega_m)), \tag{2.11}$$

where m iterates over each node in the grid, $Y(\omega_m)$ is the multidimensional input realization at grid node m, and w_m is a probability weight determined by the volume of probability represented by the grid 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,

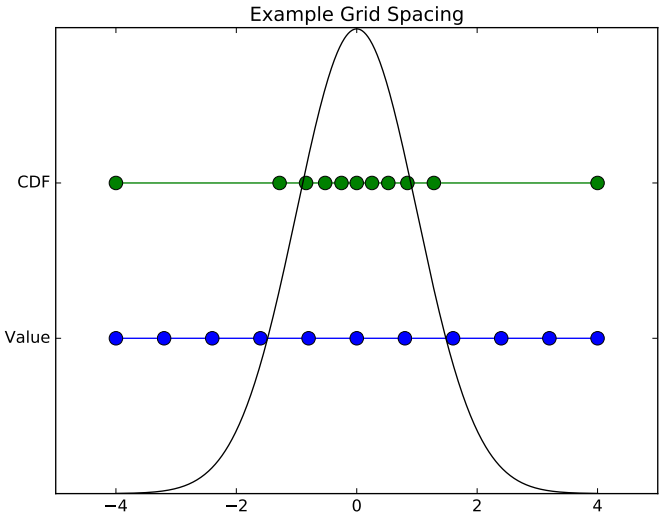

FIGURE 2.8: Grid Sampling

the second moment is approximated by

$$\mathbb{E}[u(Y)^2] = \int_\Omega \rho(Y)u(Y)^2 dY \approx \sum_{m=1}^{M} w_m u(Y(\omega_m))^2. \tag{2.12}$$

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. Figure 2.9 shows Grid sampling of a two-dimensional input space, and Figure 2.10 shows the probability weight of the sampled points [2].

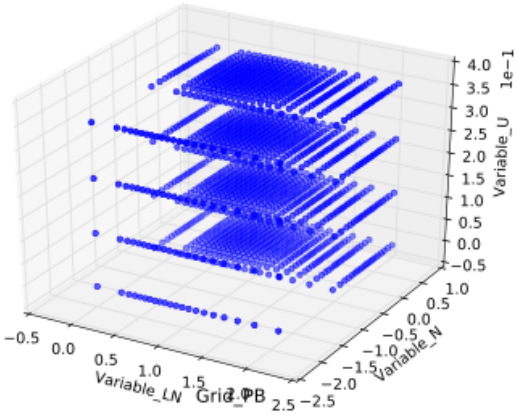

FIGURE 2.9: Example Grid Samples [2]

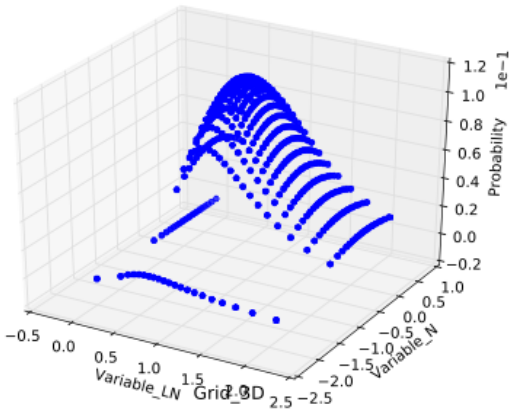

FIGURE 2.10: Example Grid Probabilities [2]

2.2.7 LHS

A cross between Monte Carlo and Grid sampling strategies, the Latin Hypercube Sampling (LHS) strategy is a sampling tool used to reduce the total samples needed without significantly sacrificing integration quality [18]. 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. Figure 2.11 shows lhs sampling of a two-dimensional input space, and Figure 2.12 shows the probability weight of the sampled points [2].

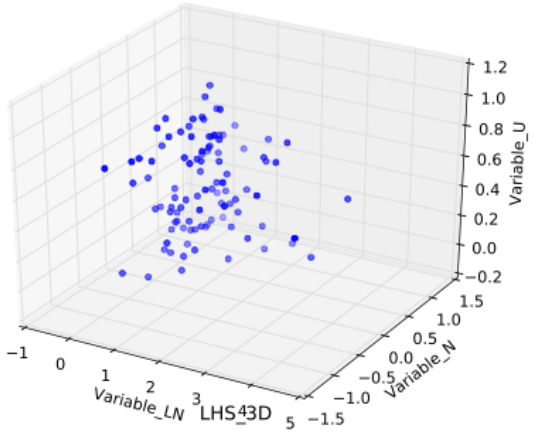

FIGURE 2.11: Example LHS Samples [2]

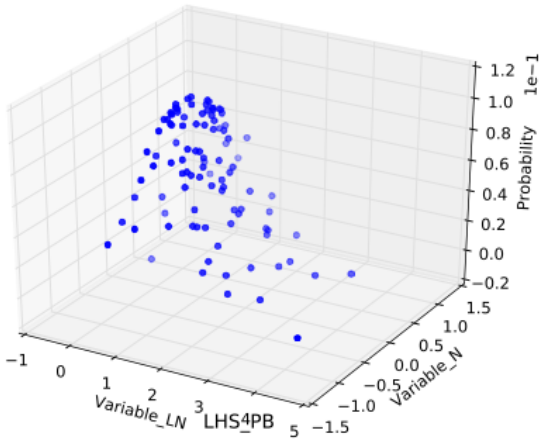

FIGURE 2.12: Example LHS Probabilities [2]

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

Chapter 3

Stochastic Collocation for Generalized Polynomial Chaos Method

3.1 Introduction

Expanding beyond the traditional uncertainty quantification methods of MC, Grid, and LHS sampling, there are more advanced methods that are very efficient in particular applications. Generalized polynomial chaos (gPC) expansion methods, for example, interpolate the model as a combination of polynomials of varying degree in each dimension of the input space. There are several advantages to expanding in polynomials [25]. 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 that can be used in place of the original model. Polynomials are generally inexpensive to evaluate, especially in comparison to complex solvers. Third, the unknowns in the expansions are scalar coefficients, which can often be efficiently calculated through numerical integration.

Originally Wiener proposed expanding models in Hermite polynomials for Gaussian-normal distributed input variables [19]. Askey and Wilson generalized Hermite polynomials to include Jacobi polynomials, including Legendre and Laguerre polynomials [26]. Xiu and Karniadakis combined these concepts to perform gPC expansions 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, in addition to Hermite polynomials for normal

distributions [25]. More information on how to use these distributions and polynomials is contained in Appendix **??**.

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) [25]. Two significant methodologies have branched from gPC applications. The first makes use of Lagrange polynomials to expand the original function or simulation code [33], as Lagrange polynomials can be made orthogonal over the same domain as the distributions; the other uses the Wiener-Askey polynomials [25]. We consider the latter in this work.

We consider a simulation code that produces a quantity of interest $u(Y)$ 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 $u(Y)$ but have no associated uncertainty; 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 gPC expansion 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. This will be discussed and demonstrated in Chapter 9. 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

$$\Phi_k(Y) = \prod_{n=1}^{N} \phi_{k_n}(Y_n), \tag{3.1}$$

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. If ϕ were taken from monomials, this

polynomial would be

$$\phi_{(2,1,4)} = y_1^2\ y_2\ y_3^4. \tag{3.2}$$

The gPC for $u(Y)$ using this notation is

$$u(Y) \approx \sum_{k \in \Lambda(L)} u_k \Phi_k(Y), \tag{3.3}$$

where u_k is a scalar weighting polynomial expansion coefficient, sometimes called a polynomial expansion moment. The polynomials used in the expansion are determined by the Λ, a set of chosen multi-indices, which can be selected in a variety of ways we will discuss in section 3.2. In the limit that Λ contains all possible combinations of polynomials of any order, Eq. 3.3 is exact. Practically, however, Λ is truncated to some finite set of multidimensional polynomials.

We make use of orthonormal Wiener-Askey polynomials

$$\int_\Omega \Phi_k(Y)\Phi_{\hat{k}}(Y)\rho(Y)dY = \delta_{k\hat{k}}, \tag{3.4}$$

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 using the properties of orthonormality.. We multiply both sides of Eq. 3.3 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,

$$u(Y) = \sum_{k \in \Lambda(L)} u_k \Phi_k(Y), \tag{3.5}$$

$$\int_\Omega u(Y)\Phi_{\hat{k}}(Y)dY = \int_\Omega u_{\hat{k}} \sum_{k \in \Lambda(L)} u_k \Phi_k(Y)\ dy, \tag{3.6}$$

$$\langle u(Y), \Phi_{\hat{k}}(Y)\rangle =< langle \Phi_{\hat{k}}(Y), \sum_{k \in \Lambda(L)} u_k \Phi_k(Y)\rangle, \tag{3.7}$$

$$\langle u(Y), \Phi_{\hat{k}}(Y)\rangle =< langle \Phi_{\hat{k}}(Y), \sum_{k \in \Lambda(L)} u_k \Phi_k(Y)\rangle, \tag{3.8}$$

$$\langle u(Y), \Phi_{\hat{k}}(Y)\rangle = \sum_{k \in \Lambda(L)} u_k \delta_{k,\hat{k}}, \tag{3.9}$$

$$\langle u(Y), \Phi_{\hat{k}}(Y)\rangle = u_{\hat{k}}, \tag{3.10}$$

and swapping $\hat{k}$ for k for convenience,

$$u_k = \langle u(Y), \Phi_k(Y)\rangle, \tag{3.11}$$

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

$$\langle f(Y), g(Y)\rangle \equiv \int_{\Omega} f(Y)g(Y)dY, \tag{3.12}$$

$$= \int_{a}^{b} f(Y)g(Y)\rho(Y)dY. \tag{3.13}$$

When $u(Y)$ has an analytic form, these coefficients can be solved by direct integration; however, in general numerical integration must be used instead. 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, domain of integration, and the orthonormal polynomials. This *stochastic collocation* numerical integration method is discussed in section 3.4, and a discussion of the synergy between probability weights, domain of integration, and the orthonormal polynomials is included in Appendix **??**. Once the polynomial expansion coefficients u_k are all obtained, the gPC expansion is complete.

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

$$\Lambda_{\mathrm{TP}}(L) = \left\{\bar{k} = (k_1, \cdots, k_N) : \max_{1\leq n\leq N} k_n \leq L\right\}. \tag{3.14}$$

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 respected. 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 [27].

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

$$\Lambda_{\mathrm{TD}}(L) = \left\{ \bar{k} = (k_1, \cdots, k_N) : \sum_{n=1}^{N} k_n \leq L \right\}. \tag{3.15}$$

The cardinality of this index set is $|\Lambda_{\mathrm{TD}}(L)| = \binom{L+N}{N}$ [27], 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.

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

$$\Lambda_{\mathrm{HC}}(L) = \left\{ \bar{k} = (k_1, \ldots, k_N) : \prod_{n=1}^{N} (k_n + 1) \leq L + 1 \right\}. \tag{3.16}$$

The cardinality of this index set is bounded by $|\Lambda_{\mathrm{HC}}(L)| \leq (L+1)(1+\log(L+1))^{N-1}$ [27]. 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 response [27]. TD tends to be most effective for

infinitely-continuous response surfaces, while HC is more effective for surfaces with limited smoothness or discontinuities.

3.2.1 Anisotropy

While using TD or HC to construct the polynomial index set combats the curse of dimensionality present in TP, the polynomial set still grows swiftly with increasing dimension, and this continues to be an issue for problems of large dimensionality. Another concept that can be applied to mitigate this issue is index set anisotropy, or the unequal treatment of various dimensions. In many models we expect different input dimensions to have different effective polynomial orders. As defined above, polynomial index sets are isotropic in dimension; anisotropy will cater to the unequal effective polynomial orders in each dimension. 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,

$$\tilde{\Lambda}_{\mathrm{TD}}(L) = \left\{\bar{p} = (p_1, \cdots, p_N) : \sum_{n=1}^{N} \alpha_n p_n \leq \frac{|\alpha|_1}{N} L\right\}, \tag{3.17}$$

$$\tilde{\Lambda}_{\mathrm{HC}}(L) = \left\{\bar{p} = (p_1, \cdots, p_N) : \prod_{n=1}^{N} (p_n + 1)^{\alpha_n} \leq (L+1)^{|\alpha|_1/N}\right\}. \tag{3.18}$$

where $|\alpha|_1$ is the one-norm of α

$$|\alpha|_1 = \sum_{n=1}^{N} \alpha_n. \tag{3.19}$$

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 chosen. Often, if a problem is well-known to an analyst, it may be enough to use intuitive judgement 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 sensitivity coefficients [48] 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-impact polynomials.

3.3 Polynomial Expansion Features

As previously mentioned, there are several benefits to the gPC expansion once constructed. First, the gPC expansion is a surrogate for the original model, and can be used in its place as long as all the inputs are within the same bounds as when the original gPC expansion 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 gPC expansion are very easy to obtain. Because the probability-weighted integral of all the Wiener-Askey polynomials is zero with the exception of the zeroth-order polynomial, and using the notation $G(y)$ to signify the gPC expansion of $u(Y)$,

$$u(Y) \approx G(Y) \equiv \sum_{k \in \lambda} u_k \Phi_k(Y), \tag{3.20}$$

the mean is simply

$$\begin{aligned} \mathbb{E}[G(Y)] &= \int_{\Omega} \sum_{k \in \Lambda} u_k \Phi_k(Y) dY, \\ &= u_{\varnothing}, \end{aligned} \tag{3.21}$$

where we use $\varnothing = (0, \cdots, 0)$. The second moment is similarly straightforward. The integral of the square of the gPC expansion 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,

$$\begin{aligned} \mathbb{E}[G(Y)^2] &= \int_{\Omega} \left[\sum_{k \in \Lambda} u_k \Phi_k(Y) \right]^2 dY, \\ &= \int_{\Omega} \sum_{k_1 \in \Lambda} \sum_{k_2 \in \Lambda} u_{k_1} \Phi_{k_1}(Y) \cdot u_{k_2} \Phi_{k_2}(Y) dY, \\ &= \sum_{k_1 \in \Lambda} \sum_{k_2 \in \Lambda} u_{k_1} \cdot u_{k_2} \delta_{k_1, k_2}, \\ &= \sum_{k \in \Lambda} u_k^2. \end{aligned} \tag{3.22}$$

3.4 Stochastic Collocation

Having outlined the gPC expansion construction and its uses, we turn to the method of calculating the polynomial expansion coefficients. 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 (see Appendix A). Quadrature integration takes the form

$$\int_a^b f(x)\rho(x) = \sum_{\ell=1}^{\infty} w_\ell f(x_\ell), \tag{3.23}$$

$$\approx \sum_{\ell=1}^{\hat{L}} w_\ell f(x_\ell), \tag{3.24}$$

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

$$q^{(\hat{L})}[f(x)] \equiv \sum_{\ell=1}^{\hat{L}} w_\ell f(x_\ell). \tag{3.25}$$

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

$$Q^{(\mathbf{L})} = q_1^{(\hat{L}_1)} \otimes q_2^{(\hat{L}_2)} \otimes \cdots, \tag{3.26}$$

$$= \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}. \tag{3.27}$$

It is worth noting each dimension's 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 optimally 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. 3.11.

$$u_k = \langle u(Y)\Phi_k(Y)\rangle. \tag{3.28}$$

We can approximate this integral with the appropriate Gaussian quadrature as

$$u_k \approx Q^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)], \tag{3.29}$$

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.

$$u_k \approx q^{(\hat{L})}[u(Y)\Phi_k(Y)], \tag{3.30}$$

$$= \sum_{\ell=1}^{\hat{L}} w_\ell u(Y_\ell)\Phi_k(Y_\ell). \tag{3.31}$$

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. 3.3 and replace it in the sum,

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell \Phi_k(Y_\ell) \sum_{k \in \Lambda(L)} u_{\hat{k}} \Phi_{\hat{k}}(Y_\ell). \tag{3.32}$$

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

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell u_k \Phi_k(Y_\ell)^2. \tag{3.33}$$

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

$$p = \frac{2k+1}{2} = k + \frac{1}{2} < k + 1, \tag{3.34}$$

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. In this case, a "quadrature floor" can be enforced, limiting the smallest allowable quadrature with which to integrate any expression.

The naive choice for multidimensional quadrature is to find the largest quadrature needed in each dimension, then take the grid of all possible combinations of the largest quadrature points as the multidimensional points and weights. While a tensor product of highest-necessary quadrature orders could serve as a suitable multidimensional quadrature set, the necessary model evaluations grows very quickly. We can make use of Smolyak-like sparse quadratures to reduce the number of model evaluations necessary for the TD and HC polynomial index set construction strategies.

3.4.1 Smolyak Sparse Grids

Smolyak sparse grids [28] are an attempt to discover the smallest necessary quadrature set to integrate a multidimensional integral with varying orders of predetermined quadrature sets. For example, consider a two-dimensional input space with a hyperbolic cross index set (see Table 3.2). The highest quadrature order needed in y_1 is fourth order for $k = (0, 3)$, and the highest quadrature order needed in y_2 is also fourth order for $k = (3, 0)$. Traditionally, this would require a multidimensional quadrature with 16 (four by four) quadrature points. However, to integrate the polynomial $k = (1, 1)$, only second order in y_1 by second order in y_2 is required, and the single-dimensional fourth-order quadratures are sufficient to cover all the other polynomials. As a result, only 12 quadrature points are needed (four for single-dimensional y_1 quadrature, four for single-dimensional y_2 quadrature, and four for two-by-two mixed quadrature). Even for this low-order simple example, the reduced sparse grid has three-fourths the points of the original tensor quadrature. This type of savings is the principle idea behind Smolyak sparse quadrature. Additional discussion and some example figures are included in [10].

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

$$Q^{(2,1,3)} = q_1^{(3)} \otimes q_2^{(2)} \otimes q_3^{(4)}. \tag{3.35}$$

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

$$Q^{(\Lambda(L))} = \bigotimes_{k \in \Lambda} Q^{(k)}. \tag{3.36}$$

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 quadrature difference Δ notation [37]

$$\Delta_k^{(\hat{L})}[f(x)] \equiv \left(q_k^{(\hat{L})} - q_{k-1}^{(\hat{L})}\right)[f(x)], \tag{3.37}$$

$$q_0^{(\hat{L})}[f(x)] = 0. \tag{3.38}$$

A Smolyak sparse grid quadrature operator $\mathcal{S}$ is then defined and applied to the desired integral in Eq. 3.11,

$$S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)] = \sum_{k \in \Lambda(L)} \left(\Delta_{k_1}^{(\hat{L}_1)} \otimes \cdots \otimes \Delta_{k_N}^{(\hat{L}_N)}\right)[u(Y)\Phi_k(Y)]. \tag{3.39}$$

Equivalently, and in a more algorithm-friendly approach [36],

$$S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)] = \sum_{k\in\Lambda(L)} c(k) \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}[u(Y)\Phi_k(Y)] \tag{3.40}$$

where

$$c(k) = \sum_{\substack{j=\{0,1\}^N, \\ k+j\in\Lambda}} (-1)^{|j|_1}, \tag{3.41}$$

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

$$u_k = \langle u(Y)\Phi_k(Y)\rangle, \tag{3.42}$$

$$\approx S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)]. \tag{3.43}$$

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 [30] and used in [10]. Essentially, the polynomial index set Λ is constructed one polynomial at a time by choosing from *prospective* polynomials. Prospective polynomials are defined as those for whom all lower-order subset polynomials have already been included. The algorithm proceeds generally as follows:

1. Begin with the mean (zeroth-order) polynomial expansion.
2. While not converged:
   (a) Collect a list of the prospective polynomial index set whose predecessors have all been evaluated.
   (b) Calculate the impact of adding each prospective polynomial to the existing polynomial index set.
   (c) If the total impact of all prospective polynomials is less than tolerance, convergence is reached.
   (d) Otherwise, add the predicted highest-impact prospective polynomial and loop back to 2a.

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.

Figures 3.1 and 3.2 show a single adaptive step and the progression of multiple steps, respectively, for a demonstrative two-dimensional model. In each, the algorithm progresses from the upper left diagram to the lower right. The blue squares indicate polynomials already included in the gPC expansion, and the green circle shows the next selected polynomial to include. It can be seen how the algorithm is including more polynomials along the x-axis variable than the y-axis variable because x has a higher impact on the response for this arbitrary model.

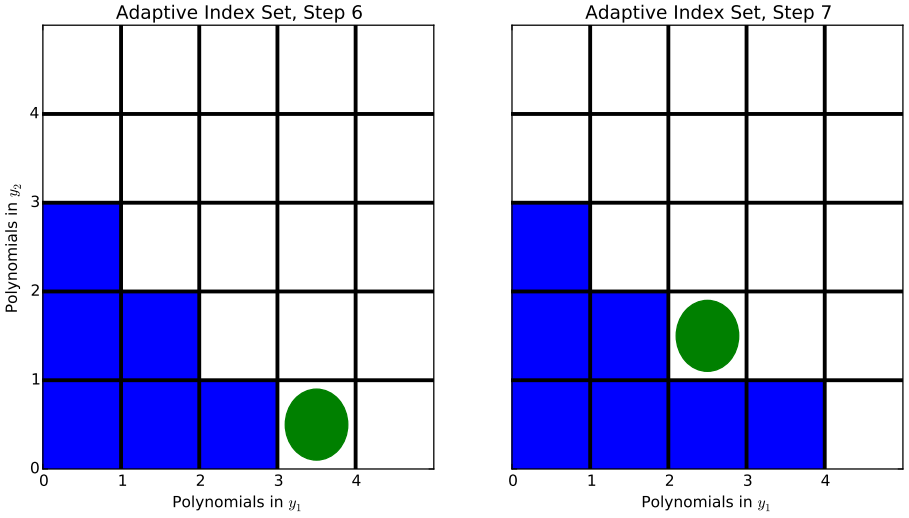

FIGURE 3.1: Adaptive Sparse Grid Step

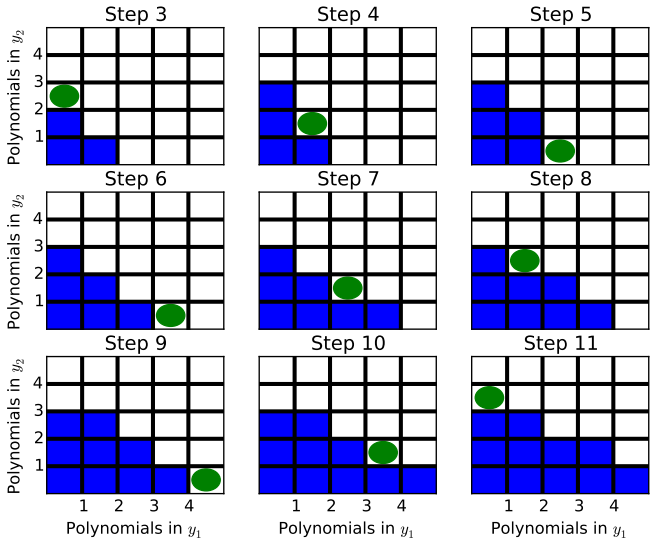

FIGURE 3.2: Adaptive Sparse Grid Progression

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 in step 2b in the adaptive algorithm above; instead, it evaluates each prospective polynomial and selects the one with the most impact. This is somewhat inefficient, because many SCgPC expansion coefficients that are calculated are not used in the final expansion. One improvement we make to this algorithm is to predict the impact of prospective polynomials based on the impact of predecessors.

In order to predict the most valuable polynomial to add to an expansion during an adaptive search, we first identify a metric by which different polynomials can be compared to determine impact. 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* parameter η_k of a polynomial to be the square of its polynomial expansion coefficient,

$$\eta_k = u_k^2. \tag{3.44}$$

This choice is made because the total variance is the sum of the square of the polynomial expansion coefficients; therefore, the impact of the square of each polynomial coefficient is also the impact on the variance. To estimate the impact of a polynomial whose coefficient is not yet calculated, we consider the average of the proceeding polynomials. That is, for a polynomial $k = (3, 2, 4)$ we average the impacts of $(2, 2, 4)$, $(3, 1, 4)$, and $(3, 1, 3)$,

$$\tilde{\eta}_k = \frac{1}{N - j} \sum_{n=1}^{N} \eta_{k - e_n}, \tag{3.45}$$

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 many evaluations 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

$$f(a, b) = a^3 b^3, \tag{3.46}$$

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 $k = (0, 0)$ are zero. The first polynomial index set point to evaluate is zeroth-order quadrature in each dimension, $\hat{L}_k = (0, 0)$. The quadrature point to evaluate this polynomial coefficient is $Y_\omega = (0, 0)$, which, when evaluated, gives $f(0, 0) = 0$.

The next polynomial index set combinations are $k = (0, 1)$ and $k = (1, 0)$. For $k = (0, 1)$, the quadrature points required are $Y_\omega = (0, \pm\sqrt{1/3})$. This evaluates to $f(0, \pm\sqrt{1/3}) = 0$, as well. Because of model symmetry, we obtain the same result for $k = (1, 0)$. 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 consider. 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 not often contribute to an accurate expansion unless lower-order polynomials contribute as well.

Chapter 4

Results for Stochastic Collocation for Generalized Polynomial Chaos

4.1 Introduction

In this chapter we present results obtained using stochastic collocation for generalized polynomial chaos expansions (SCgPC) using the three presented static polynomial sets (tensor product, hyperbolic cross, and total degree) as well as the adaptive construction approach for polynomial sets. We present performance on a variety of increasingly-complex models, from linear polynomials to discontinuous products. For each model, we demonstrate the efficiency of each collocation method on a variety of input space sizes where possible. The increasingly complex models in addition to the increasing input space sizes will provide a survey of where collocation-based methods clearly outperform Monte Carlo (MC) and where they fall short.

We use these six analytic models as a means to study convergence on the first two statistical moments of the response. This requires a high-precision evaluation of the moments, which is not practical for non-analytic models. We are concerned chiefly with the convergence rate of each model; that is, for the cost of increasing the number of computation solves, how much error can be reduced. Because Monte Carlo converges linearly on a log-log plot of error versus solves, this linear convergence provides the benchmark for collocation methods.

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, 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. MC performance is taken at a few representative points. Error bars are provided for MC and are estimated using the population variance,

$$\epsilon_{95} = \frac{2\bar{\sigma}_M}{\sqrt{M}}, \tag{4.1}$$

$$\bar{\sigma}_M^2 = \frac{M}{M-1}\sigma_M^2 = \frac{M}{M-1}\left(\frac{1}{M}\sum_{m=1}^{M} u(Y(\omega_m))^2 - \bar{u}(Y)_N^2\right), \tag{4.2}$$

where $Y(\omega_m)$ are a set of M independent realizations taken from the identically-distributed input space. These error bars estimate where the value of the statistic is with a probability of at least 0.75, as discussed in section 2.2.1. The estimate of this error improves as additional samples are taken.

Convergence figures are log-log error graphs with the number of computational solves required on the x-axis and error with respect to the analytic solution on the y-axis. The distinct series plotted demonstrate results obtained for each UQ method. The series we show here are analog traditional MC; static SCgPC expansions using the hyperbolic cross index set (SC:HC), total degree index set (SC:TD), and (where possible) tensor product index set (SC:TP); and adaptive SCgPC (SC:adapt). Each series obtains additional values by increasing the refinement of the method. For MC, additional random 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.

The measure of success for a method is not dependent on the absolute value of the error shown. While informative, we are more concerned with how increasing refinement reduces error. The rate of convergence as refinement increases determines the desirability of the method for that model. We expect the rate of convergence to depend primarily two factors: the dimensionality of the uncertain space for the model, and the continuity or smoothness of the response measured. The value of the error, on the other hand, will additionally depend on how well a particular choice of polynomials matches the analytic polynomial representation of the model. We consider the convergence of both the mean and the standard deviation for each model.

The uncertainty quantification analysis in this chapter is all performed in `RAVEN` [2] on external python models. 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

4.2.1 Description

The simplest model we make use of is a first-order tensor polynomial (tensor monomial) combination [10]. Each term in this polynomial expression is at most linear in any dimension. This provides a simple calculation of the statistical moments, and no second-order polynomials are required to exactly reproduce this model. The mathematical expression for tensor monomials is

$$u(Y) = \prod_{n=1}^{N} (y_n + 1). \tag{4.3}$$

For example, for $N = 3$ we have

$$u(Y) = y_1 y_2 y_3 + y_1 y_2 + y_1 y_3 + y_2 y_3 + y_1 + y_2 + y_3 + 1. \tag{4.4}$$

For this model we can distribute the uncertain inputs in several ways because of its simplicity: uniformly on [-1,1], uniformly on [0,1], and normally on $[\mu, \sigma]$. A summary of analytic statistics is given in Table 4.1. The two-dimensional representation of this response is given in Figure 4.1.

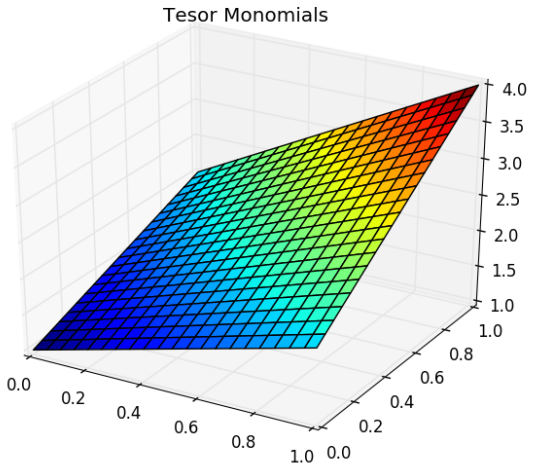

FIGURE 4.1: Tensor Monomials Response

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 4.1: Analytic Expressions for Tensor Monomial Case

For purposes of demonstration, we pick several increasing orders of dimensionality: three input variables, five variables, and ten variables.

4.2.2 Discussion

As this polynomial contains only combinations of first-order polynomials, we expect the Tensor Product index set construction method to be very efficient in absolute error. As such, it will be difficult to observe the convergence rate for this method, as it converges exactly with first-order polynomials. Because the model has infinite continuity, we expect all collocation-based methods to be quite efficient. Plots with the values and errors of the mean and standard deviation are given for each of three (Figures 6.1 through 6.4), five (Figures 6.5 through 6.8), and ten (Figures 6.9 through 6.12) input parameters. Note especially that TP exactly reproduces the original model with expansion order 1, so no convergence is observed past the initial sampling point.

4.2.3 Tensor Monomials: 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. The convergence of the mean is instant for all methods, since the linear nature of the problem means only the zeroth-order polynomial term is required to exactly reproduce the mean. More convergence behavior can be seen for the standard deviation. Because hyperbolic cross polynomials emphasize single-variable polynomials over cross terms, it is the slowest to reach effectively zero error. Similarly, the total degree quickly obtains most of the polynomials in the exact expansion, but takes a few levels to include the term that has all the input variables in it. Because first-order tensor product polynomials is exactly the model itself, it converges instantly. The adaptive SCgPC method initially explores high-order polynomials before finding the remaining tensor monomials, which makes it less directly efficient than tensor product but otherwise desirable over the other two static polynomial sets.

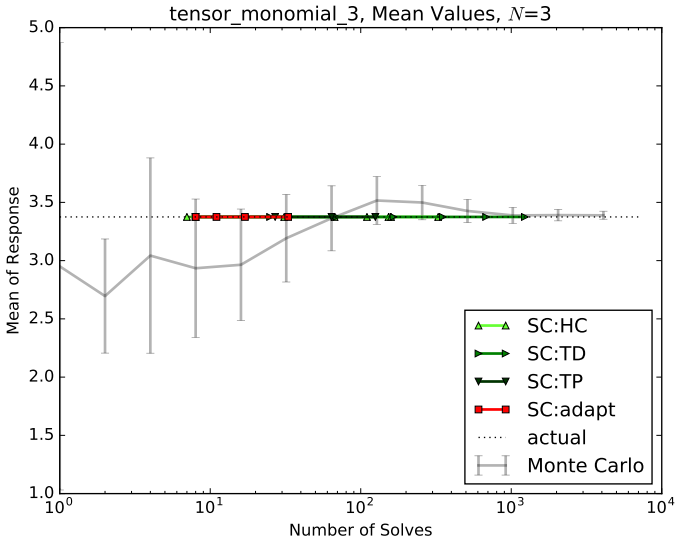

FIGURE 4.2: Tensor Monomial, $N = 3$, Mean Values

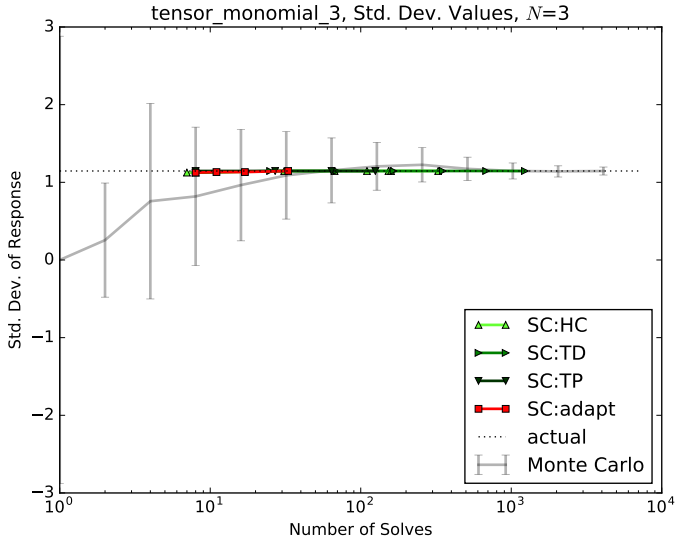

FIGURE 4.3: Tensor Monomial, $N = 3$, Std. Dev. Values

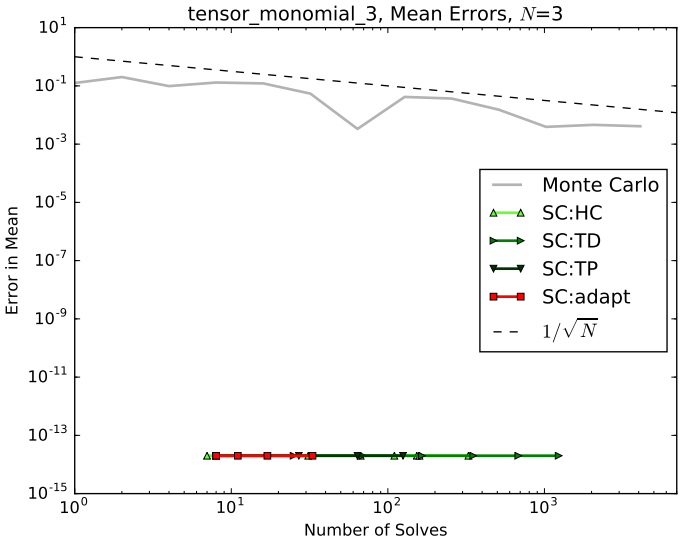

FIGURE 4.4: Tensor Monomial, $N = 3$, Mean Convergence

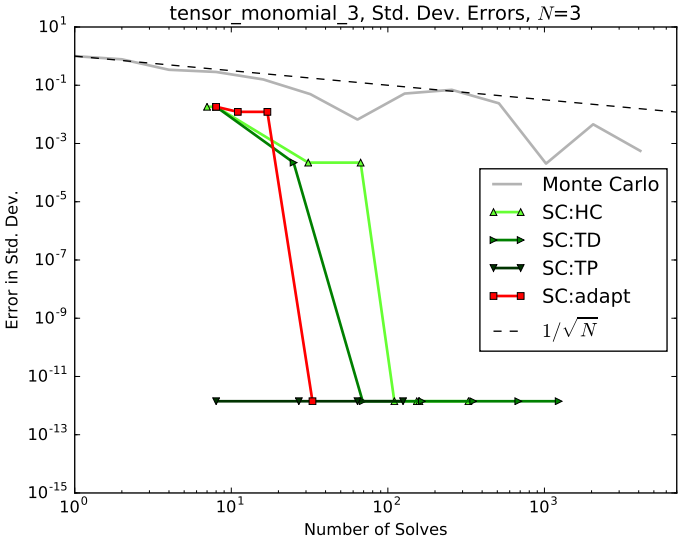

FIGURE 4.5: Tensor Monomial, $N = 3$, Std. Dev. Convergence

4.2.4 Tensor Monomials: 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 on the standard deviation for collocation-based methods. As with the three variable case, the mean is trivial and obtained with the zeroth-order polynomial. Exponential convergence can be seen for the total degree and hyperbolic cross polynomials, while the adaptive method is still exploring higher-order polynomials as more likely candidates for inclusion in the expansion and hasn't seen the

same rapid convergence curve yet. This is a flaw in the default search parameters for the adaptive algorithm when applied to this model. Total Degree outperforms Hyperbolic Cross and Adaptive, as the adaptive search algorithm struggles 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.

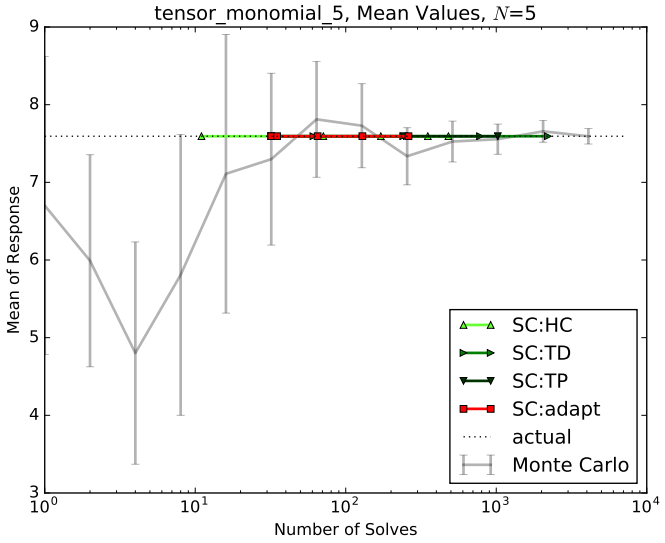

FIGURE 4.6: Tensor Monomial, $N = 5$, Mean Values

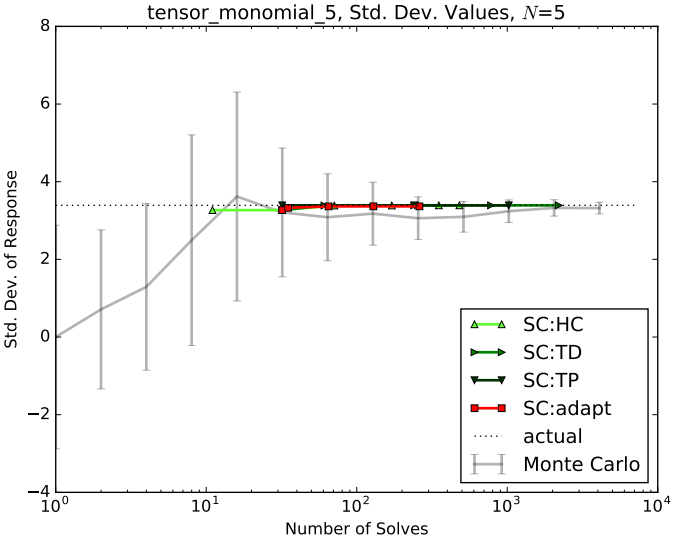

FIGURE 4.7: Tensor Monomial, $N = 5$, Std. Dev. Values

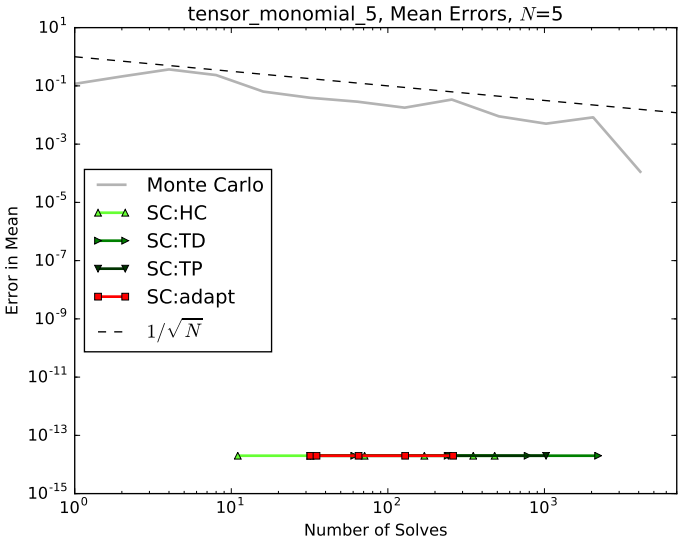

FIGURE 4.8: Tensor Monomial, $N = 5$, Mean Convergence

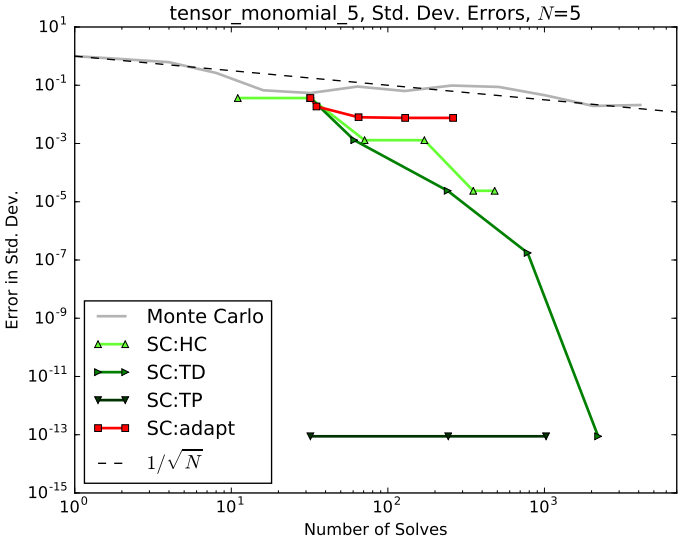

FIGURE 4.9: Tensor Monomial, $N = 5$, Std. Dev. Convergence

4.2.5 Tensor Monomials: 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 might be exponential convergence, the curvature is quite large, and only somewhat better than linear convergence is observed for up to 1000 computational solves. One reason the adaptive method does 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 ten of the inputs.

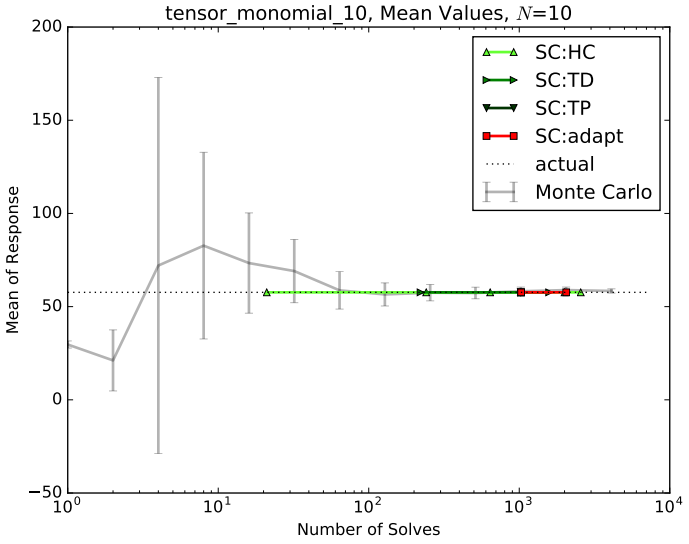

FIGURE 4.10: Tensor Monomial, $N = 10$, Mean Values

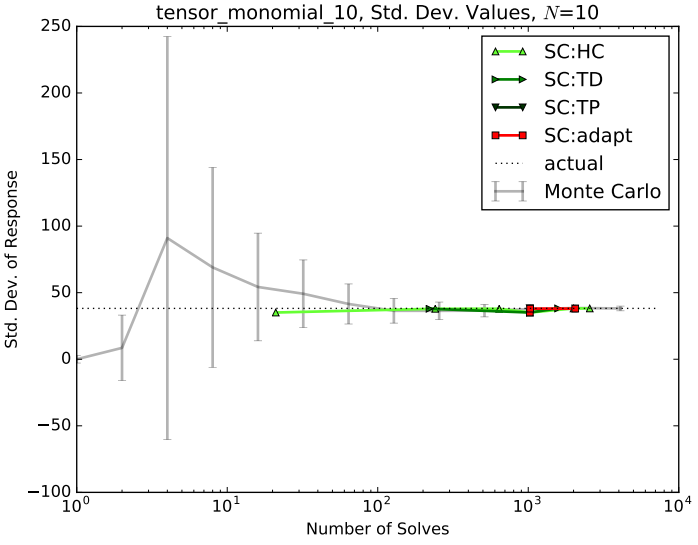

FIGURE 4.11: Tensor Monomial, $N = 10$, Std. Dev. Values

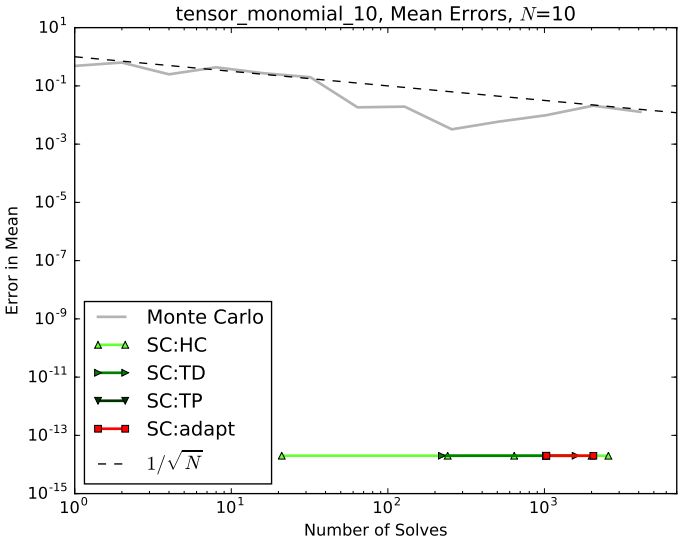

FIGURE 4.12: Tensor Monomial, $N = 10$, Mean Convergence

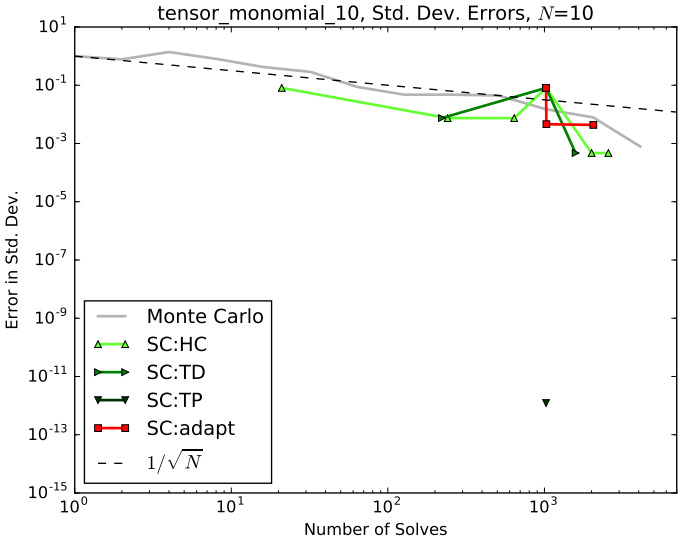

FIGURE 4.13: Tensor Monomial, $N = 10$, Std. Dev. Convergence

4.3 Sudret Polynomial

4.3.1 Description

The polynomial used by Sudret in his work [43] is another tensor-like polynomial, and is a test case traditionally used to identify convergence on sensitivity parameters. It is similar to tensor monomials because it is constructed by the tensor product of simple polynomials; in this case, Sudret used second-order polynomials. As a result, only zeroth

or second-order polynomials exist in the expression. Statistical moments are also quite straightforward for this model. The mathematical expression for Sudret polynomials is

$$u(Y) = \frac{1}{2^N} \prod_{n=1}^{N} (3y_n^2 + 1). \tag{4.5}$$

The two-dimensional representation of this response is given in Figure 4.14. The vari-

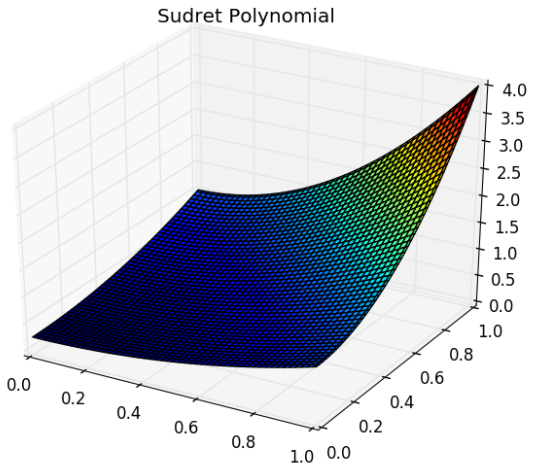

FIGURE 4.14: Sudret Polynomial Response

ables are distributed uniformly on [0,1].

The statistical moments and sensitivities for this model are given in Table 4.2, 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 4.2: Analytic Expressions for Sudret Case

Because of its similarity to tensor polynomials, the cases we show are three inputs and five inputs.

4.3.2 Discussion

The Sudret polynomial model is a near neighbor to the tensor monomials model; however, it includes only even-ordered polynomials. Because the model is still a tensor

product, the tensor product collocation method converges most directly in all dimensionality cases.

An interesting feature of the mean for this model is that the zeroth-order expansions tend to estimate the variance more accurately than the first-order expansions; as a result, it appears that the error is very small then grows rapidly. This is misleading to considering convergence, however, as the initial approximation exhibits some cancellation of errors to obtain such an "accurate" result.

4.3.3 Sudret Polynomial: 3 Inputs

As with the tensor monomials, we see a good rate of convergence for many of the polynomial methods. With this model, the mean is not trivially given by the zeroth-order polynomial, and so some convergence is seen in obtaining the expected value. The total degree and adaptive methods converge at a similar rate for the mean, while the hyperbolic cross demonstrates its poor convergence for highly regular systems with nonlinear cross-term effects. Quickest to converge (aside from the tensor product case) is the adaptive SCgPC, because its search method allows it to discover the second-order polynomials quickly.

Similar behavior is seen for the standard deviation. The tensor product still converges very rapidly, and total degree shows a good rate of convergence, while hyperbolic cross demonstrates poor convergence.

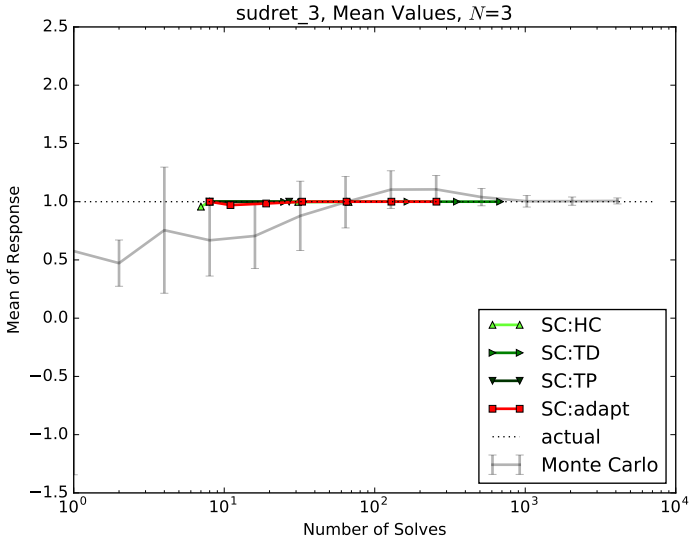

FIGURE 4.15: Sudret Polynomial, $N = 3$, Mean Values

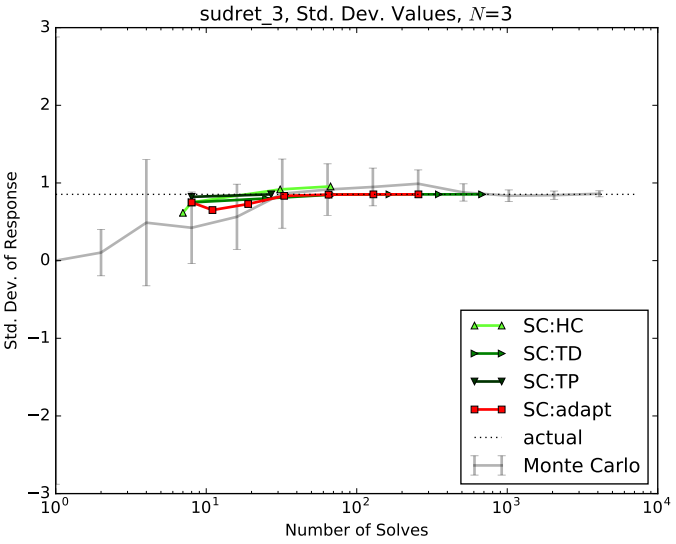

FIGURE 4.16: Sudret Polynomial, $N = 3$, Std. Dev. Values

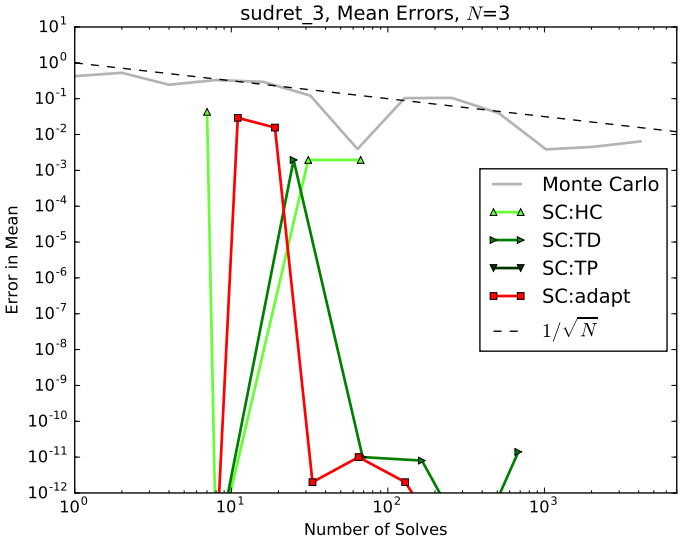

FIGURE 4.17: Sudret Polynomial, $N = 3$, Mean Convergence

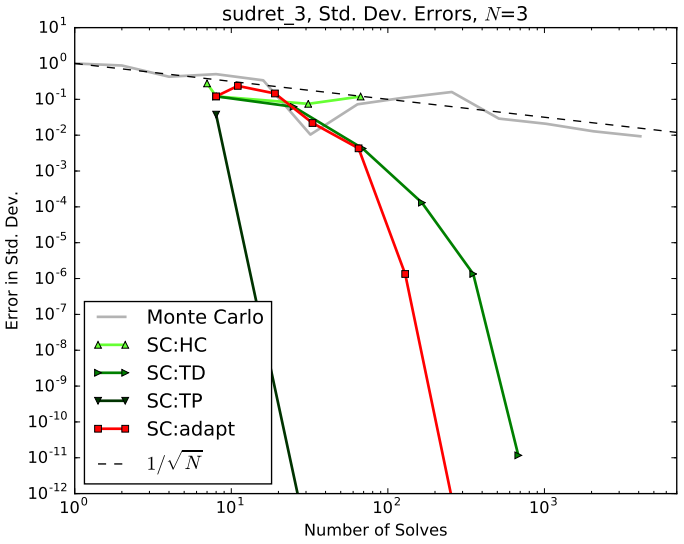

FIGURE 4.18: Sudret Polynomial, $N = 3$, Std. Dev. Convergence

4.3.4 Sudret Polynomial: 5 Inputs

In the five-input case for the Sudret polynomials, we see slower but strong convergence for adaptive and total degree methods. For the mean, each method is showing some level of exponential convergence, with the exception of the hyperbolic cross method. For the standard deviation, however, the radius of curvature for the convergence is quite large. This demonstrates the negative impact growing input spaces have on the effectiveness of collocation methods in comparison with MC.

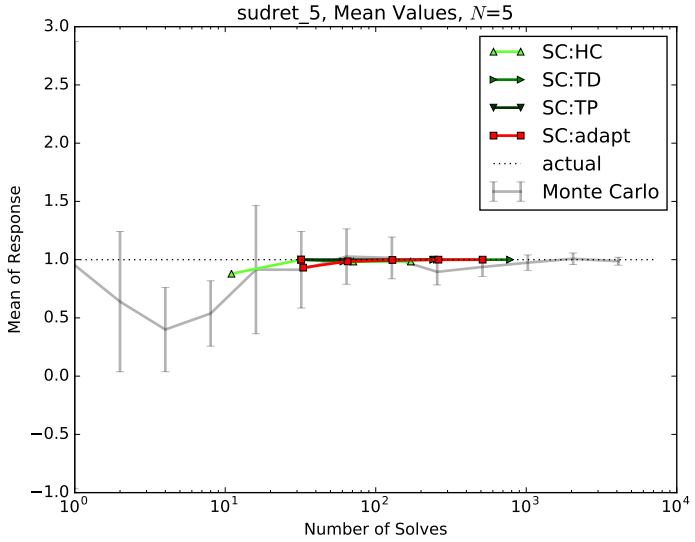

FIGURE 4.19: Sudret Polynomial, $N = 5$, Mean Values

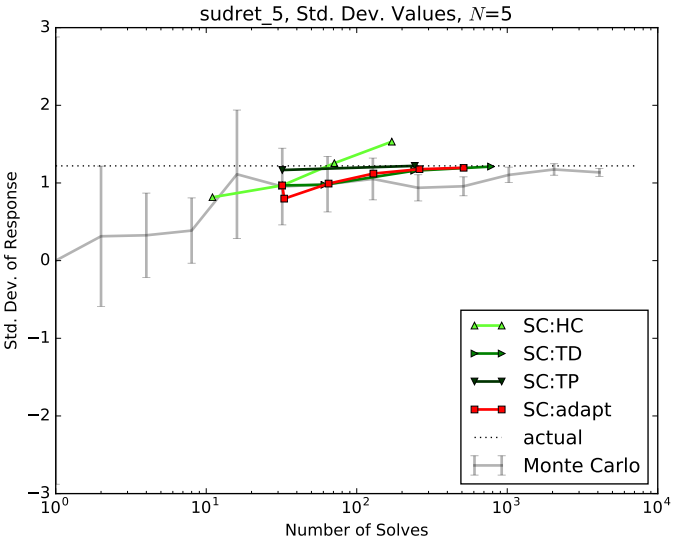

FIGURE 4.20: Sudret Polynomial, $N = 5$, Std. Dev. Values

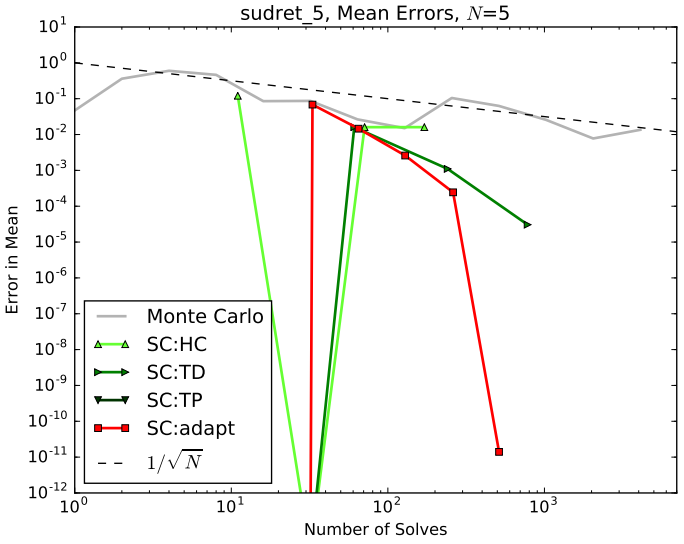

FIGURE 4.21: Sudret Polynomial, $N = 5$, Mean Convergence

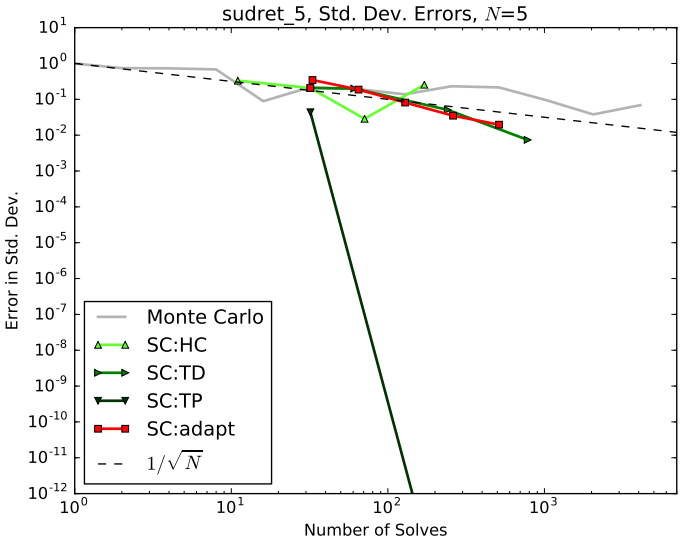

FIGURE 4.22: Sudret Polynomial, $N = 5$, Std. Dev. Convergence

4.4 Attenuation

4.4.1 Description

While this model is effectively a tensor product of polynomials and has an analytic response, it also is the model for a common physical problem. Consider a one-dimensional geometry that consists of a material with unit length and vacuum to the left and right of the material. We consider a beam of neutral particles that have a probability of interacting with the material through absorption, or passing through it. This beam enters the material on the left and exits on the right with a fraction of its original flux. See for example Figure **??**, where the dark arrows represent beam travel direction and their thickness generally depicts the beam attenuating through the material. The spatial domain is [0,1]. The quantity of interest is the percent of particles that pass through the material without interacting anywhere along its length. The boundary conditions for this problem are a constant positive current on the left boundary, and a vacuum on the right boundary.

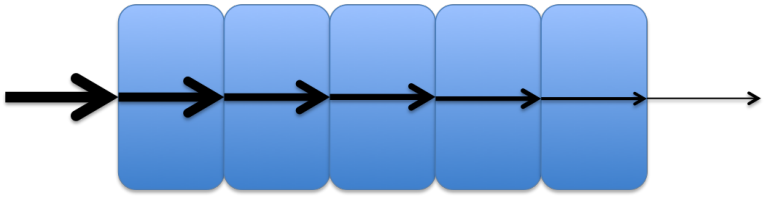

FIGURE 4.23: Attenuation Model Visualization ($N = 5$)

This model represents the idealized single-dimension system where an beam of particles impinges on a purely-absorbing material with total scaled length of 1. The material is divided into N segments, each of which has a distinct uncertain interaction cross section y_n. The cross section has units of probable interactions per unit length, and the integral of a cross section over a length provides the probability of interaction within that length. The response (percent of the beam to pass out the right boundary) takes the form

$$u(Y) = \prod_{n=1}^{N} \exp(-y_n/N). \tag{4.6}$$

The two-dimensional representation of this response is given in Figure 4.24. Because

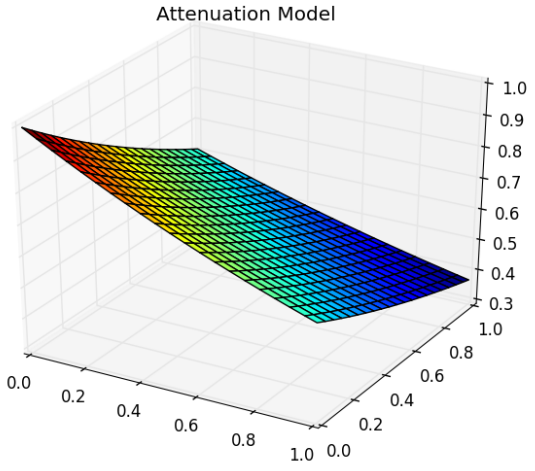

FIGURE 4.24: Attenuation Model Response

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 4.3.

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

TABLE 4.3: 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 (about the origin) includes all increasing

		Polynomial Order (y_1)				
		0	1	2	3	4
Polynomial Order (y_2)	0	1	a	$a^2/2$	$a^3/6$	$a^4/24$
	1	a	a^2	$a^3/2$	$a^4/6$	$a^5/24$
	2	$a^2/2$	$a^3/2$	$a^4/4$	$a^5/12$	$a^6/48$
	3	$a^3/6$	$a^4/6$	$a^5/12$	$a^6/36$	$a^7/144$
	4	$a^4/24$	$a^5/24$	$a^6/48$	$a^7/144$	$a^8/576$

TABLE 4.4: Coefficient Magnitudes, Tensor Taylor Development of e^{-ay}

polynomial orders,

$$e^{-ay} = 1 - ay + \frac{(ay)^2}{2} - \frac{(ay)^3}{6} + \frac{(ay)^4}{24} - \frac{(ay)^5}{120} + \mathcal{O}(y^6). \tag{4.7}$$

As a result, the product of several exponential functions is effectively a tensor combination of polynomials for each dimension. The coefficients of higher-order polynomials are smaller than lower-order polynomials. Further, coefficients for combined polynomials have larger coefficients than single-dimensional polynomials with the same effective order. For example, for a two-dimensional exponential function and considering effective fourth-order polynomials, the coefficient for $y_1^2 y_2^2$ is $a^4/4$, while the coefficient for y_1^4 is $a^4/24$. This is shown graphically in Table 4.5. The magnitude of each coefficient for the tensor product of two Taylor expansions of the exponential function are given tabularly. The grayed entries are those for which the total polynomial order is four. We can see that the coefficients for polynomials equally split between the two input variables have a smaller denominator than those made up entirely of one input variable, and so those that are equal in both dimensions are more important to include in the expansion than those that are solely of one order or another. Note that in our case, $a = 1/N$, which causes the polynomial coefficients to drop off more quickly, so that low-order polynomial orders are more important to the expansion.

4.4.2 Discussion

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 (about the origin) includes all increasing polynomial orders,

$$e^{-ay} = 1 - ay + \frac{(ay)^2}{2} - \frac{(ay)^3}{6} + \frac{(ay)^4}{24} - \frac{(ay)^5}{120} + \mathcal{O}(y^6). \tag{4.8}$$

		Polynomial Order (y_1)				
		0	1	2	3	4
Polynomial Order (y_2)	0	1	a	$a^2/2$	$a^3/6$	$a^4/24$
	1	a	a^2	$a^3/2$	$a^4/6$	$a^5/24$
	2	$a^2/2$	$a^3/2$	$a^4/4$	$a^5/12$	$a^6/48$
	3	$a^3/6$	$a^4/6$	$a^5/12$	$a^6/36$	$a^7/144$
	4	$a^4/24$	$a^5/24$	$a^6/48$	$a^7/144$	$a^8/576$

TABLE 4.5: Coefficient Magnitudes, Tensor Taylor Development of e^{-ay}

As a result, the product of several exponential functions is effectively a tensor combination of polynomials for each dimension. The coefficients of higher-order polynomials are smaller than lower-order polynomials. Further, coefficients for combined polynomials have larger coefficients than single-dimensional polynomials with the same effective order. For example, for a two-dimensional exponential function and considering effective fourth-order polynomials, the coefficient for $y_1^2 y_2^2$ is $a^4/4$, while the coefficient for y_1^4 is $a^4/24$. This is shown graphically in Table 4.5. The magnitude of each coefficient for the tensor product of two Taylor expansions of the exponential function are given tabularly. The grayed entries are those for which the total polynomial order is four. We can see that the coefficients for polynomials equally split between the two input variables have a smaller denominator than those made up entirely of one input variable, and so those that are equal in both dimensions are more important to include in the expansion than those that are solely of one order or another. Note that in our case, $a = 1/N$, which causes the polynomial coefficients to drop off more quickly, so that low-order polynomial orders are more important to the expansion. As a result, we see good convergence from the collocation methods generally.

4.4.3 Attenuation: 2 Inputs

With only two input parameters, we see excellent convergence for all methods in the response mean, while Hyperbolic Cross struggles to accurately represent the standard deviation. As mentioned in the description, this is likely because monomials are less important in the Taylor representation, while Hyperbolic Cross emphasizes monomials over cross terms. Interestingly, while Tensor Product demonstrates the smallest error, its apparent curvature is slightly larger than for the other methods. Because of the small input space, the total degree, hyperbolic cross, and adaptive methods all perform similarly. Because the adaptive method uses the impact of previous polynomials to choose future polynomials, its convergence rate appears to improve as it continues.

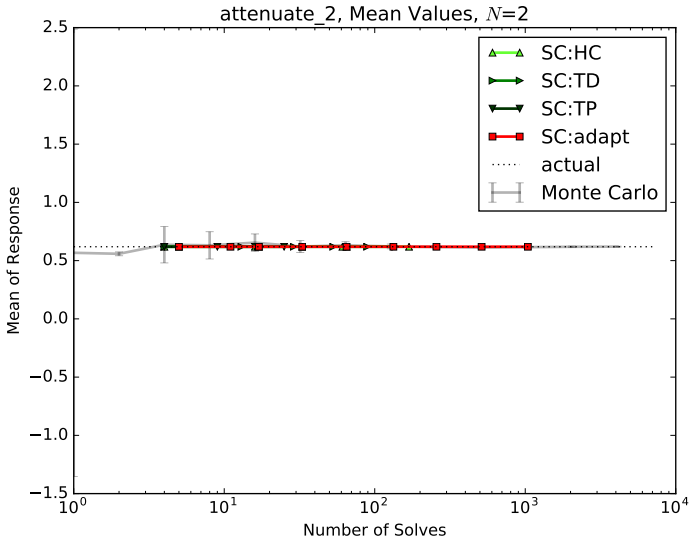

FIGURE 4.25: Attenuation, $N = 2$, Mean Values

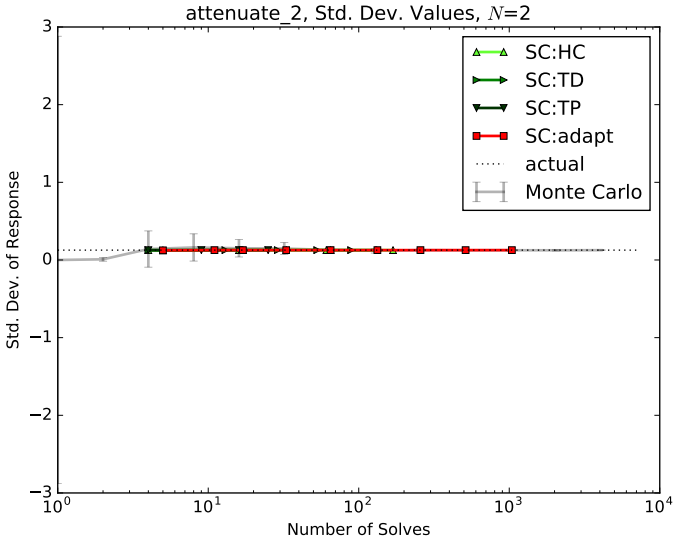

FIGURE 4.26: Attenuation, $N = 2$, Std. Dev. Values

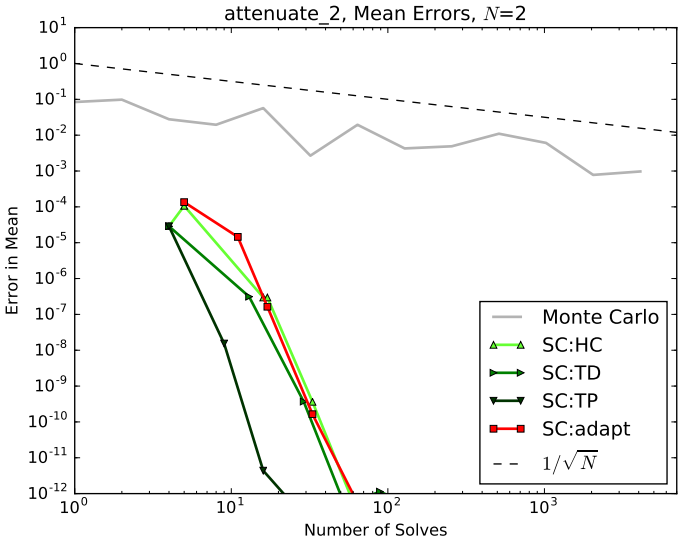

FIGURE 4.27: Attenuation, $N = 2$, Mean Convergence

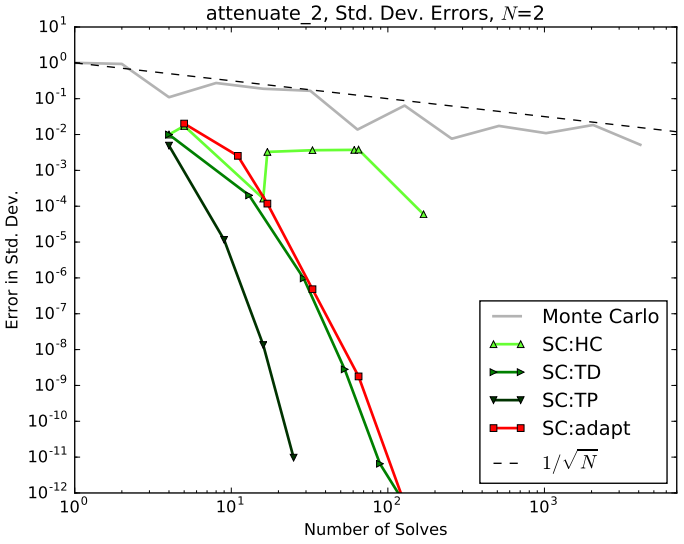

FIGURE 4.28: Attenuation, $N = 2$, Std. Dev. Convergence

4.4.4 Attenuation: 4 Inputs

As with the two-input case, all methods show good convergence on the mean, and only the Hyperbolic Cross polynomials show poor performance for the standard deviation. Interestingly, despite Tensor Product matching the construction shape of the model well, both Total Degree and Adaptive perform quite similar to TP and converge quickly as well.

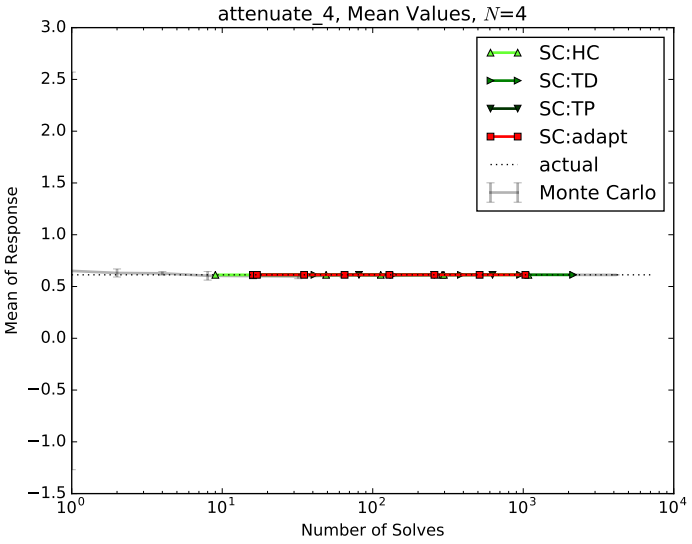

FIGURE 4.29: Attenuation, $N = 4$, Mean Values

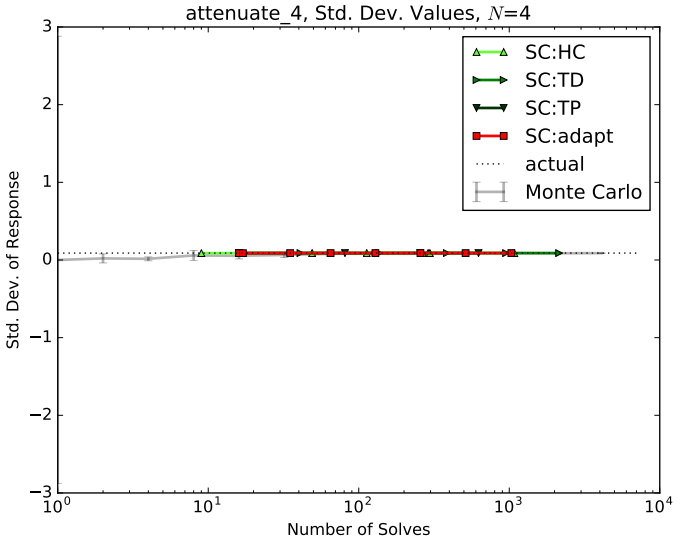

FIGURE 4.30: Attenuation, $N = 4$, Std. Dev. Values

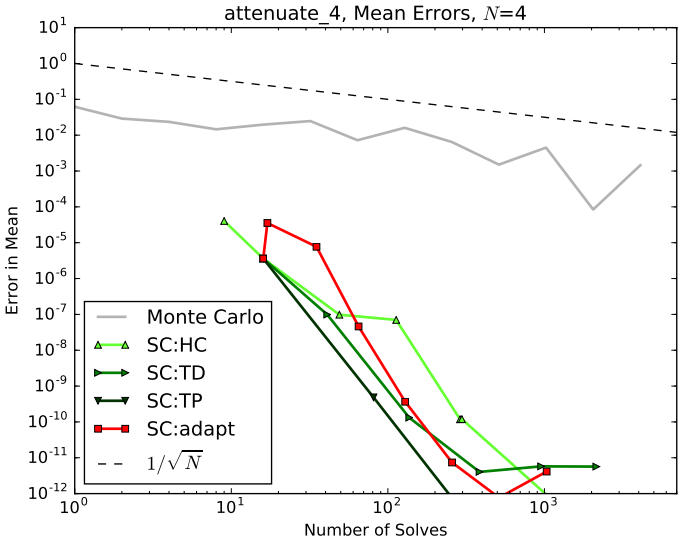

FIGURE 4.31: Attenuation, $N = 4$, Mean Convergence

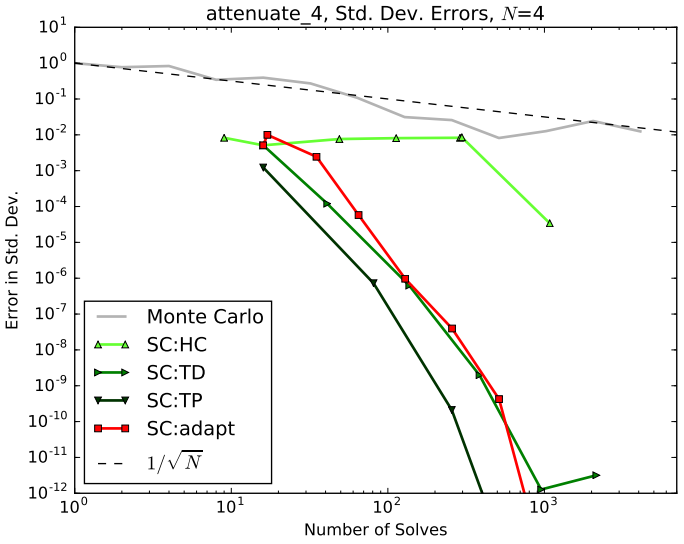

FIGURE 4.32: Attenuation, $N = 4$, Std. Dev. Convergence

4.4.5 Attenuation: 6 Inputs

The general trend in the two-input and four-input cases continues for six inputs, with one exception. For six inputs, the Adaptive method struggles to find the most suitable set of polynomials to include in the expansion. This is likely because of the large number of polynomial combinations available to consider with the larger input space. If there is any tendency to inaccurately guess the path to take, this misstep is likely to be taken many times before the more accurate path is discovered. Otherwise, exponential convergence is still observed, but with a larger radius of curvature than the lower-dimension cases.

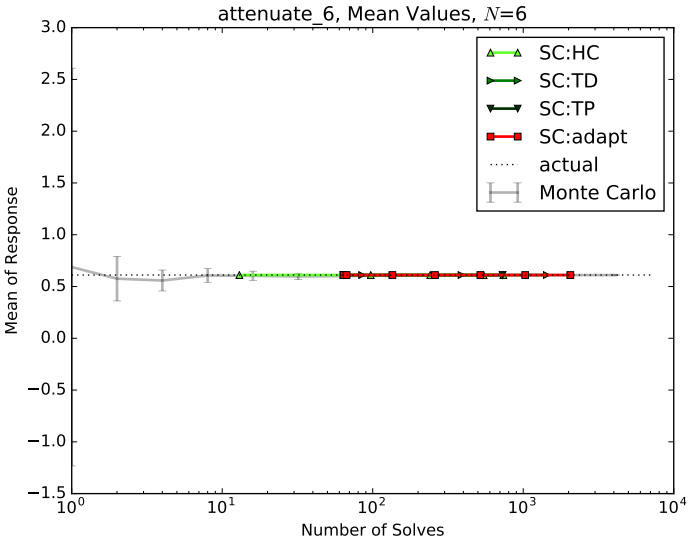

FIGURE 4.33: Attenuation, $N = 6$, Mean Values

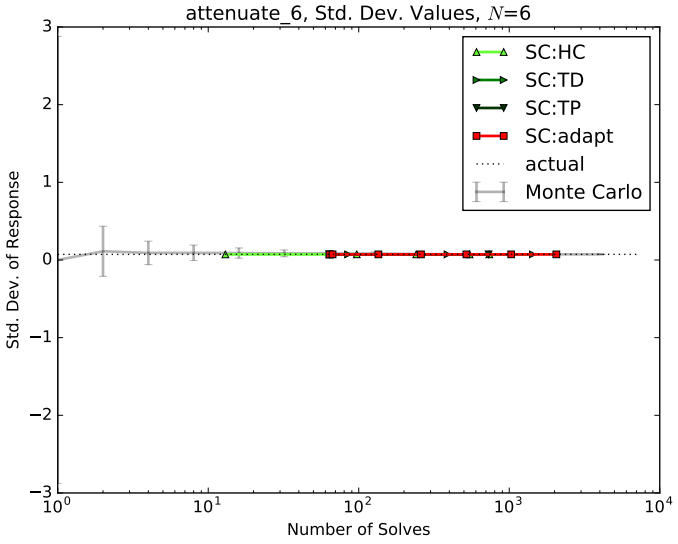

FIGURE 4.34: Attenuation, $N = 6$, Std. Dev. Values

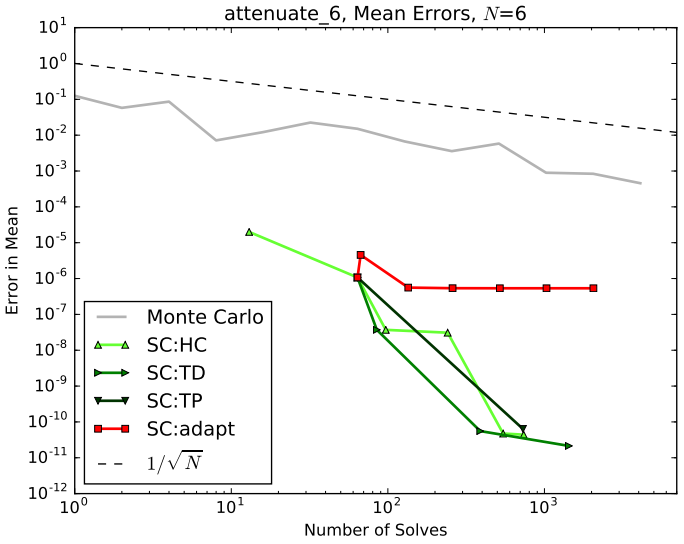

FIGURE 4.35: Attenuation, $N = 6$, Mean Convergence

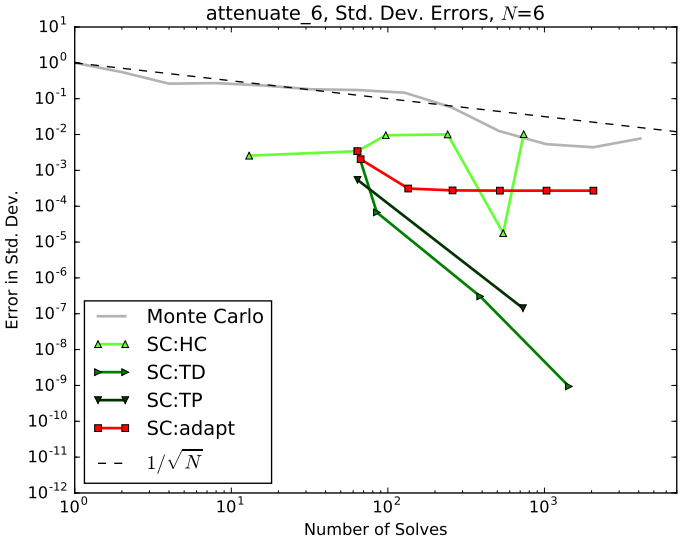

FIGURE 4.36: Attenuation, $N = 6$, Std. Dev. Convergence

4.5 Gauss Peak

4.5.1 Description

Similar to the attenuation model, the Gaussian peak [44] instead uses square arguments to the exponential function. A tuning parameter a can also 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

$$u(Y) = \exp\left(-\sum_{n=1}^{N} a^2(y_n - \mu)^2\right). \tag{4.9}$$

We allow each y_n to vary uniformly on [0,1] and set peakedness to $a = 3$, with the center of the peak at (0.5,0.5). The two-dimensional representation of this response is given in Figure 4.37. A summary of analytic statistics is given in Table 4.6.

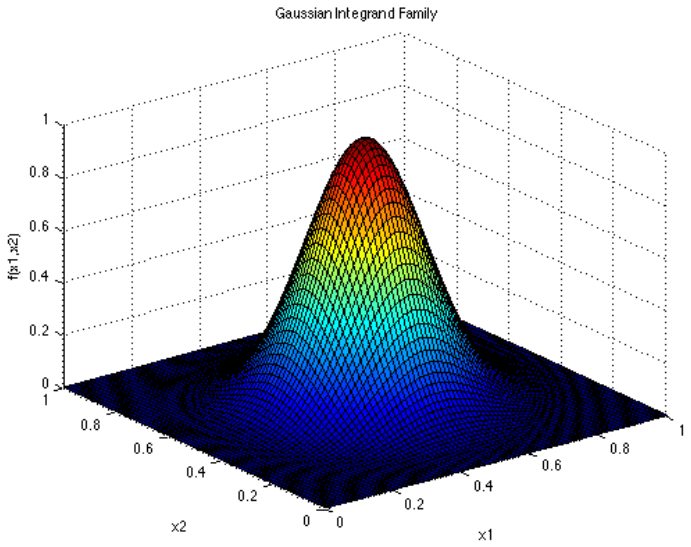

FIGURE 4.37: Gaussian Peak Response [3]

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

TABLE 4.6: Analytic Expressions for Gaussian Peak Case

4.5.2 Discussion

This case offers particular challenge because of its Taylor development, which only includes even powers of the uncertain parameters,

$$e^{-a^2y^2} = 1 - a^2y^2 + \frac{a^4}{2}y^4 - \frac{a^6}{6}y^6 + \frac{a^8}{24}y^8 + \mathcal{O}(y^10). \tag{4.10}$$

This suggests added difficulty in successive representation, especially for an adaptive algorithm. A visual demonstration of this is shown in Table 4.7. This Taylor development shows the same tendencies as the Attenuation model Taylor development; however, only

		Polynomial Order (y_1)				
		0	1	2	3	4
Polynomial Order (y_2)	0	1	0	a^2	0	$a^4/2$
	1	0	0	0	0	0
	2	a^2	0	a^4	0	$a^6/2$
	3	0	0	0	0	0
	4	$a^4/2$	0	$a^6/2$	0	$a^8/4$

TABLE 4.7: Coefficient Magnitudes, Tensor Taylor Development of $e^{-a^2y^2}$

even polynomials are present, and these drop off at a slower rate than the Attenuation model. Also, because a is greater than one for this model, it counteracts the polynomial coefficient dropoff seen in the expansion, whereas in the Attenuation model a was less than one and coefficients dropped off more quickly as a result. Due to these factors, we see poorer performance for the collocation methods on converging this model than the Attenuation model, despite their apparent similarities.

4.5.3 Gauss Peak: 3 Inputs

For this smaller input space, we see good exponential convergence on the mean for the Hyperbolic Cross, Total Degree, and Tensor Product index sets. However, the adaptive method fails entirely. This is because none of the first-order polynomials have any contribution even when integrated coarsely; as a result, the adaptive algorithm is duped into believing it is converged. This same behavior is seen for the five-input case as well. As expected, the standard deviation shows poorer performance for all three methods than the mean; in fact, only the Tensor Product is clearly converging exponentially for the standard deviation even with only three inputs. This demonstrates the challenge of this model to be represented well with low-order polynomials.

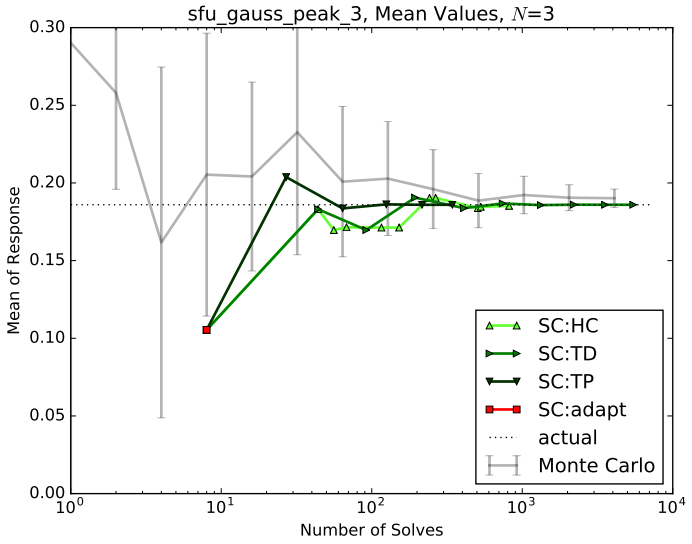

FIGURE 4.38: Gauss Peak, $N = 3$, Mean Values

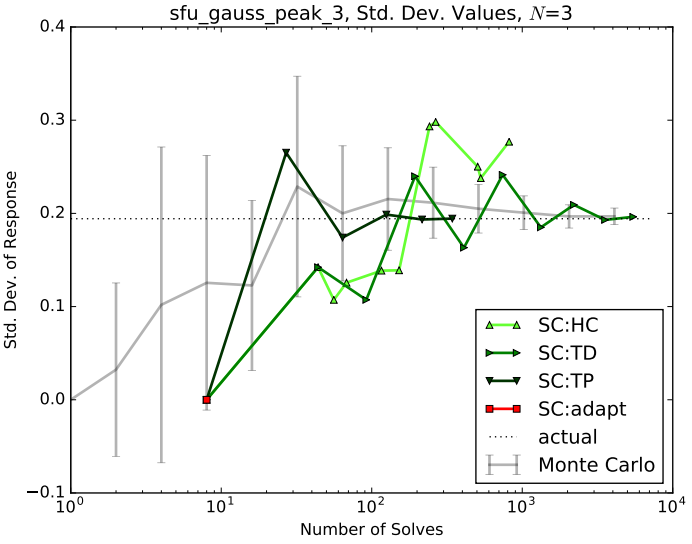

FIGURE 4.39: Gauss Peak, $N = 3$, Std. Dev. Values

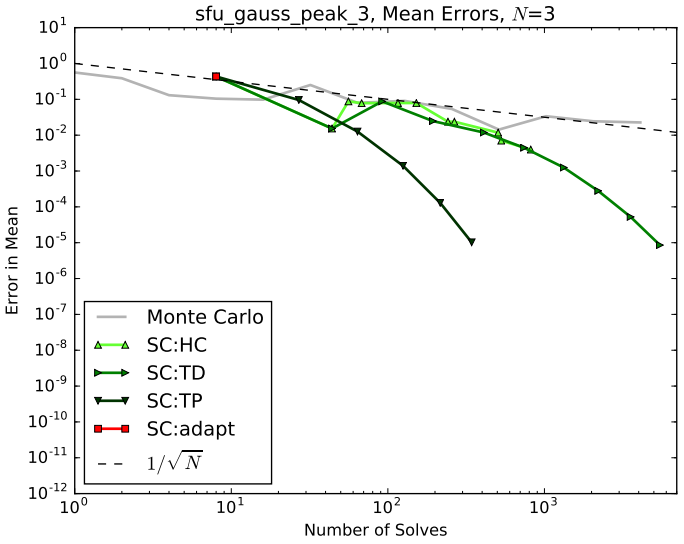

FIGURE 4.40: Gauss Peak, $N = 3$, Mean Convergence

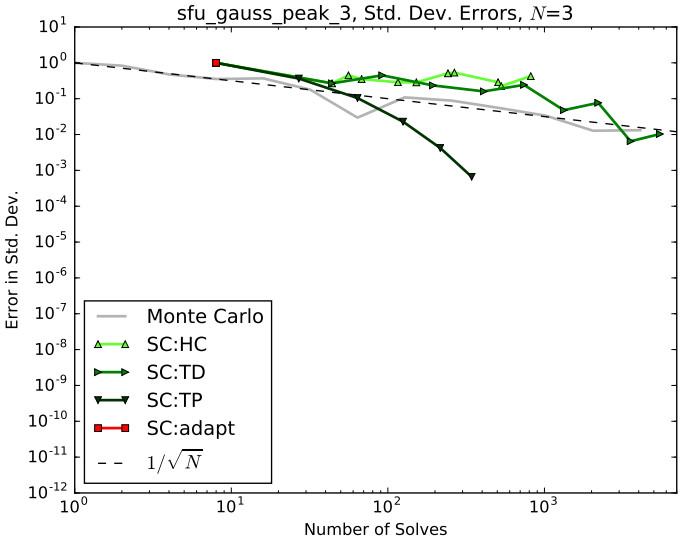

FIGURE 4.41: Gauss Peak, $N = 3$, Std. Dev. Convergence

4.5.4 Gauss Peak: 5 Inputs

The same trends are observed for five inputs as for three, but with poorer convergence in all methods. While it appears there is some exponential convergence benefits in the collocation methods, for up to 1000 computation solves there is little advantage over MC.

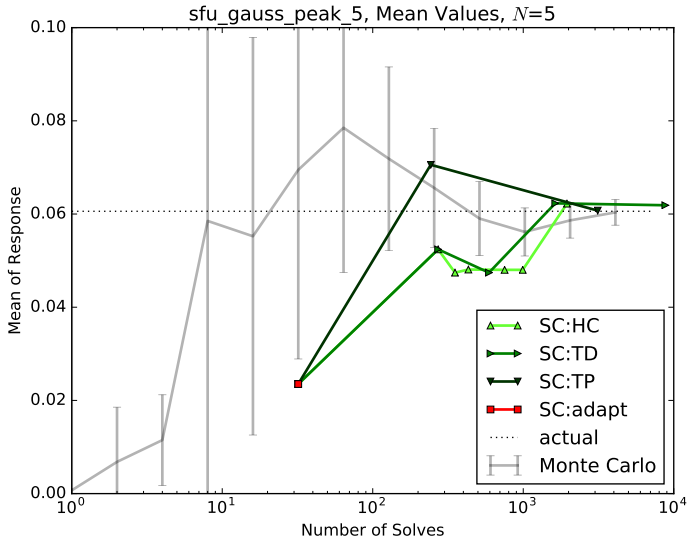

FIGURE 4.42: Gauss Peak, $N = 5$, Mean Values

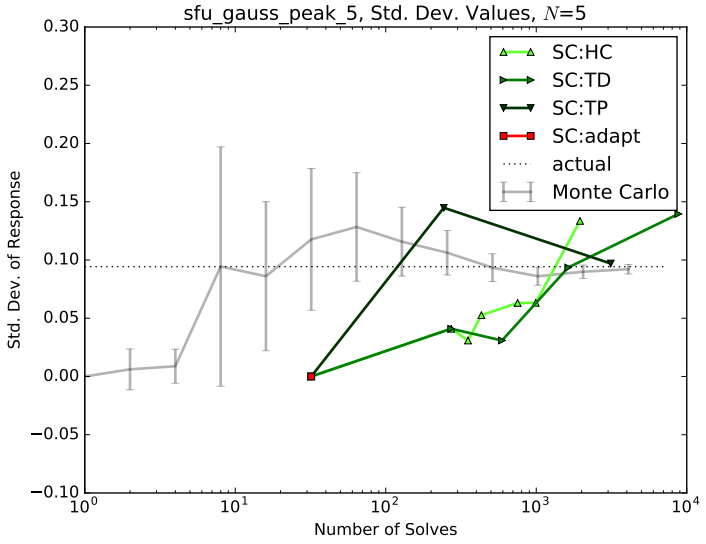

FIGURE 4.43: Gauss Peak, $N = 5$, Std. Dev. Values

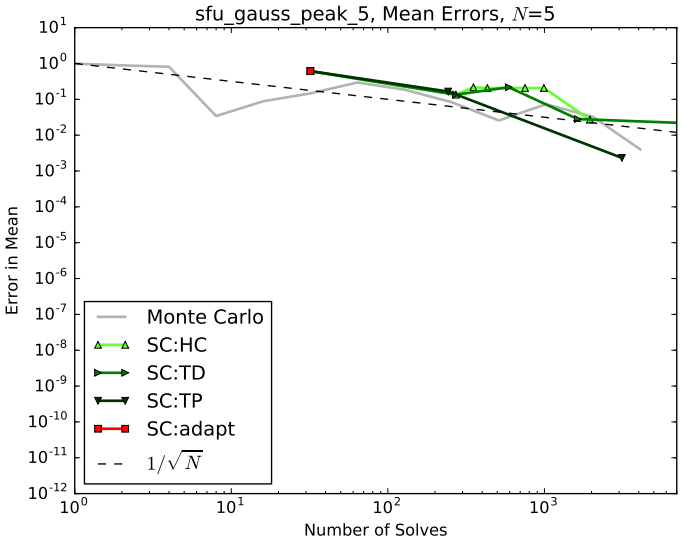

FIGURE 4.44: Gauss Peak, $N = 5$, Mean Convergence

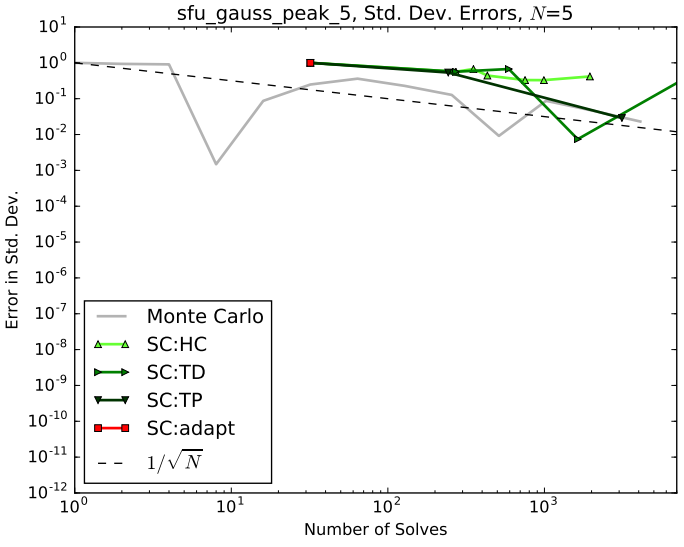

FIGURE 4.45: Gauss Peak, $N = 5$, Std. Dev. Convergence

4.6 Ishigami

4.6.1 Description

The Ishigami function [45] is a commonly-used function in performing sensitivity analysis. It is given by

$$u(Y) = \sin y_1 + a \sin^2 y_2 + b y_3^4 \sin(y_1). \tag{4.11}$$

In our case, we will use $a = 7$ and $b = 0.1$ as in [46]. The graphical representation of this response is given in Figure 4.46, with the three axes as the three inputs and the color map as the function values ranging approximately from -10.74 to 17.74. In particular

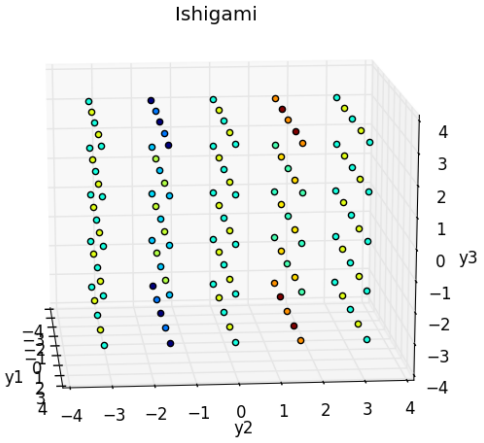

FIGURE 4.46: Ishigami Model Response

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 4.8, 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 4.8: Analytic Expressions for Ishigami Case

4.6.2 Discussion

The Ishigami function is sinusoidal in y_1 and y_2. Because the sine function is exclusively odd, this presents a similar challenge as previous models to adaptive methods, at least for these two dimensions. y_3, however, only appears as a fourth-order coefficient to the sine of y_1, which makes for a relationship that is difficult for the polynomial representations

to capture. For this model, there is no flexibility in the dimensionality of the input space; we show the only case ($N = 3$) here.

4.6.3 Ishigami: 3 Inputs

For the mean we see good convergence for the three static methods, and surprisingly good convergence for the Hyperbolic Cross polynomials. Because the two dominant parameters are largely independent, the polar focus of the Hyperbolic Cross set captures the essential components with less computation than the other two static methods. The adaptive method, as predicted, struggles to find any important polynomials before finding false convergence.

For the standard deviation, however, we see significant divergence for both the Hyperbolic Cross and Adaptive methods. Despite some oscillations, however, we do see exponential convergence for both the Tensor Product and Total Degree methods. Despite this, marked improvements over MC are not distinct until after 100 computational solves, despite the small input space.

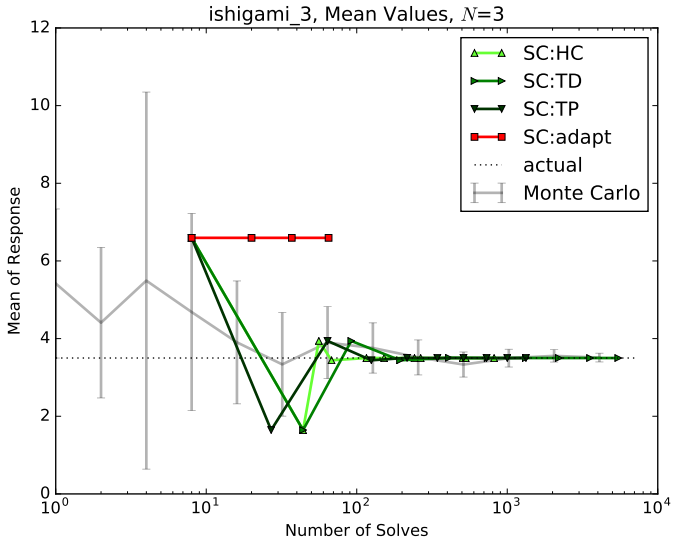

FIGURE 4.47: Ishigami, $N = 3$, Mean Values

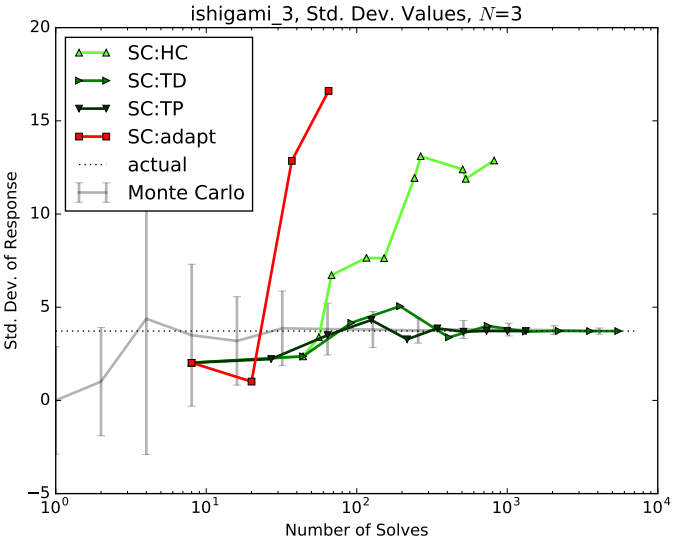

FIGURE 4.48: Ishigami, $N = 3$, Std. Dev. Values

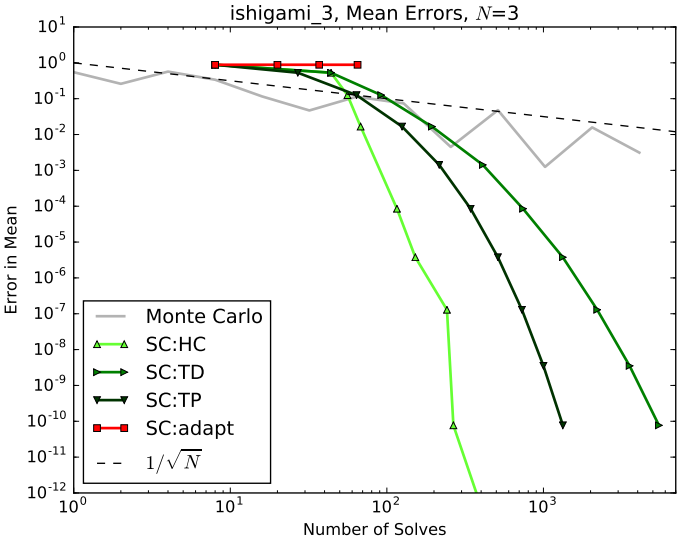

FIGURE 4.49: Ishigami, $N = 3$, Mean Convergence

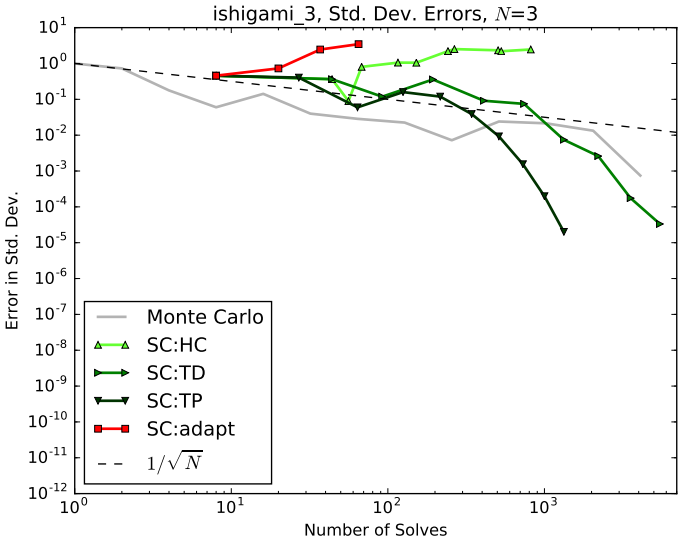

FIGURE 4.50: Ishigami, $N = 3$, Std. Dev. Convergence

4.7 Sobol G-Function

4.7.1 Description

The so-called "g-function" introduced by Saltelli and Sobol [48] is a discontinuous function used most commonly as a test for sensitivity coefficients. The function is often used as an integrand for numerical estimation methods [49]. The function is given by

$$u(Y) = \prod_{n=1}^{N} \frac{|4y_n - 2| - a_n}{1 + a_n}, \tag{4.12}$$

where

$$a_n = \frac{n-2}{2}. \tag{4.13}$$

The two-dimensional representation of this response is given in Figure 4.51.

There are some implementations [49] that force $a_n \geq 0$, which allows for a simple understanding of the sensitivity coefficients:

- $a_n = 0$: y_n is very important
- $a_n = 1$: y_n is relatively important,
- $a_n = 9$: y_n is non-important,
- $a_n = 99$: y_n is non-significant.

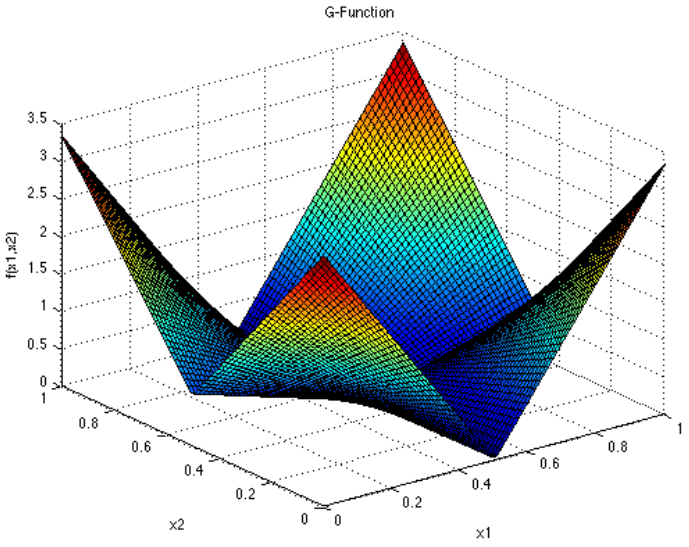

FIGURE 4.51: Sobol G-Function Response [3]

However, for our purposes, we set no limit to the value of a_n, as our interest is primarily in the moments instead of the sensitivity coefficients.

We select this model because it offers the challenge of a function without a continuous first derivative. We expect the polynomial representations to perform poorly in this instance, and more so as dimensionality increases.

4.7.2 Discussion

As expected, even for 3 input variables this discontinuous model provides a difficult challenge for polynomial representations. Despite thousands of computational solves, there is no discernible benefit in using collocation methods over traditional MC. As with the Gaussian peak, the adaptive method completely stalls in trying to converge this model.

4.7.3 Sobol G-Function: 3 Inputs

As evidenced in the figures, there is little justification for using any of the collocation methods for uncertainty quantification with this model. The level of discontinuity renders the benefits of the polynomial expansions moot.

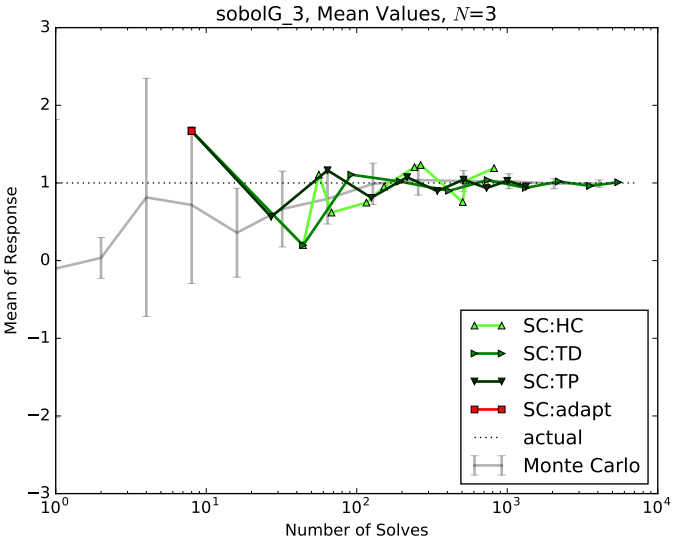

FIGURE 4.52: Sobol G-Function, $N = 3$, Mean Values

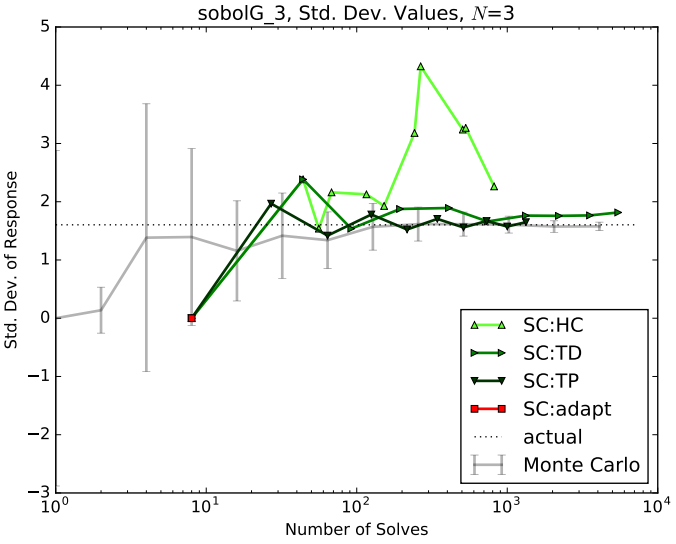

FIGURE 4.53: Sobol G-Function, $N = 3$, Std. Dev. Values

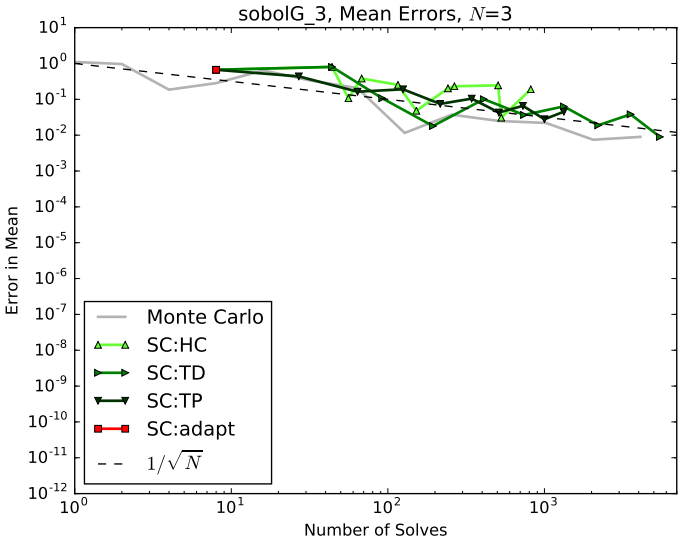

FIGURE 4.54: Sobol G-Function, $N = 3$, Mean Convergence

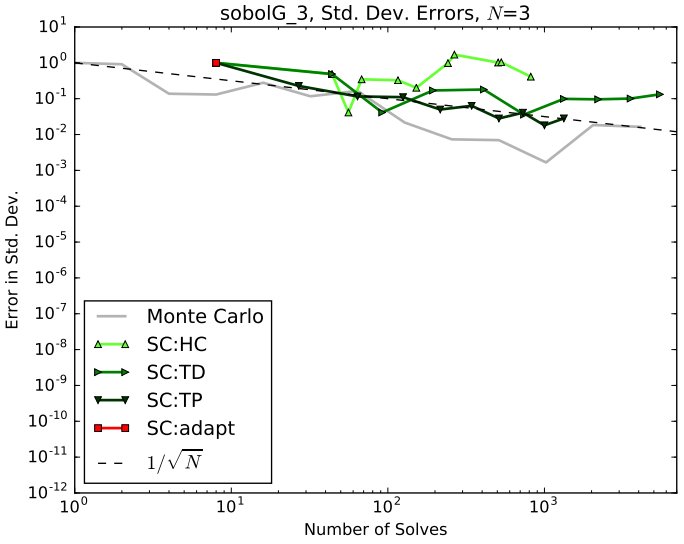

FIGURE 4.55: Sobol G-Function, $N = 3$, Std. Dev. Convergence

4.7.4 Sobol G-Function: 5 Inputs

As with the smaller input space, it is clear that discontinuous models such as this are poor candidates for collocation-based uncertainty analysis.

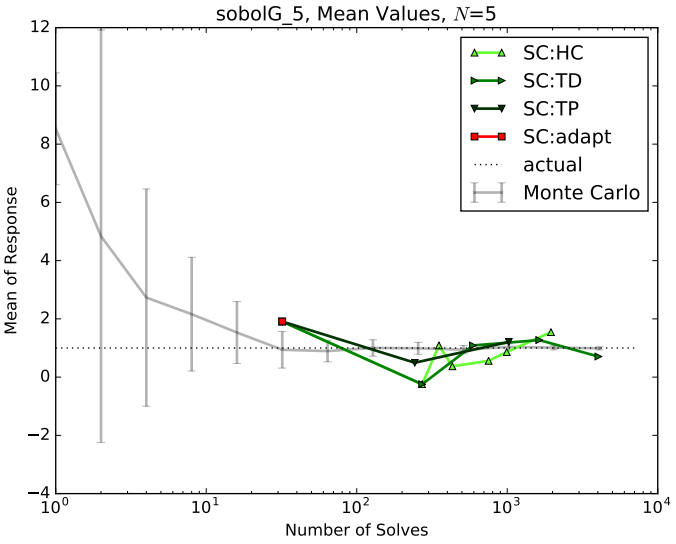

FIGURE 4.56: Sobol G-Function, $N = 5$, Mean Values

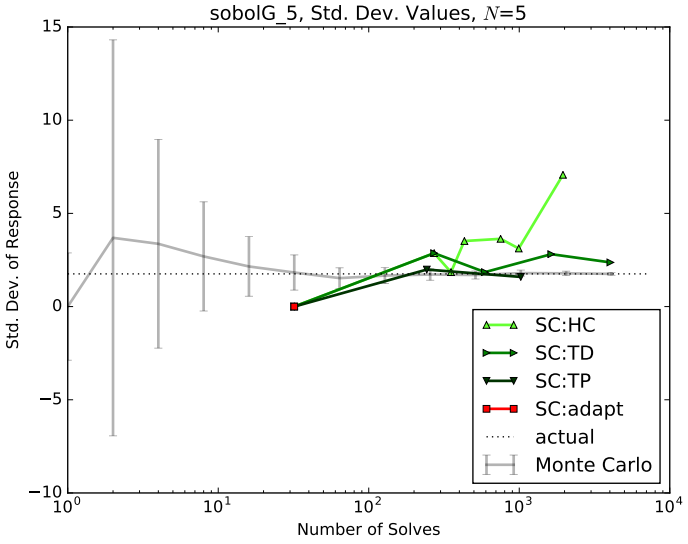

FIGURE 4.57: Sobol G-Function, $N = 5$, Std. Dev. Values

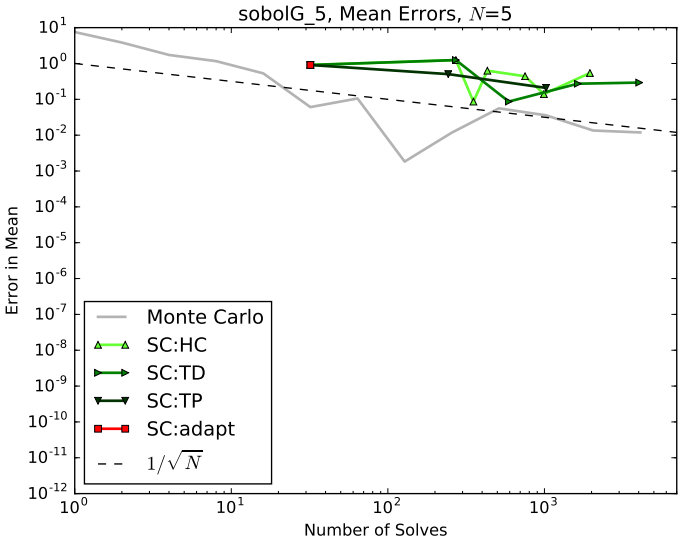

FIGURE 4.58: Sobol G-Function, $N = 5$, Mean Convergence

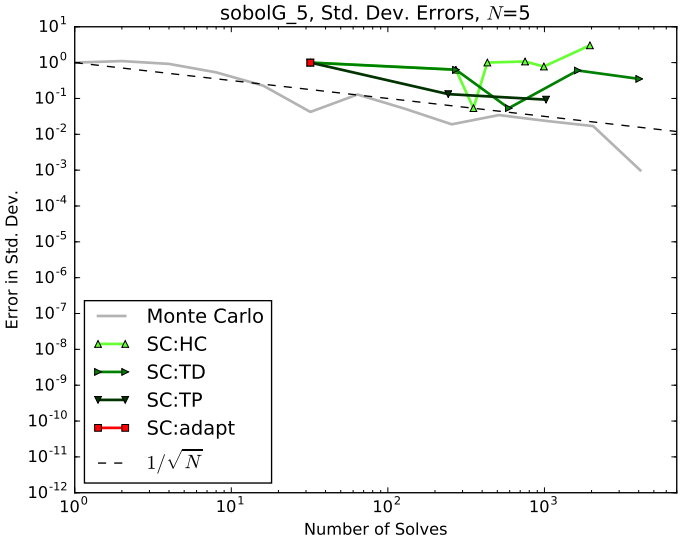

FIGURE 4.59: Sobol G-Function, $N = 5$, Std. Dev. Convergence

4.8 Conclusions

We have demonstrated the performance of SCgPC using a variety of polynomial set construction techniques described in Chapter 3: tensor product, total degree, and hyperbolic cross, as well as adaptive. There are several conclusions that can be drawn from these models.

First, in all cases the ability of collocation-based methods to be efficient in converging second-order statistics was reduced in direct correlation with the dimensionality of the

input space. For an input space of five or less variables, in general the collocation methods performed well, while for more than five variables, performance was degraded significantly.

Second, as a model demonstrates less regularity, the performance of SCgPC methods in comparison to MC degrades. In the case of the Sobol G-Function, which is only zeroth-order continuous, SCgPC fails to offer any benefits over MC. However, in models with more smoothness, in general SCgPC converged exponentially on the response statistics.

The combination of dimensionality and smoothness can be seen through the several models presented. The Tensor Monomials and Attenuation models demonstrate performance of collocation methods on tensor-construction models with polynomial representations whose coefficient magnitudes are monotonically decreasing; that is, there are no polynomial order coefficients which are zero, and each polynomial coefficient of a higher order is smaller than one of a lower order. In these conditions, all the collocation methods performed very well for reasonably-sized input spaces.

The Sudret Polynomial and Gauss Peak models represent functions in which there are "missing" polynomial orders; that is, there are some polynomial orders whose coefficients are zero, while higher-order polynomials have nonzero coefficients. These presented a greater challenge to the SCgPC methods, especially the adaptive method. Because the adaptive method uses lower-order coefficients to predict higher-order impacts, these missing polynomials cause poor performance in the search algorithm. In particular for the Gauss Peak model, the high-order polynomials drop off slowly in importance, making polynomial representation costly.

The Ishigami function demonstrates performance when there is an irregular relationship between the input variables and the response. While two of the static methods (total degree and tensor product) demonstrated good exponential convergence, the adaptive method was not convergent. Finally, in the Sobol G-Function, we saw that even first-order tensor polynomials, when made zeroth-order continuous, are very difficult for SCgPC methods to converge.

In conclusion, the SCgPC methods excel when the input space is of small dimensionality and the response is regular with respect to the input space. The adaptive method performs well when the polynomial representation of the response has monotonically decreasing reliance on increasing polynomial orders. SCgPC methods also perform especially well when the model is represented well by relatively low-order polynomials. SCgPC methods should not be used on discontinuous responses or models with large input space dimensionality, as MC is a more practical tool to converge second-order statistics for such models.

Chapter 5

High-Dimension Model Representation Methods

5.1 Introduction

As demonstrated in Chapter 4, the Stochastic Collocation for generalized Polynomial Chaos (SCgPC) methods for uncertainty quantification defined in Chapter 3 can be very efficient tools in comparison with traditional Monte Carlo to converge second-order statistics. In particular, SCgPC methods excel when the input dimensionality of a model is low and the response of interest is regular with respect to the input space. Conversely, they perform poorly with discontinuous responses and high dimensionality input spaces.

Another useful model reduction method is High-Dimension Model Representation [34], which is based on Sobol decomposition and is an ANalysis Of VAriance (ANOVA) method. ANOVA is a class of methodologies in statistics that seeks to determine the source of variances in a response when considering the input space. ANOVA methods originate in sampling statistics, where the structure of a collection of samples is desired [65]. The HDMR expansion helps mitigate the problem of large input space dimensionality by dividing the model into a linear superposition of terms, each of which only depends on a subset of the full input space. The subsets are developed by integrating out the undesired dimensions.

Let $H(Y)$ represent the HDMR expansion of $u(Y)$,

$$u(Y) = H(Y) = h_0 + \sum_{n=1}^{N} h_n + \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} h_{n_1,n_2} + \cdots + h_{1,2,\ldots,N}, \tag{5.1}$$

The h_0, the expected value, is given by

$$h_0 \equiv \int_{\Omega_1} \dots \int_{\Omega_N} u(y_1, \dots, y_N)\ dy_1 \dots\ dy_N, \tag{5.2}$$

$$= \int_{\Omega} u(Y) dY, \tag{5.3}$$

where Ω denotes the uncertainty space spanned by Y. Recall also the definition of integration over the input space Ω given in Eq. 2.3. The first-order expansion terms h_n are integrated as

$$h_n(y_n) \equiv \int_{\hat{\Omega}_n} u(Y)\ d\hat{Y}_n - h_0, \tag{5.4}$$

where we use "hat" notation to refer to all elements except the one listed; for example,

$$\hat{Y}_n \equiv (y_1, \cdots, y_{n-1}, y_{n+1}, \cdots, y_N), \tag{5.5}$$

$$\hat{Y}_{m,n} \equiv (y_1, \cdots, y_{m-1}, y_{m+1}, \cdots, y_{n-1}, y_{n+1}, \cdots, y_N). \tag{5.6}$$

Similarly, $\hat{\Omega}$ is the slice of the input space only containing variables in $\hat{Y}$ and integration over $\hat{\Omega}$ is weighted with respect to $\rho(\hat{Y})$. Second and higher-order HDMR expansion terms are defined as

$$h_{n_1,n_2}(y_{n_1}, y_{n_2})) \equiv \int_{\hat{\Omega}_{n_1,n_2}} \hat{\rho}_{n_1,n_2}(\hat{Y}_{n_1,n_2}) u(Y)\ d\hat{Y}_{n_1,n_2} - h_{n_1} - h_{n_2} - h_0, \tag{5.7}$$

and so on. Written another way, each subset is constructed by integrating over the domain with respect to all input parameters the subset is not dependent on, then subtracting all other expansion subsets who depend on an input space that is a subspace of the space on which the subset depends, including the expected value.

$$h_{\mathbf{s}} = \int_{\hat{\Omega}_{\mathbf{s}}} u(Y) d\hat{Y} - \sum_{\substack{s \subset \mathbf{s} \\ |s|_1 < |\mathbf{s}|_1}} h_s, \tag{5.8}$$

where $\mathbf{s}$ is a vector of dimensional ordinates with length less than or equal to the dimensionality N of the input space.

There are many useful properties of this generic HDMR expansion. First, each term represents the contribution of that subset to the original response; that is, h_1 provides the contributions to the response solely from variable y_1. Further, the total contribution of a variable is the sum of all subsets for whom variable is part of the subspace. For example, the total contribution of y_1 to the response is the sum of contributions from $h_1, h_{1,2}, \cdots, h_{1,2,3}$, etc.

Second, full HDMR can easily be approximated by truncating terms from the expansion.

In particular, the expansion can be limited to interactions between a finite number of variables. Experience has shown [17] that most of the high-order interaction terms are negligible. The resulting truncated expansion contains terms that are much lower-order than the original model, often without incurring significant truncation error. The full expansion requires many integrals to construct, often making it inefficient when compared with using the original model. Because the full expansion takes significant work to construct, we assume some level of truncation is performed when using HDMR.

Third, the individual terms in the HDMR expansion are orthogonal with respect to the probability weight over the input space; that is,

$$\int_{\Omega} h_a h_b dY = 0 \quad \forall \ a \neq b. \tag{5.9}$$

Because of this, the second statistical moment of the HDMR expansion with respect to any subset dimension is the equivalent to the second statistical moment of the associated subset,

$$\int_{\Omega_n} H(Y)^2 dy_n = \int_{\Omega_n} h_n^2 dy_n. \tag{5.10}$$

This in turn directly yields Sobol sensitivity coefficients. Sobol sensitivity coefficients measure the impact on the variance of a response as the result of changes in the variance of an input (or combination of inputs). For the HDMR expansion,

$$\mathcal{S}_n \equiv \frac{\text{var}[h_n]}{\text{var}[H(Y)]}, \tag{5.11}$$

$$\mathcal{S}_{m,n} \equiv \frac{\text{var}[h_{m,n}]}{\text{var}[H(Y)]}, \tag{5.12}$$

and so on.

5.2 Cut-HDMR

The primary challenge in implementing HDMR for arbitrary responses is the integrals in Eq. 5.2, 5.4, and 5.7. These integrals are of as high dimensionality as those required for SCgPC. As a result, at first glance HDMR seems to offer no benefits. However, we make use of an approximation for HDMR called *cut-HDMR* [16] that makes a simplifying assumption. In cut-HDMR, we assume the integral of a function over a dimension can be approximated by evaluating the function at a set of reference values $\bar{Y} = (\bar{y}_1, \bar{y}_2, \cdots, \bar{y}_N)$, where bar notation indicates the expected value of each input variable. The reference value in this case is a single point in the input space, often the mean of the input multidimensional probability distribution. The reference point, as well as planes and

hyperplanes passing through the reference point, make up the *cuts* that give this method its name. The cut-HDMR expansion $T(Y)$ of model $u(Y)$ is expressed as

$$u(Y) = T(Y) = t_r + \sum_{n=1}^{N} t_n + \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} t_{n_1,n_2} + \cdots + t_{1,2,\ldots,N}. \tag{5.13}$$

Eq. 5.18 is identical in form to the traditional ANOVA HDMR expansion in Eq. 5.1, but the subset components are defined differently,

$$t_r \equiv u(\bar{Y}), \tag{5.14}$$

$$t_n(y_n) \equiv u(y_n, \hat{\bar{Y}}_n) - t_r, \tag{5.15}$$

$$t_{m,n}(y_m, y_n) \equiv u(y_m, y_n, \hat{\bar{Y}}_{m,n}) - t_m - t_n - t_r, \tag{5.16}$$

and so on. Note that $\bar{Y}$ is the reference input realization, and $\hat{\bar{Y}}_n$ denotes a partial input realization where all inputs are at reference values and y_n is excluded:

$$\hat{\bar{Y}}_n = (\bar{y}_1, \cdots, \bar{y}_{n-1}, \bar{y}_{n+1}, \cdots, \bar{y}_N). \tag{5.17}$$

In the limit where each subset of cut-HDMR is at most linearly dependent on an input parameter, cut-HDMR and ANOVA are identical and exact. Additionally, both ANOVA and cut-HDMR converge exactly on the model if no truncation is performed.

The immediate benefit from cut-HDMR is the ability to computationally calculate the terms in Eq. 5.18; we only need the reference input realization $\bar{Y}$ to construct the expansion. However, there are two drawbacks to this expansion. First, cut-HDMR approximates the expected value of a model as the value of the model evaluated at the input parameters' expected value. While higher-order expansion terms correct this assumption, for low-order truncations it can have significant impact. Second, cut-HDMR component terms are not orthogonal, unlike standard HDMR (hereafter referred to as ANOVA to avoid confusion). This results in difficulty when attempting to algorithmically determine statistical moments. Fortunately, this will be resolved in section B.1. First, however, we consider how to represent the subset terms in the cut-HDMR expansion in an algorithmically-efficient manner.

5.3 gPC and cut-HDMR

Consider the cut-HDMR expansion,

$$u(Y) = T(Y) = t_r + \sum_{n=1}^{N} t_n + \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} t_{n_1,n_2} + \cdots + t_{1,2,\ldots,N}, \tag{5.18}$$

with subsets t defined in section 5.2. Each subset besides the reference solution t_r is a function of at least one uncertain input; for example, $t_1(y_1)$ and $t_{1,3,7}(y_1, y_3, y_7)$. We can consider each of these an independent uncertain model, with many of the same features as the entire model $u(Y)$. These subset terms have their own mean, variance, and sensitivities. Additionally these subsets can each be represented by a SCgPC expansion,

$$t_n \approx \sum_{k' \in \Lambda'(L')} t_{n;k'} \Phi_{k'}(Y_n), \tag{5.19}$$

where we make use of prime notation k', Λ', L' to denote a SCgPC expansion for a subset term of a cut-HDMR expansion and $t_{n;k'}$ are the scalar expansion coefficients. Eq. 5.18 can then be written

$$\begin{aligned} T(Y) \approx t_r &+ \sum_{n=1}^{N} \left(\sum_{k' \in \Lambda'_n(L')} t_{n;k'} \Phi_{k'}(Y_n) \right) \\ &+ \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} \left(\sum_{k' \in \Lambda'_{m,n}(L')} t_{m,n;k'} \Phi_{k'}(Y_m, Y_n) \right) \\ &+ \cdots \\ &+ \left(\sum_{k' \in \Lambda'_{1,\cdots,N}(L')} t_{1,\cdots,N;k'} \Phi_{k'}(Y_1, \cdots, Y_N) \right). \end{aligned} \tag{5.20}$$

There are several synergies make available by using SCgPC to expand the subset terms in the cut-HDMR expansion as in Eq. 5.20. First, the scalar expansion coefficients can be calculated using the same collocation-based methods developed for the SCgPC method. As we demonstrate in chapter 4, these collocation methods are most efficient when the dimensionality is low and the response is smooth. Because we expect the cut-HDMR expansion to be truncated at some finite level, consider the progression of the terms retained in Eq. 5.18. The first term has zero dimensionality, the next set of terms all have dimensionality of one, the next set two, and so forth. All of the terms kept in cut-HDMR expansions truncated to third-level interactions are dimensionality three or smaller, which is ideal size for exceedingly efficient convergence of SCgPC methods.

In addition, SCgPC methods are most efficient when the response is regular, or has a high degree of continuity. Whatever the continuity of the original model, the continuity of the subsets in the HDMR expansion of that model are at least as continuous, and can often be more continuous. This is because the subsets are obtained by removing the dependence on some of the constituent variables. If any discontinuity in the original response is contributed by any of those variables, the resulting continuity is greater for the subset. Since cut-HDMR naturally divides up the subset space, it will converge the smooth subsets rapidly, possibly converging on the original model more efficiently than purely SCgPC can without using cut-HDMR for somewhat discontinuous responses.

Second, SCgPC polynomials are constructed to be inherently orthonormal. As long as consistency is maintained in the polynomial families between different cut-HDMR subsets, this orthonormality extends into interactions between subsets. We explore using this advantage to reconstruct ANOVA terms in Appendix B.

5.4 On convergence of gPC and cut-HDMR with gPC

We pause momentarily to make a note about the convergence of SCgPC methods alone in contrast to using SCgPC as part of a cut-HDMR expansion. There are two degrees of freedom in defining static SCgPC expansion construction. The first is the polynomial construction strategy, such as hyperbolic cross, total degree, or tensor product, along with level of anisotropy. The second is the polynomial order limit L. For cut-HDMR, however, in addition to the polynomial set and level, we add another option: the HDMR truncation level. The HDMR truncation level determines the maximum dimensionality of any subset in the cut-HDMR expansion. Equivalently, it determines the order of interactions between variables to include in the expansion. For instance, second-order HDMR truncation limits the expansion to at most pairwise interactions.

Consider a cut-HDMR expansion that uses isotropic total degree polynomial index set construction strategy with a limiting total polynomial order of L for its subset gPC terms, and a comparable pure SCgPC expansion with the same isotropic total degree polynomial index set construction strategy and same limiting total polynomial order L. In this situation, cut-HDMR *without truncation* is equivalent to the SCgPC expansion. Any truncation of the cut-HDMR yields an approximation to the pure generalized polynomial expansion, and as mentioned in section 5.1, an untruncated HDMR expansion is by nature inefficient. As a result, for a given polynomial order limit L, cut-HDMR can at most match the convergence of the corresponding generalized polynomial chaos expansion. Usually, it will be less accurate than the SCgPC equivalent because of HDMR truncation error. Additionally, the full cut-HDMR expansion will use a very similar

number of numerical evaluations to obtain an equal level of convergence. This means there is no improved efficiency for cut-HDMR over SCgPC.

However, the real benefit of cut-HDMR is seen in models with large input dimensionality. In this case, even a first-order SCgPC expansion using total degree index set construction could take thousands of evaluations to construct. Because cut-HDMR can be truncated to limited interactions, however, for far fewer evaluations, cut-HDMR can be constructed. For models that are computationally expensive and thousands of solves are prohibitive, the error incurred by truncating cut-HDMR may be worth the reduction in necessary evaluations. Even though cut-HDMR might be less efficient, it may still be constructed when the SCgPC cannot.

5.5 Reconstructing ANOVA from cut-HDMR

In general, it is not straightforward to calculate second-moment statistics of a cut-HDMR expansion, as the subset terms are not orthogonal. This means evaluating the integral of the product of every pair combination of subsets in the expansion, which almost surely cannot be done analytically. When using gPC to represent individual cut-HDMR subsets however, it is simple to analytically recover ANOVA statistics for a cut-HDMR expansion, despite the lack of orthogonality in cut-HDMR terms. This is because the gPC components of each subset term are replete with orthogonal relationships. Note that while this method will obtain ANOVA results for cut-HDMR terms, the statistics gathered are for the cut-HDMR expansion, not for the original model. If the cut-HDMR expansion is truncated as expected, the ANOVA terms will only be as accurate to the original model as the cut-HDMR expansion itself is.

Ultimately, because of orthogonality, the variance contribution for each subset is the sum of the squares of all polynomial coefficients whose associated polynomials are at least first-order polynomials with respect to all of the subset's input space, and only the subset's input space. The total variance is the sum of each subset's variance contribution.

More discussion and an example of this process is provided in Appendix B.

5.6 Adaptive HDMR

As discussed regarding the adaptive SCgPC method in section 3.5, it is not only possible but likely that different input variables have different effective polynomial order impact on a response. When constructing an HDMR expansion, we similarly expect that often some subsets in the expansion will require fewer polynomials to represent well than

others. Often, however, an analyst cannot know a priori to what order input subsets should be expanded to accurately represent a response This is especially true when working with abstract inputs such as those provided through a Karhunen-Leove expansion [41]. As with the adaptive SCgPC, it is convenient to have an algorithmic adaptive HDMR expansion construction strategy. Such an algorithm was proposed by Gerstner and Griebel [30] and demonstrated by Ayres and Eaton [10]. The algorithm is used to determine which subsets in the HDMR expansion should be included depending on what computational resources are available, in an adaptive manner. We extend existing methodology to include predictive algorithms for choosing forward directions. Additionally, we consider an intermingled adaptive approach of adaptive HDMR construction with adaptive sparse grid generalized polynomial chaos expansions for subsets.

5.6.1 Adaptive Algorithm

The existing algorithm [10, 30] is expanded by considering both adaptive SCgPC and adaptive HDMR acting simultaneously, as well as using predictive searching. The algorithm proceeds as follows:

1. Begin by constructing all the first-order HDMR subsets up to first-order polynomial expansions.
2. While not converged:
   (a) Iterate through each existing HDMR subset and determine the estimated impact of adding the next-favored polynomial for that subsets SCgPC expansion.
   (b) Determine the Sobol sensitivity coefficient for each HDMR subset using ANOVA (see Appendix B).
   (c) Predict the expected impact of adding each new eligible subset to the HDMR expansion.
   (d) Compare the product of a polynomial impact and its subset Sobol sensitivity with the expected impact of the eligible subsets.
   (e) Perform the most likely impactful operation (adding a new subset or adding a polynomial to an existing subset)
   (f) Use the estimated impact of eligible HDMR subsets and eligible polynomials for each subset to estimate total truncation error (total of all SCgPC truncation errors as well as HDMR truncation error).
   (g) Use previous iterations to approximate the convergence of the algorithm

(h) Combine estimated remaining variance with approximated convergence to determine convergence

(i) If convergence and estimated remaining variance are less than tolerance, convergence is reached.

(j) Otherwise, continue the algorithm.

This process is diagrammed in Figure 5.1. Note that this diagram assumes there is a sample submission process in a larger uncertainty quantification framework such as `RAVEN` [2], which handles computation resources in an efficient manner. In the flow chart, green indicates initialization, purple is the HDMR portion, blue is the SCgPC algorithms, yellow is the convergence process, and red indicates successful exit. Note that a path to the exit has been added for reaching some user-defined maximum number of runs; this is useful for allowing the algorithm to perform a search using a restricted amount of computational resources.

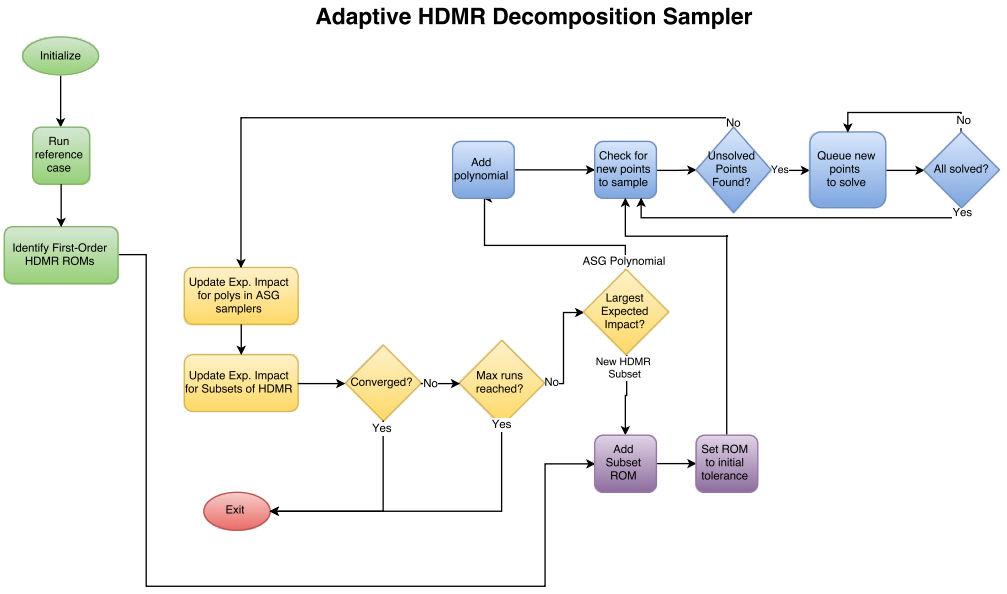

FIGURE 5.1: Adaptive HDMR with Adaptive Sparse Grid Flow Chart

Recall from section 3.5 that the impact of a polynomial within a subset generalized polynomial chaos expansion is given by Eq. 3.45,

$$\tilde{\eta}_k = \frac{1}{N-j} \sum_{n=1}^{N} \eta_{k-e_n}, \tag{5.21}$$

where we omit a superscript (y_n) to indicate the subset for which this polynomial is part of the SCgPC expansion. As we discuss in section 5.6.4, we restrict each polynomial to be eligible for addition to only one HDMR subset apiece, making the distinction unnecessary. The Sobol sensitivities provide the (current) impact of an existing subset,

$$S_\beta = \frac{\text{var}[h_\beta]}{\text{var}[T(Y)]}, \tag{5.22}$$

computing h as described in appendix section B.1, and introducing subset descriptor β which can be any number and combination of input variables y_n such that $\beta \subset Y$, or equivalently $\beta \subset (y_1, \cdots, y_N)$.

The estimated global impact $\tilde{\xi}_k$ of a polynomial k within a subset t_β is the product of its (estimated) local and (current) subset sensitivities,

$$\tilde{\xi}_k = \tilde{\eta}_k \cdot S_\beta. \tag{5.23}$$

Analogously, the actual global impact ξ_k of a polynomial k within a subset t_β is the product of its local and subset sensitivities,

$$\xi_k = \eta_k \cdot S_\beta, \tag{5.24}$$

with η_k given in Eq. 3.44. The estimated impact $\tilde{S}_\beta$ of adding a new subset to the HDMR expansion is the average of its dependent subsets' Sobol sensitivities,

$$\tilde{S}_\beta = \frac{1}{\dim(\beta)} \sum_{\substack{\gamma \subset \beta \\ 1 + \dim(\gamma) = m}} S_\gamma, \tag{5.25}$$

where by $\dim(\beta)$ we denote the dimensionality of β. For example,

$$\tilde{S}_{y_1, y_2} = \frac{1}{2}(S_{y_1} + S_{y_2}). \tag{5.26}$$

The philosophy behind combining polynomial impact parameters and Sobol sensitivity parameters is to provide a means to allow the adaptive algorithm to optimize computational resources. In effect, taking the product of the polynomial impact with the Sobol sensitivity provides a local-to-global contribution,

$$\eta_k \cdot S_\beta = \xi_k, \tag{5.27}$$

$$\frac{\text{local contribution}}{\text{local variance}} \cdot \frac{\text{local variance}}{\text{global variance}} = \frac{\text{local contribution}}{\text{global variance}}. \tag{5.28}$$

We consider momentarily the decision-making process of the adaptive algorithm. There are four possible comparisons the adaptive HDMR with adaptive SCgPC has to make: between polynomials within a HDMR subset, between polynomials in different HDMR subsets, between potential new HDMR subsets, and between polynomials and a potential new HDMR subset. In each case, the algorithm must decide which modification to existing expansions is most likely to contribute the largest to converging on second-order statistics.

Comparing polynomials within a subset is simple, and involves inquiring the truth of a statement like the following:

$$\tilde{\eta}_{k_1} \overset{?}{>} \tilde{\eta}_{k_2}. \tag{5.29}$$

Similarly, comparing new subsets is straightforward,

$$\tilde{S}_{\beta_1} \overset{?}{>} \tilde{S}_{\beta_2}. \tag{5.30}$$

Comparing polynomials from different subsets requires only weighting them by their subset Sobol sensitivities,

$$\tilde{\eta}_{k_1} S_{\beta_1} \overset{?}{>} \tilde{\eta}_{k_2} S_{\beta_2}. \tag{5.31}$$

Comparing polynomials to subsets is somewhat more ambiguous,

$$\tilde{\eta}_{k_1} S_{\beta_1} \overset{?}{>} \tilde{S}_{\beta_3}. \tag{5.32}$$

Three cases can occur in comparing subsets to polynomials:

- If $S_{\beta_1} = \tilde{S}_{\beta_3}$, because $\tilde{\eta}_{k_1}$ must be equal to or less than one, the algorithm will choose to add the new subset.
- If $S_{\beta_1} < \tilde{S}_{\beta_3}$, it is more reasonable to add a new subset before attempting to improve the resolution of the existing subset.
- If $S_{\beta_1} > \tilde{S}_{\beta_3}$, the determination is left up to the polynomial's sensitivity. If the polynomial is expected to have a low impact on the Sobol sensitivity coefficient of its subset, the algorithm will likely choose to add a new subset. If, however, the Sobol sensitivity coefficient is poorly converged, the algorithm will prefer to resolve it adding new subsets to the HDMR expansion.

5.6.2 Adaptive Search Preference Parameter

Because the predictive algorithm is somewhat arbitrary, we additionally offer a method to provide analysts a parameter to guide the adaptive process. By introducing a preferential factor $\alpha \in (0, 2)$, the user can push the algorithm to prefer either new subsets over polynomials or vice versa.

$$(\tilde{\eta}_{k_1})^{\alpha} S_{\beta_1}^{2-\alpha} \overset{?}{>} \left(\tilde{S}_{\beta_3}\right)^{2-\alpha}. \tag{5.33}$$

If α is zero, the polynomial impact is entirely ignored, and algorithm progression depends entirely on the Sobol sensitivity data. This means that even if a Sobol sensitivity coefficient is entirely resolved, as long as it is the largest coefficient, additional polynomials will be added to it. If α instead is 2, the Sobol sensitivity information is ignored and only polynomial impacts are considered. In this case, no new subsets will ever be added. The default algorithm is restored with $\alpha = 1$. While we recommend strongly against $\alpha = 0$ and $\alpha = 2$, there is a range of values that can provide some manual control to either prefer resolving polynomials ($\alpha < 1$) or prefer adding new subsets ($\alpha > 1$).

The use of predictive measures to algorithmically choose a path of polynomial exploration is an addition from previous efforts to couple SCgPC with HDMR. Previously, such as in [30], the algorithm evaluates all potential candidates then keeps the most impactful one. While this is more guaranteed to find the most effective path, it also is much more computationally expensive. Even if the predictive adaptive algorithm guesses incorrectly, it can do so many times before matching the expense of the non-predictive algorithm.

Whenever an iteration is taken in the algorithm, it is important to re-calculate all the sensitivities and impacts, both estimated and actual, for each subset and polynomial. Because of the properties of both the SCgPC and HDMR expansions demonstrated in Appendix B, this is quite computationally inexpensive. Whenever a new element is added to the global expansion, it changes the total variance, and so adjusts all the impact parameters.

5.6.3 Initializing Subsets

When the algorithm determines a new subset should be added to the expansion, traditionally we would initialize the subset as we would a typical adaptive sparse grid expansion, with a single first-order polynomial in each direction making up initial the polynomial index set. However, this is a poor choice for this algorithm. Because the

(1,0)	
(0,0)	(0,1)

TABLE 5.1: Standard SCgPC Initialization

(1,0)	(1,1)
(0,0)	(0,1)

TABLE 5.2: SCgPC Initialization for Adaptive HDMR

TABLE 5.3: Index Set Initialization, Adaptive HDMR

algorithm has selected a new subset, the impact of the new subset will be zero unless it adds at least a polynomial that is first order in all the inputs that are part of the subset.

As a result, for this combined adaptive algorithm we initialize each subset with a tensor combination of linear polynomials instead of the traditional collection of non-interactive first-order polynomials only. This assures at least tensor first-order behavior in the subspace is added to the HDMR expansion whenever a subset is added. This is shown graphically in Table 5.3.

5.6.4 Polynomial Uniqueness

We note that the same polynomial may appear in several different subset terms and have a different impact in each. For example, the first-order polynomial $\phi_1(y_1)$ appears both in the SCgPC expansion for subset t_{y_1} as well as subset t_{y_1,y_2} (as $\phi_{1,0}(y_1, y_2)$). As a result, the impact parameter η_{y_1} might ambiguously have multiple definitions depending on which subset is considered. However, since $\phi_1(y_1) = \phi_{1,0}(y_1, y_2)$ is technically only a function of y_1, it should be associated with subset t_{y_1} only. To simplify this problem, we restrict eligible polynomials in each subset to include all nonzero orders for inputs on which the subset relies. For example, the polynomial order $k = (1, 0, 0)$ in a three-dimensional problem is potentially eligible for subset t_1 but we restrict it from being eligible for $t_{1,2}$ in the adaptive search algorithm.

If a subset is selected to add a polynomial in the adaptive algorithm and the selected polynomial depends on a polynomial with lower effective dimensionality, that lower-order polynomial is added to the lower-dimensional subset at the same time the selected polynomial is added to the selected subset. For example, consider an adaptive cut-HDMR expansion $T(x, y)$ of a two-dimensional model $u(x, y)$ that consists of three subsets $t_x(x)$, $t_y(y)$, and $t_{x,y}(x, y)$. This example is shown graphically in Table 5.7. Let the adaptive polynomial set for $t_x(x)$ be $\Lambda_x = ((0, 0), (1, 0))$, for $t_y(y)$ be $\Lambda_y = ((0, 0), (0, 1))$, and for $t_{x,y}(x, y)$ be $\Lambda_{x,y} = ((0, 0), (0, 1)(1, 0), (1, 1))$. Further, let the adaptive cut-HDMR algorithm have selected the polynomial $k = (1, 2)$ to add to subset $t_{x,y}(x, y)$. Traditionally, this would require $k = (0, 2)$ to be present first. However, the algorithm indicates there is more to be gained from expanding the interaction of x, y. Because $(0, 2)$ belongs to subset $t_x(x)$, the adaptive algorithm step adds $k = (1, 2)$ to

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

TABLE 5.4: Λ_x for $t_x(x)$

(1,0)
(0,0)

TABLE 5.5: Λ_y for $t_y(y)$

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

TABLE 5.6: $\Lambda_{x,y}$ for $t_{x,y}(x,y)$

TABLE 5.7: Polynomial Dependency in Adaptive HDMR

$t_{x,y}(x,y)$ and $(0,2)$ to $t_x(x)$ simultaneously. This is most likely to occur when there are strong interaction effects that overshadow single-input effects.

We consider this example again as shown graphically in Table 5.7. On the left is the index set for the subset only dependent on x, while on the right is the index set for the subset dependent on x, y. The unmarked indices are the starting indices for each subset. The boxed index (1,2) on the right is the polynomial the adaptive algorithm has selected to add to subset $t_{x,y}(x,y)$. As a result, however, the polynomial in gray (0,2) must be added, both to the index set for $t_{x,y}(x,y)$ as well as to the index set for $t_x(x)$. We add index (0,2) in the same adaptive step that the index (1,2) is added.

5.7 Conclusion

We have presented existing algorithms using the HDMR and cut-HDMR expansion, and expanded those algorithms to include interoperability with SCgPC as well as predictive algorithmic searching. We have discussed some of the obstacles encountered while implementing these features and offered solutions to ensure more reliable adaptive searches. There are several further improvements that can be made to this combined adaptive algorithm, which we discuss in section 10.4.

In Chapter 6 we analyze the performance of HDMR methods in contrast to pure SCgPC and MC on the analytic models introduced in Chapter 4.

Chapter 6

Results for High-Dimension Model Representation

6.1 Introduction

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

As with SCgPC, the objective in introducing HDMR methods is to reduce the number of computational model solves necessary to obtain reasonable second-order statistics for models. The analytic models we use for initial demonstration are described in Chapter 4, along with the performance of SCgPC methods in representing the same. In this chapter, we add HDMR methods to the analysis, and consider their performance in comparison to MC and SCgPC methods.

We consider three static HDMR cases as well as adaptive HDMR using adaptive SCgPC to expand subset terms. In static HDMR, each series represents an HDMR truncation level. The HDMR truncation level determines the maximum level of interactions allowable in the expansion. For example, HDMR 1 includes only first-order interactions. We consider first-, second-, and third-order HDMR truncations. For each of these truncations, we use SCgPC expansions of growing polynomial order limit, with polynomials selected by using the total degree index set construction method. Each successive data point in each static series is obtained by increasing the limiting total polynomial order L for the subset terms.

As discussed in section 5.4, we do not expect the convergence rate of the static HDMR methods to exceed the convergence rate of their corresponding SCgPC counterparts. However, there may be some static HDMR constructions that require less computational solves than SCgPC methods. In addition, we expect the flexibility of the adaptive HDMR method to allow improved convergence over other methods for some models.

As in Chapter 4, the performance of each method is analyzed using *value figures* and *convergence figures* for both the mean and standard deviation of each model's response. Value figures will provide actual values obtained for the moments, while convergence figures show the relative error of the values obtained to the analytic benchmark value. Monte Carlo error bars are calculated as described in Chapter 4.

All computations shown here were performed using the `RAVEN` [2] framework. As noted previously, 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.

6.2 Tensor Monomials

This model is described in section 4.2.1. As the tensor product of linear polynomials, it is very conducive to SCgPC. Because all terms are equally important, however, truncating the HDMR expansion of this model removes important elements, which makes HDMR less ideal for this model in general.

6.2.1 3 Inputs

With only three input parameters, we can clearly see the contribution of the first-order interaction terms, second-order interaction terms, and third-order interaction terms from HDMR 1, HDMR 2, and HDMR 3. As expected, with only first-order polynomials HDMR 3 converges exactly; however, HDMR 1 and HDMR 2 neglect critical polynomials in the expansion, and so are less suitable methods for this model. Note also that after first-order polynomials, adding additional polynomial orders does not reduce error for the static HDMR methods, as no higher-order polynomials exist in the original model. The adaptive HDMR method, however, performs admirably for this model, quickly finding the appropriate search direction for the response dependencies and converging as rapidly as the adaptive SCgPC method.

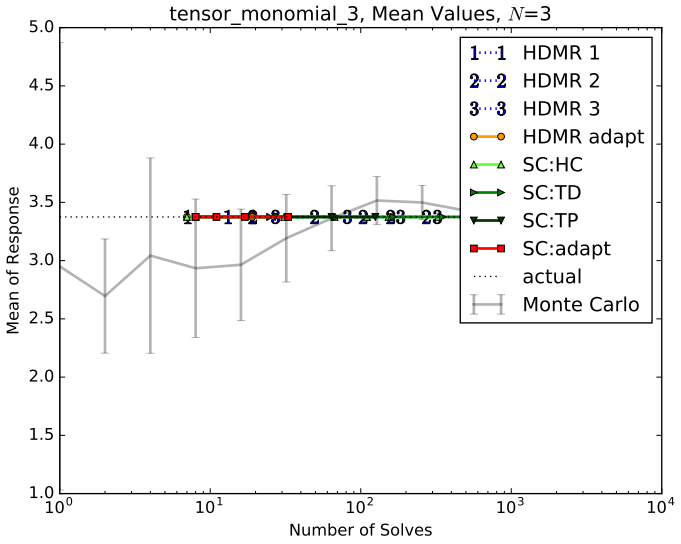

FIGURE 6.1: Tensor Monomial, $N = 3$, Mean Values

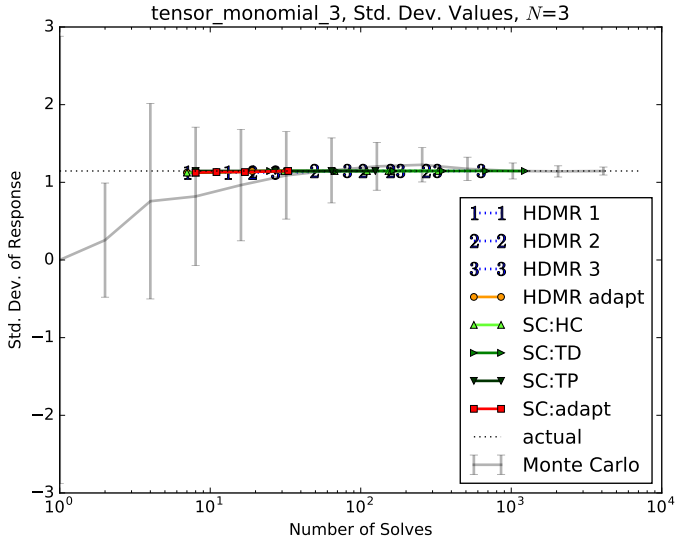

FIGURE 6.2: Tensor Monomial, $N = 3$, Std. Dev. Values

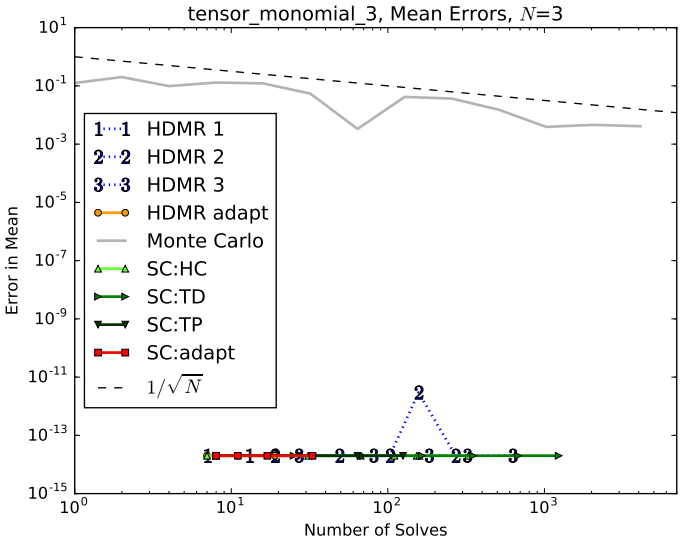

FIGURE 6.3: Tensor Monomial, $N = 3$, Mean Convergence

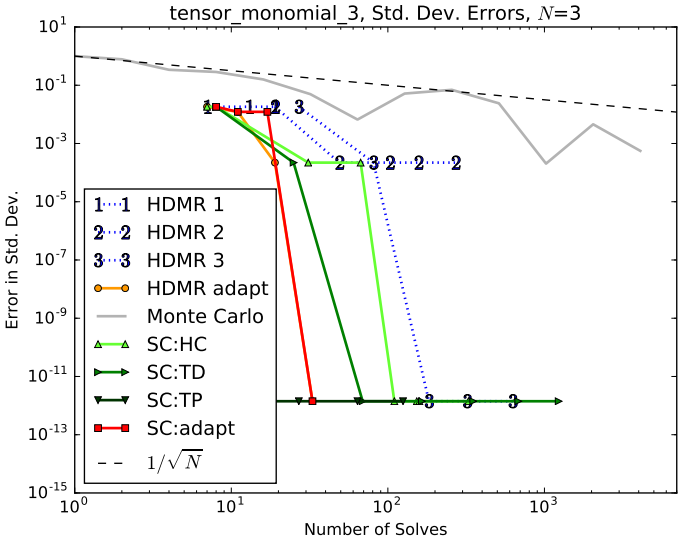

FIGURE 6.4: Tensor Monomial, $N = 3$, Std. Dev. Convergence

6.2.2 5 Inputs

Increasing the dimensionality serves to enforce those observations already recorded for the three-dimensional input space. HDMR methods perform no better than their SCgPC counterparts, and because of their truncation are limited in their ability to converge the statistical moments for this model. Interestingly, however, the adaptive HDMR method outperforms the adaptive SCgPC method in finding the exact solution, because it searches both subspaces to add as well as polynomials, and wastes less time searching

higher-order polynomials that do not exist in the expansion. In essence, it eliminates portions of the Hilbert spaced spanned by the bases polynomials more quickly than the adaptive SCgPC method.

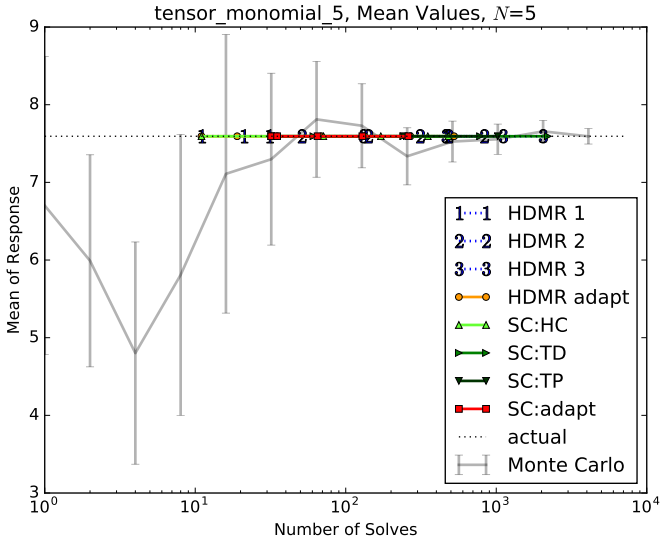

FIGURE 6.5: Tensor Monomial, $N = 5$, Mean Values

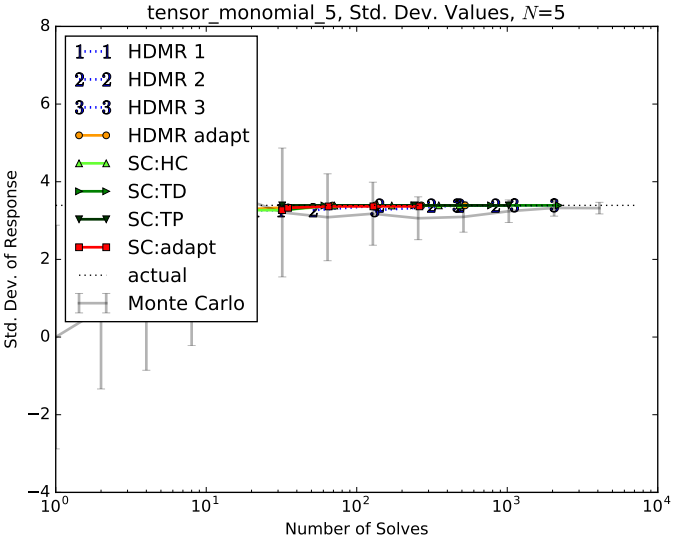

FIGURE 6.6: Tensor Monomial, $N = 5$, Std. Dev. Values

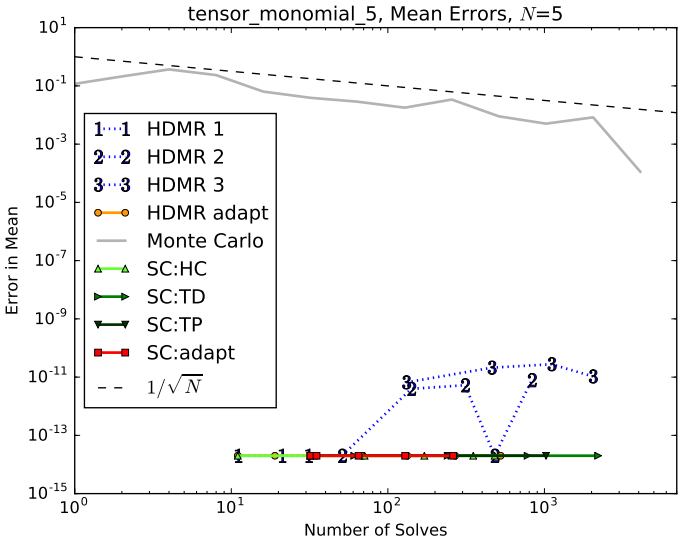

FIGURE 6.7: Tensor Monomial, $N = 5$, Mean Convergence

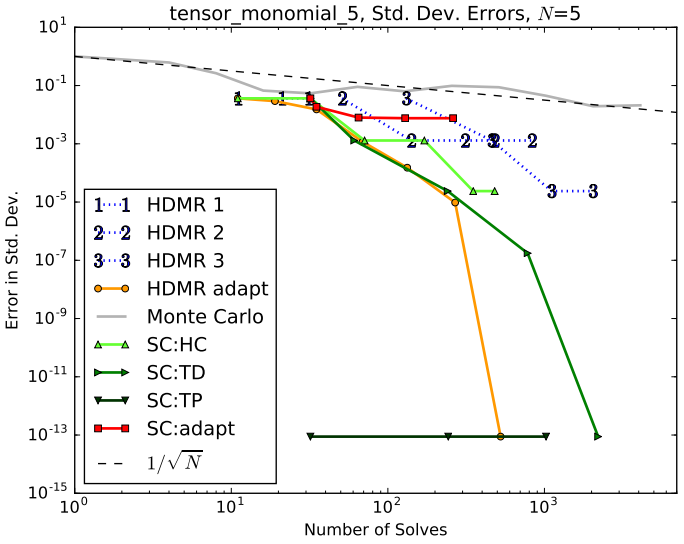

FIGURE 6.8: Tensor Monomial, $N = 5$, Std. Dev. Convergence

6.2.3 10 Inputs

Moving to an input space with dimensionality 10, we predictably see the static HDMR methods performing quite poorly for this model. Because the model includes polynomial interactions up to tenth order, truncating at even three orders incurs significant error. However, we note the adaptive HDMR method appears to be performing at least as well as any other method for this larger dimensionality, for the same reasons as discussed in the 5-input case. Note also that the adaptive HDMR method obtains representation long before the total degree or tensor product methods do.

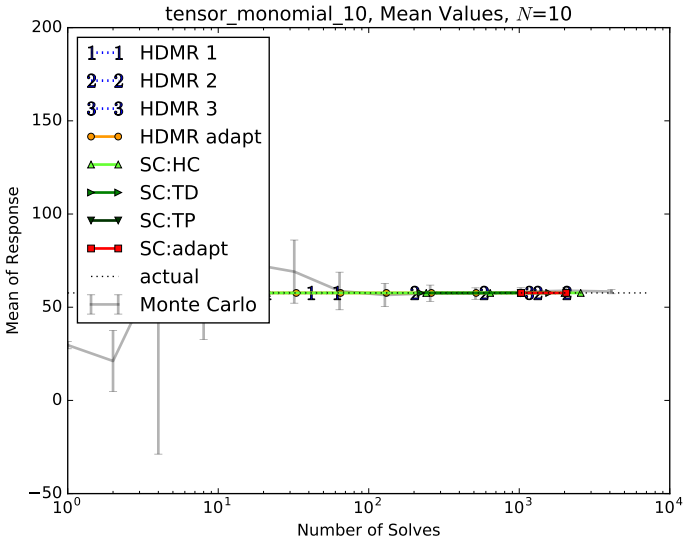

FIGURE 6.9: Tensor Monomial, $N = 10$, Mean Values

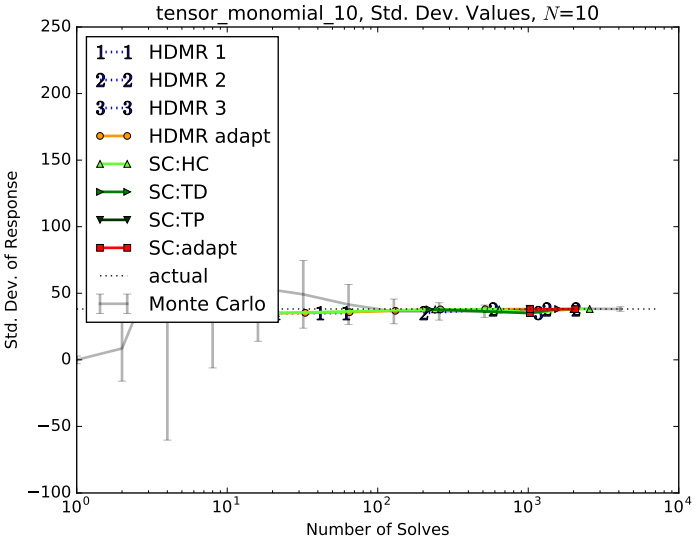

FIGURE 6.10: Tensor Monomial, $N = 10$, Std. Dev. Values

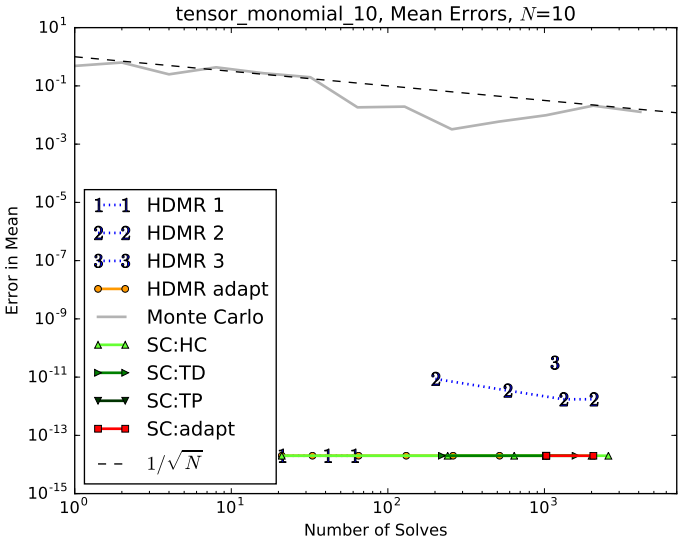

FIGURE 6.11: Tensor Monomial, $N = 10$, Mean Convergence

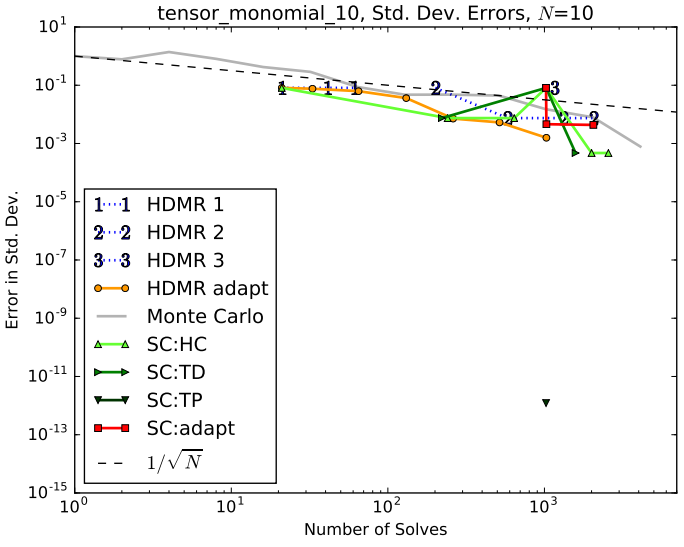

FIGURE 6.12: Tensor Monomial, $N = 10$, Std. Dev. Convergence

6.3 Sudret Polynomial

This model is described in section 4.3.1. This model is similar to the tensor linear polynomials, but instead is a tensor product of second-order polynomials. We observe similar performance here as for the tensor monomials, but with faster degredation as the input dimensionality increases.

6.3.1 3 Inputs

Because of the tensor construction of these polynomials, the HDMR truncation level once again plays a critical role in determining the error of the HDMR methods. The three plateaus for HDMR 1, HDMR 2, and HDMR 3 show that adding higher than second-order polynomials will not substantially decrease the error in these expansions, indicating that the error is dominated by the HDMR truncation error. Note also that while the adaptive HDMR method performs well, it is outperformed by adaptive SCgPC. This is because it is challenging for the adaptive HDMR method to find the second-order polynomials while the first-order polynomials have no contribution to the expansion. This is especially seen in the convergence of the standard deviation. Note that for the standard deviation, HDMR 3 is still converging; this is because the HDMR subsets are expanded in total degree index sets, which require higher-order polynomial limits to obtain the tensor product of second-order polynomials; in particular, third-order interaction subsets require total degree order 6 to obtain the polynomial with orders $k = (2, 2, 2)$, which is required for this model.

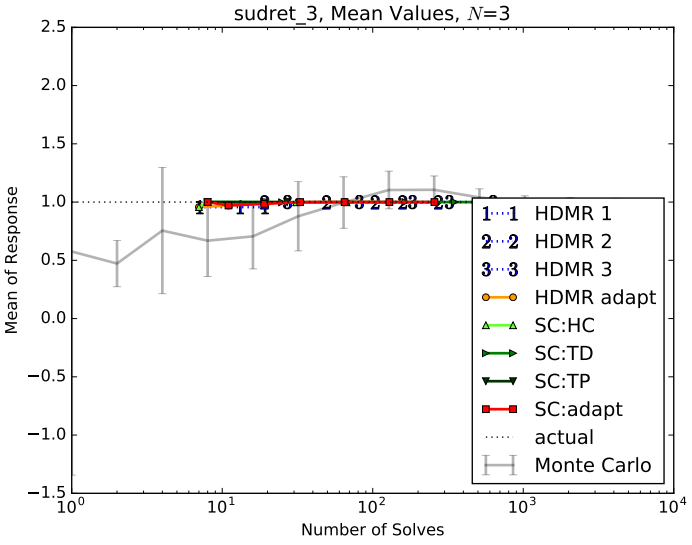

FIGURE 6.13: Sudret Polynomial, $N = 3$, Mean Values

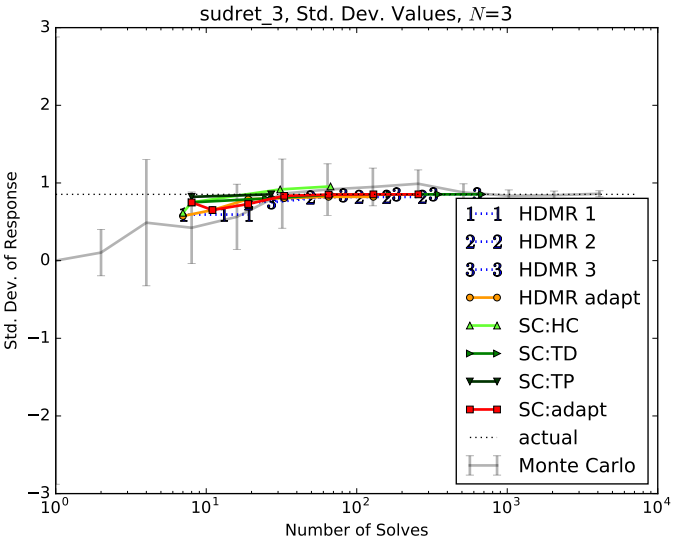

FIGURE 6.14: Sudret Polynomial, $N = 3$, Std. Dev. Values

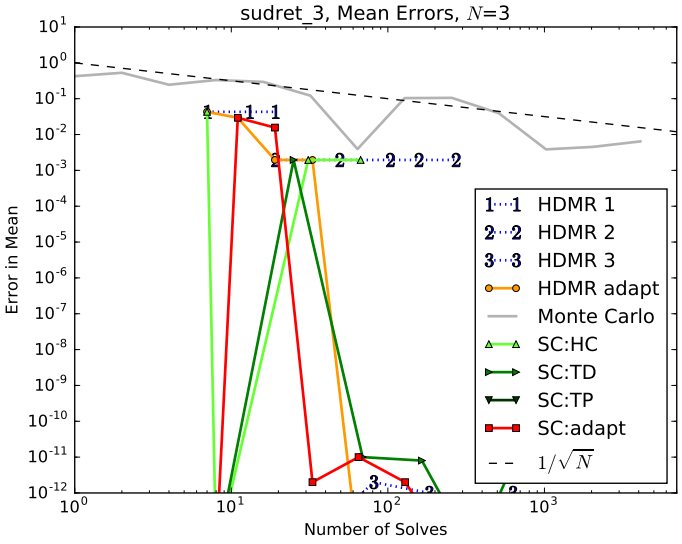

FIGURE 6.15: Sudret Polynomial, $N = 3$, Mean Convergence

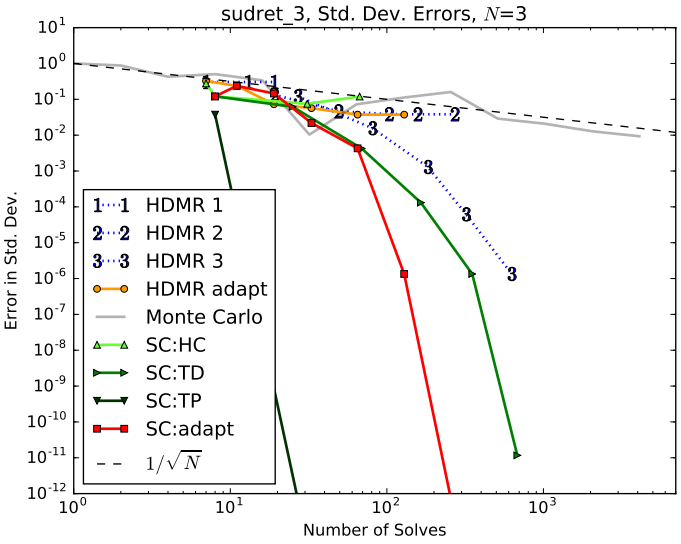

FIGURE 6.16: Sudret Polynomial, $N = 3$, Std. Dev. Convergence

6.3.2 5 Inputs

We continue to see degredation of performance from HDMR methods moving from three inputs to five. The truncation of HDMR 1, HDMR 2, and HDMR 3 incurs too much error to converge significantly, and the adaptive HDMR method struggles like the adaptive SCgPC method to explore the polynomial space effectively.

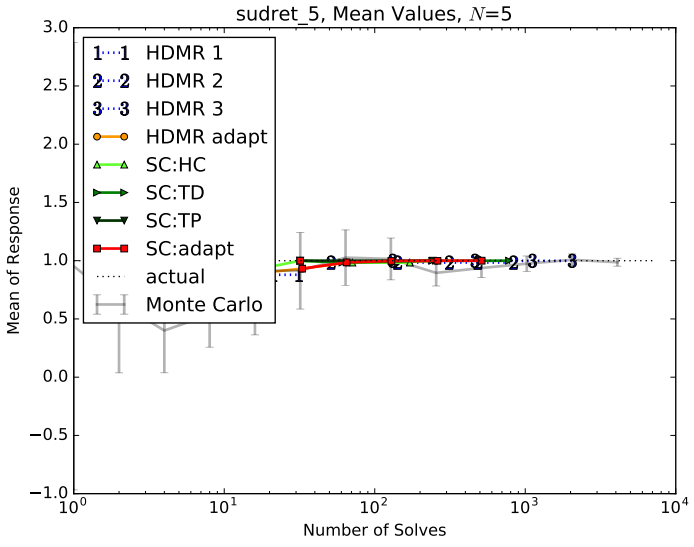

FIGURE 6.17: Sudret Polynomial, $N = 5$, Mean Values

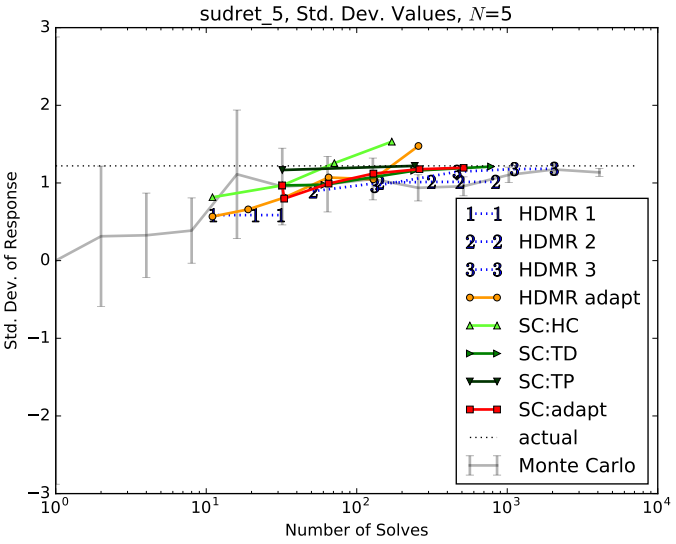

FIGURE 6.18: Sudret Polynomial, $N = 5$, Std. Dev. Values

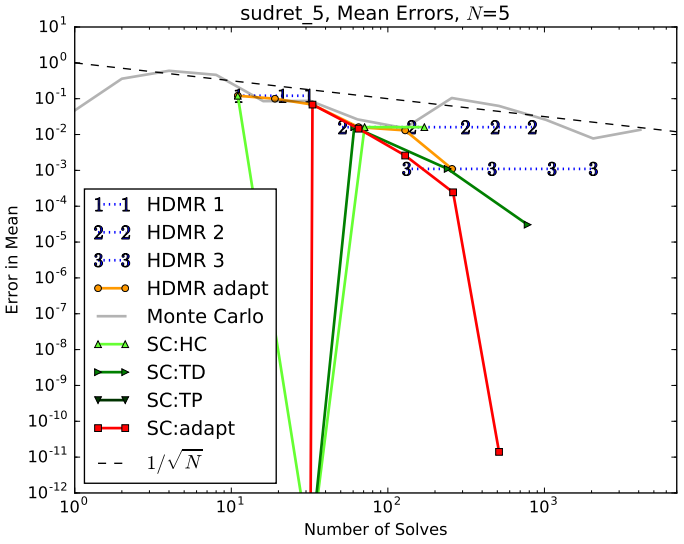

FIGURE 6.19: Sudret Polynomial, $N = 5$, Mean Convergence

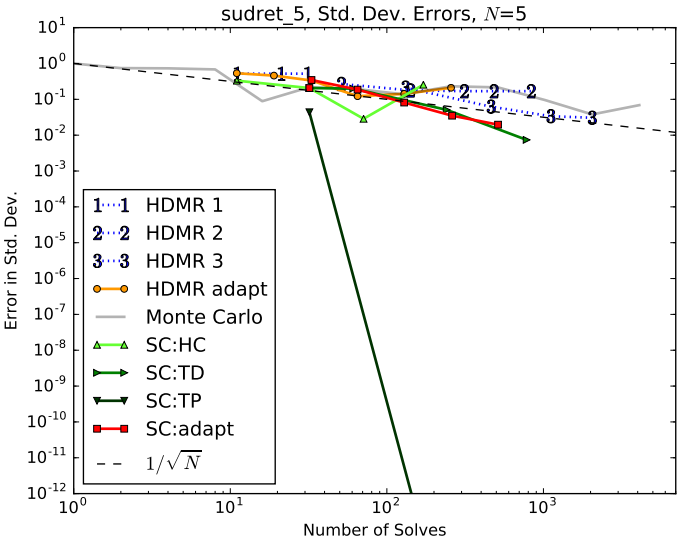

FIGURE 6.20: Sudret Polynomial, $N = 5$, Std. Dev. Convergence

6.4 Attenuation

This model is described in section 4.4.1. As the tensor product of polynomials whose scaling drops off with increasing order (see Table 4.5), this model is well-suited to HDMR methods. However, this model demonstrates some of the limitations in the default search parameters for the adaptive HDMR method. While combinations of polynomials of similar order are most important to this expansion, the adaptive HDMR method tends to expand polynomials with low interaction because of the sensitivity estimation methods. This demonstrates when the adaptive HDMR might be ill-suited to exploring the input space.

6.4.1 2 Inputs

Because the input space is only two-dimensional, HDMR 3 is equivalent to HDMR 2 and not shown in this case. As with the SCgPC methods, all HDMR methods show exponential convergence on the mean and standard deviation for this response, although the adaptive HDMR does not perform as well as the adaptive SCgPC method because of the way the polynomial coefficients decay. The HDMR 1 method stagnates at first-order interactions, and error is dominated by the HDMR truncation instead of polynomial truncation.

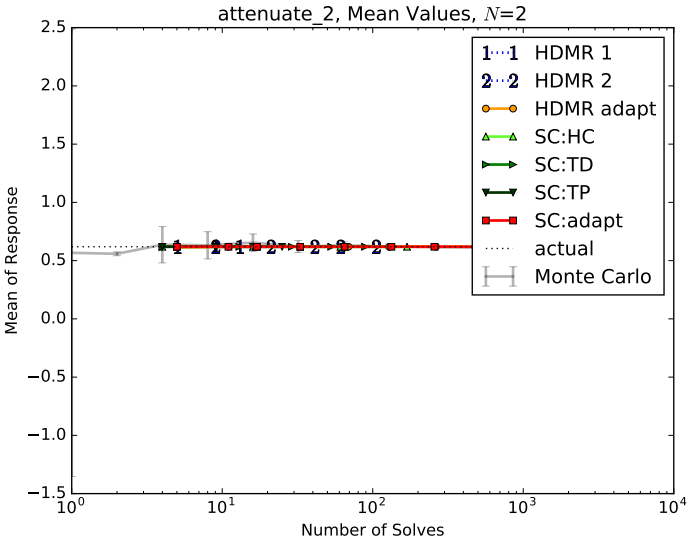

FIGURE 6.21: Attenuation, $N = 2$, Mean Values

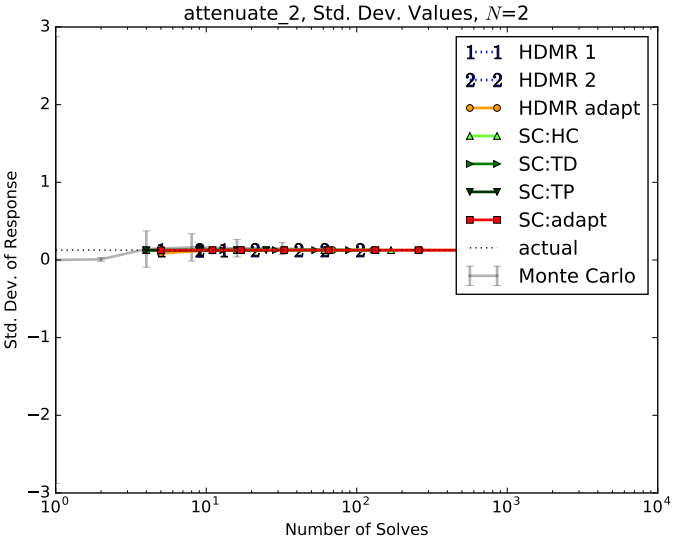

FIGURE 6.22: Attenuation, $N = 2$, Std. Dev. Values

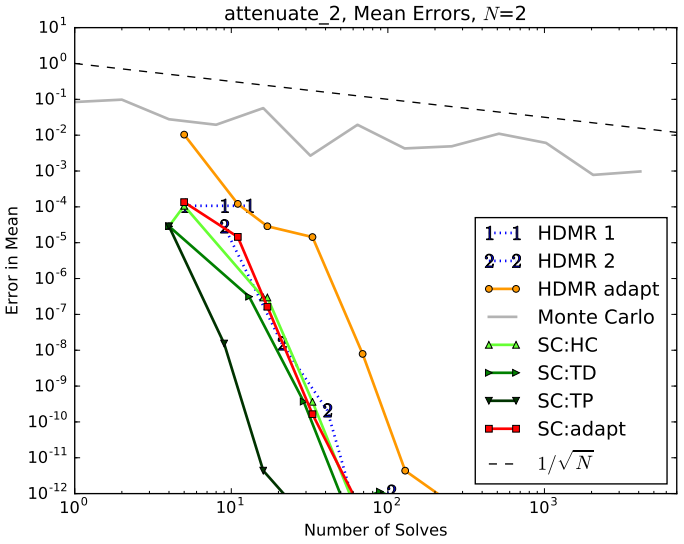

FIGURE 6.23: Attenuation, $N = 2$, Mean Convergence

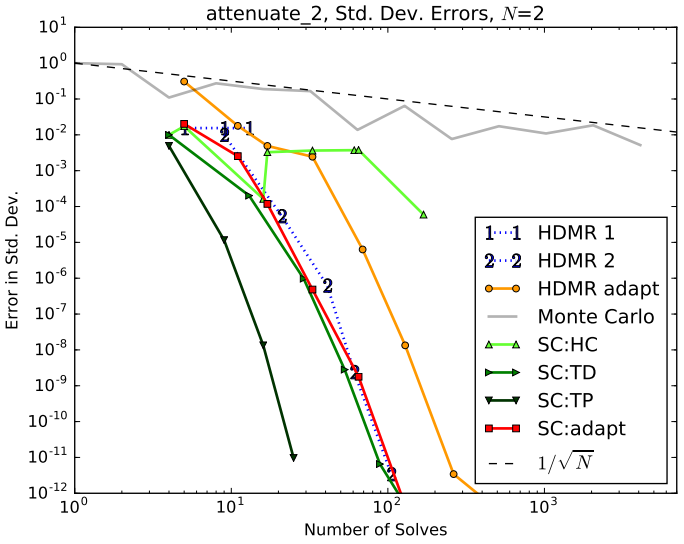

FIGURE 6.24: Attenuation, $N = 2$, Std. Dev. Convergence

6.4.2 4 Inputs

The same trends exist here as for the two-dimensional case, but with degradation because of the increase in dimensionality. The plateaus for HDMR 1, HDMR 2, and HDMR 3 are all clearly evident. These plateaus indicate when error is dominated by HDMR truncation instead of polynomial truncation. It is also worth noting that while the adaptive HDMR method hits a plateau for some time in this model, it obtains a decent approximation of the standard deviation and mean earlier than most of the other static methods.

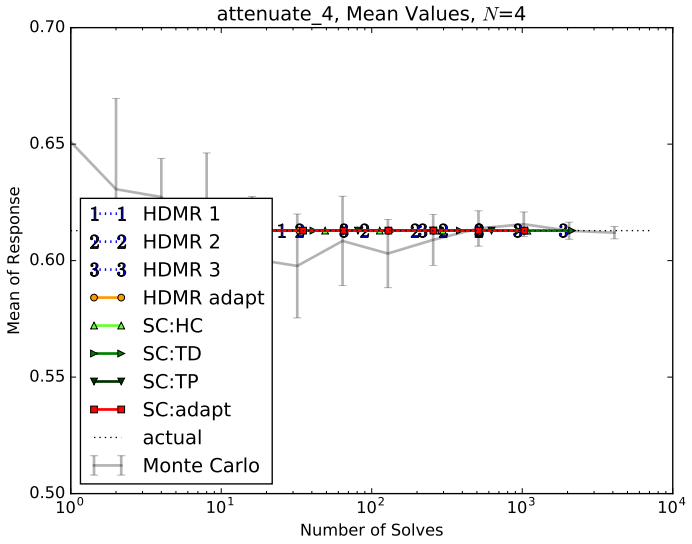

FIGURE 6.25: Attenuation, $N = 4$, Mean Values

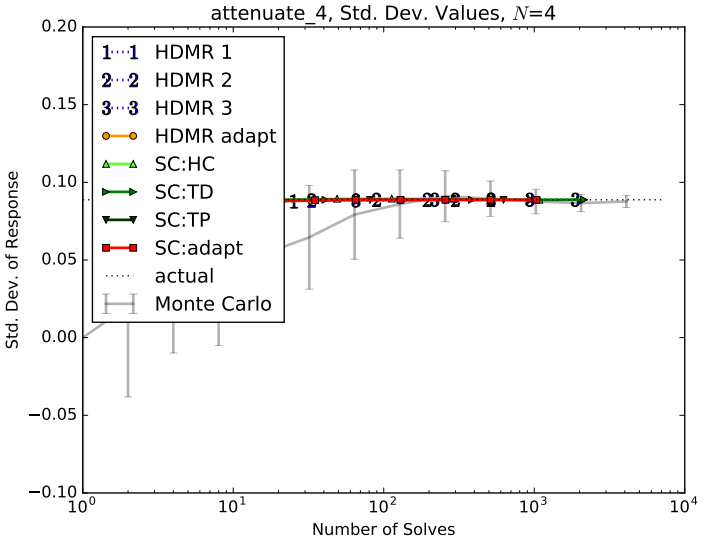

FIGURE 6.26: Attenuation, $N = 4$, Std. Dev. Values

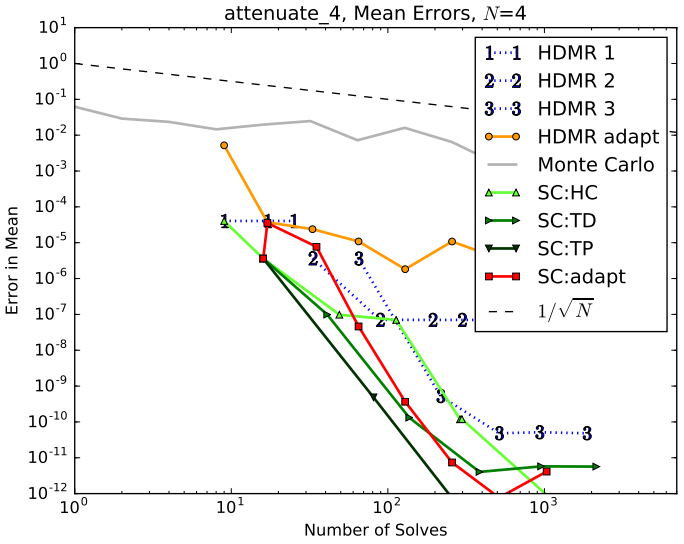

FIGURE 6.27: Attenuation, $N = 4$, Mean Convergence

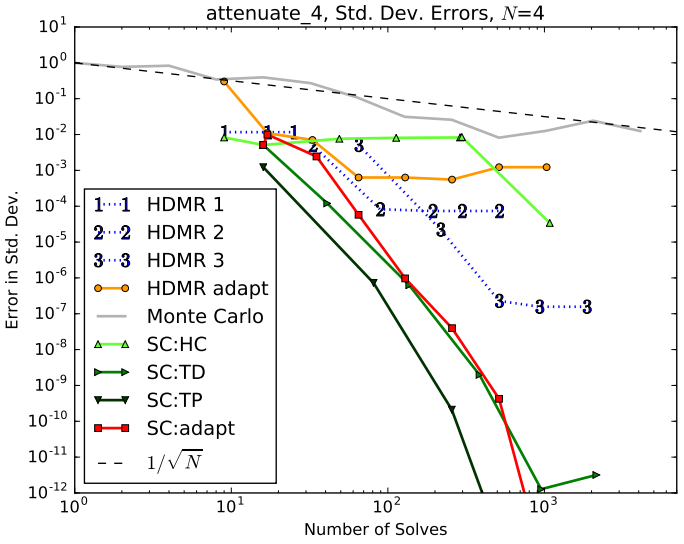

FIGURE 6.28: Attenuation, $N = 4$, Std. Dev. Convergence

6.4.3 6 Inputs

The SCgPC and HDMR methods continue to degrade as input space increases in dimensionality. The static method plateaus are still evident. The adaptive SCgPC and adaptive HDMR methods both struggle to find the appropriate polynomials to use to most accurately represent this model.

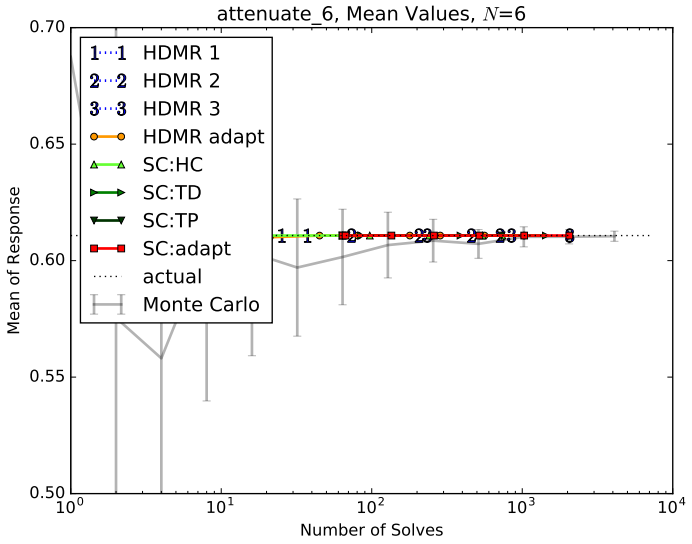

FIGURE 6.29: Attenuation, $N = 6$, Mean Values

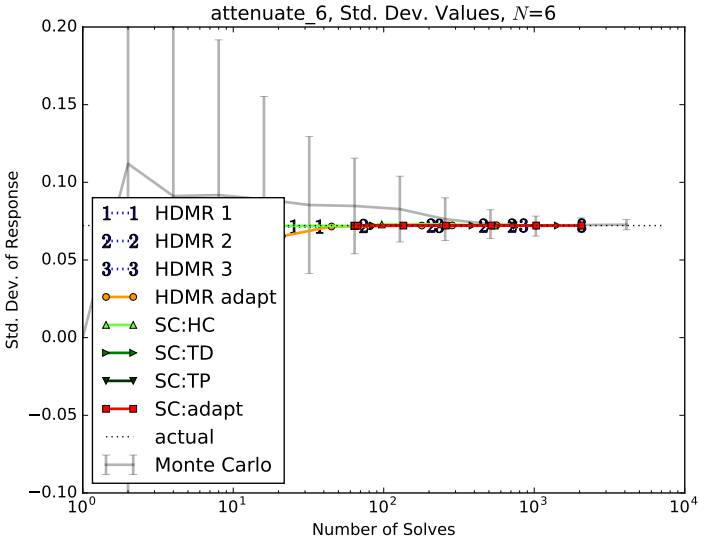

FIGURE 6.30: Attenuation, $N = 6$, Std. Dev. Values

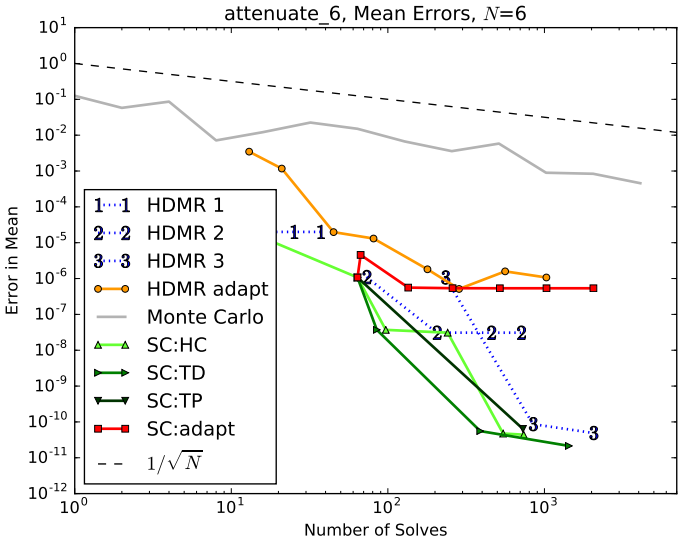

FIGURE 6.31: Attenuation, $N = 6$, Mean Convergence

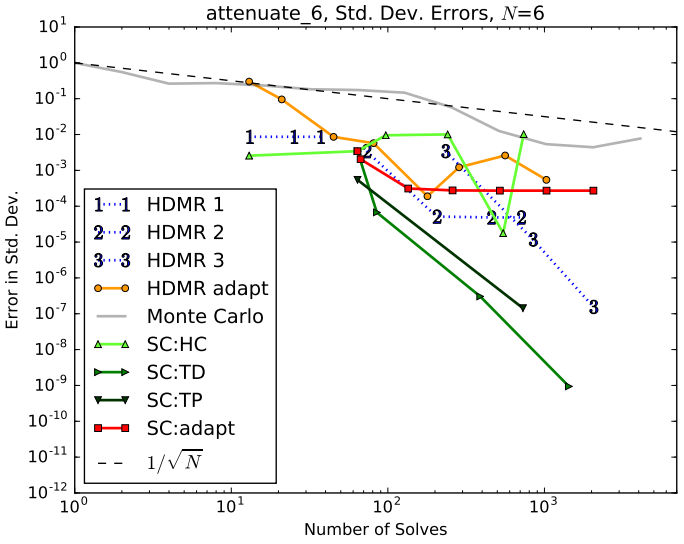

FIGURE 6.32: Attenuation, $N = 6$, Std. Dev. Convergence

6.5 Gauss Peak

This model is described in section 4.5.1. As discussed there, this model exhibits slow polynomial coefficient drop off, and all polynomials containing an odd number have a zero coefficient. This yields the adaptive search algorithms paralyzed, as any attempts the assumption of monotonically-decreasing polynomial coefficients is a poor assumption for this model.

6.5.1 3 Inputs

The static HDMR methods show no improvement over Monte Carlo for this model. It requires a great number of polynomials in tensor combination to accurately reproduce this response; as a result, even HDMR 3 shows poor convergence for this model.

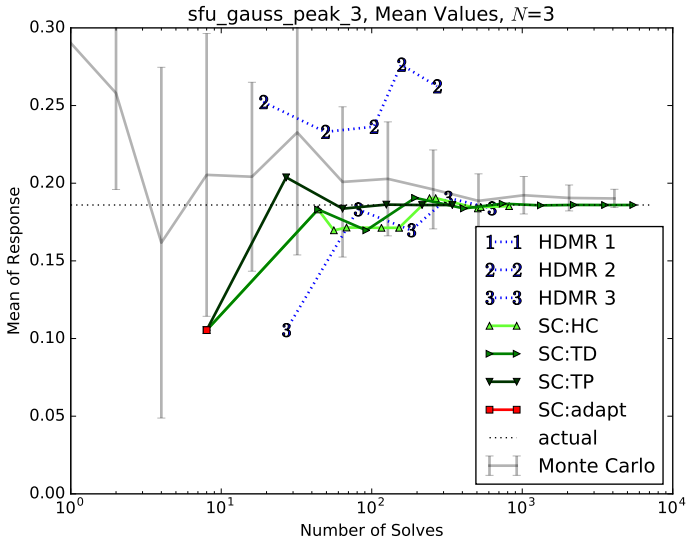

FIGURE 6.33: Gauss Peak, $N = 3$, Mean Values

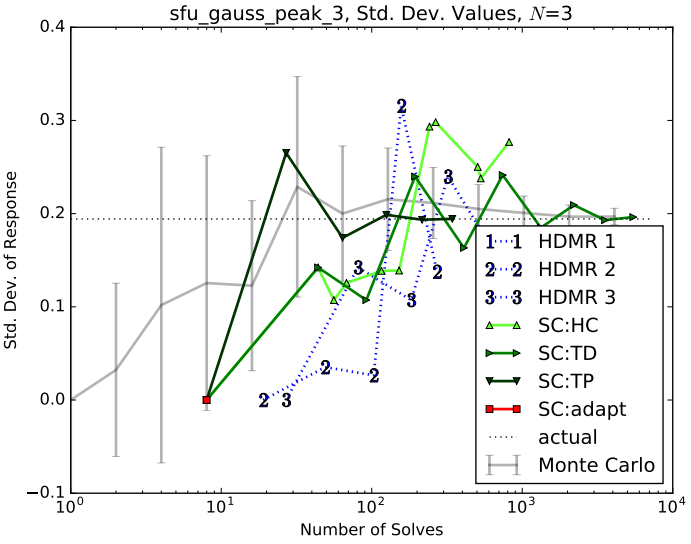

FIGURE 6.34: Gauss Peak, $N = 3$, Std. Dev. Values

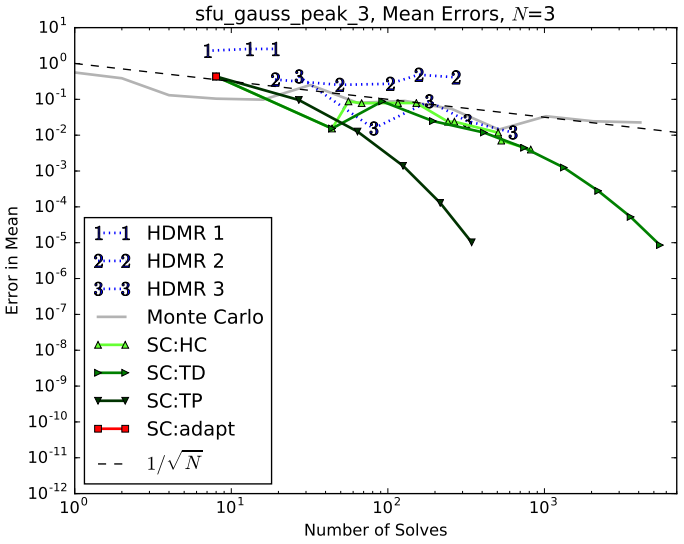

FIGURE 6.35: Gauss Peak, $N = 3$, Mean Convergence

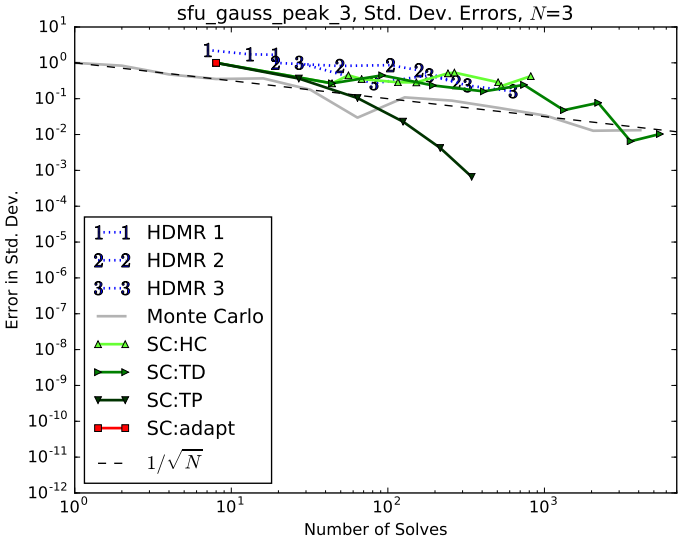

FIGURE 6.36: Gauss Peak, $N = 3$, Std. Dev. Convergence

6.5.2 5 Inputs

The same trends for the three-input case are observed here for the five-input case, and exacerbated. This model requires a great number of high-order, tensor-product polynomials to produce an accurate surrogate, and neither HDMR nor SCgPC methods are well-equipped to provides them efficiently.

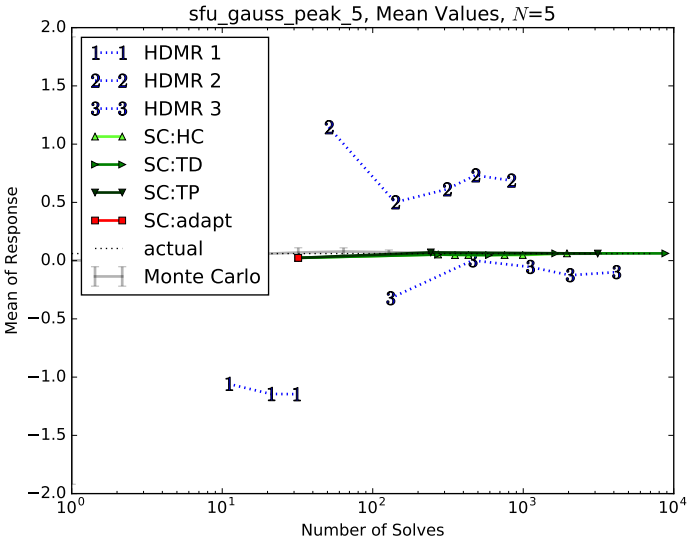

FIGURE 6.37: Gauss Peak, $N = 5$, Mean Values

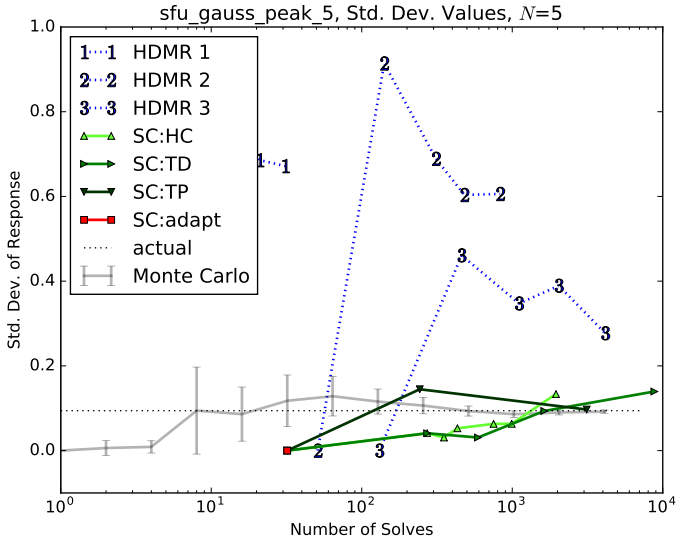

FIGURE 6.38: Gauss Peak, $N = 5$, Std. Dev. Values

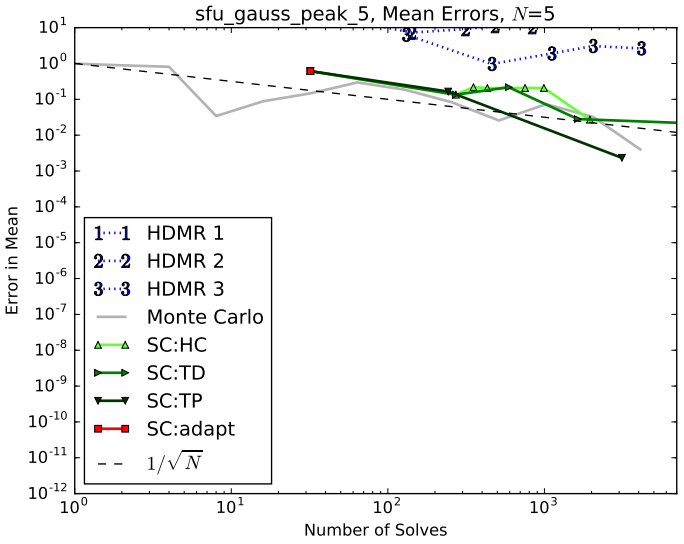

FIGURE 6.39: Gauss Peak, $N = 5$, Mean Convergence

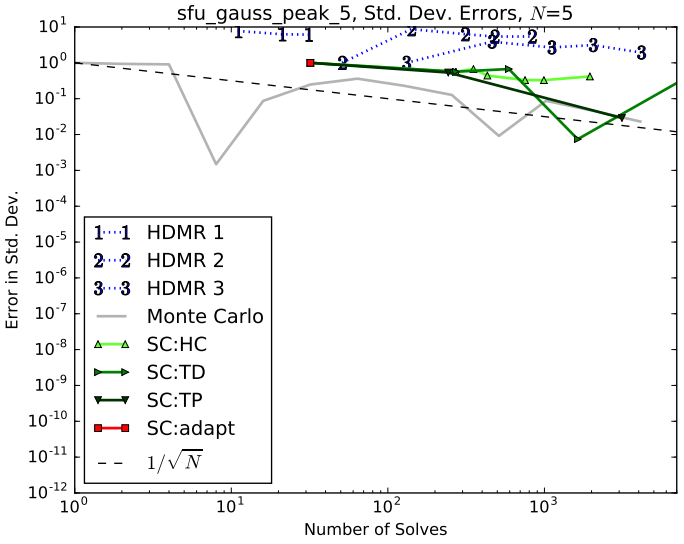

FIGURE 6.40: Gauss Peak, $N = 5$, Std. Dev. Convergence

6.6 Ishigami

6.6.1 3 Inputs

This model is described in section 4.6.1. The behavior for the HDMR method on this model are quite interesting. Because much of this response is determined by single-order interactions, the static HDMR 1 method converges the mean quite effectively compared to other methods. Both HDMR 2 and HDMR 3 also perform well, but HDMR 3 wastes

substantial effort as there are no third-order interactions in this model. The adaptive HDMR method fails in its search because it misses the important interaction between y_3 and y_1, which isn't observed until the fourth-order polynomial of y_3. Additionally, both y_1 and y_2 are arguments to sine functions, which when expanded in polynomials only contain odd powers. As a result, upon obtaining zero contribution from even-ordered polynomials, the adaptive algorithm has no metric by which to search for additional contributions.

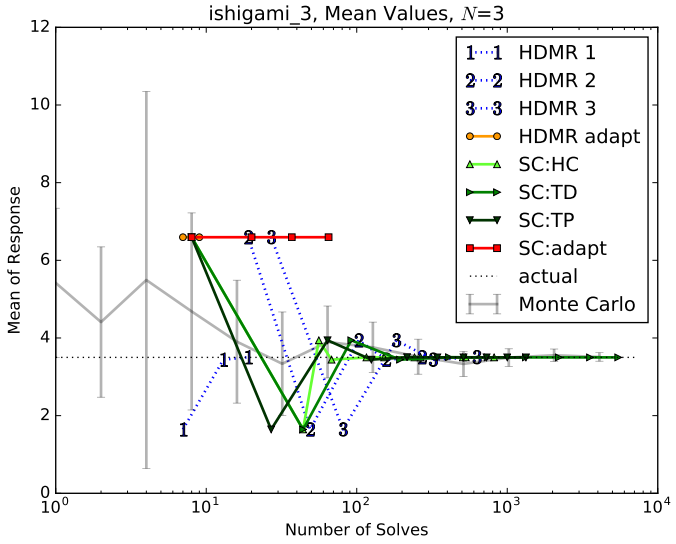

FIGURE 6.41: Ishigami, $N = 3$, Mean Values

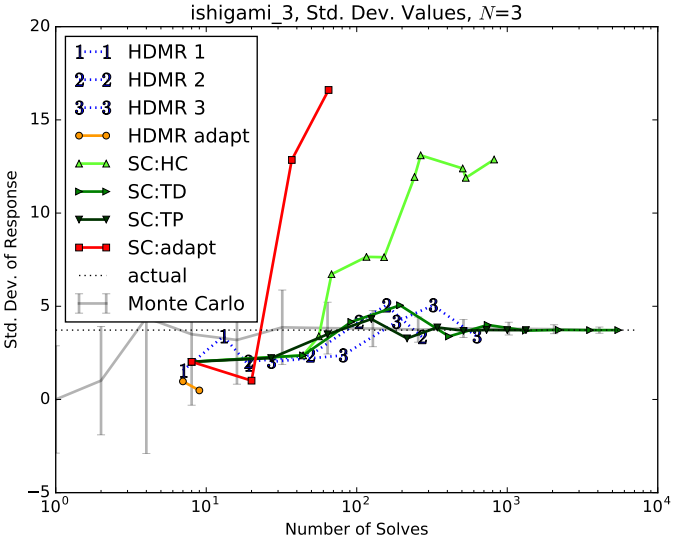

FIGURE 6.42: Ishigami, $N = 3$, Std. Dev. Values

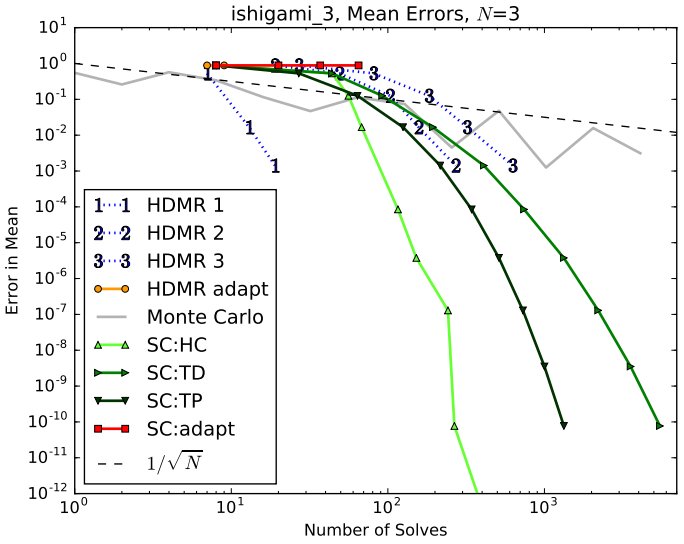

FIGURE 6.43: Ishigami, $N = 3$, Mean Convergence

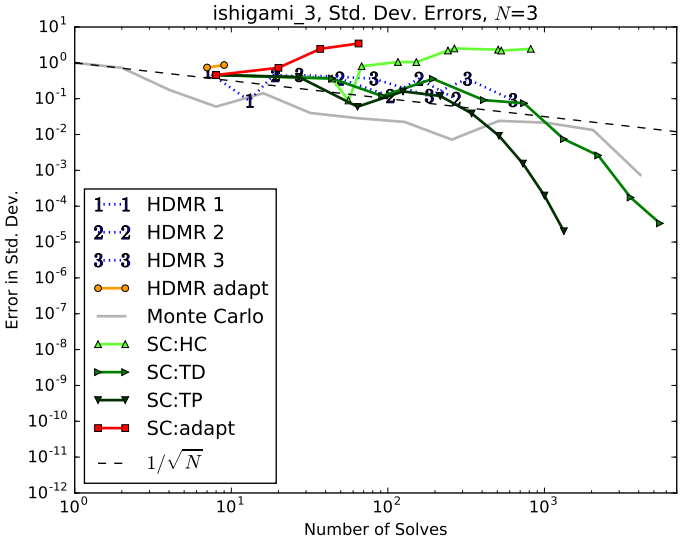

FIGURE 6.44: Ishigami, $N = 3$, Std. Dev. Convergence

6.7 Sobol G-Function

This model is described in section 4.7.1. Unsurprisingly, this zeroth-order continuous response continues to provide a great challenge to any polynomial-based method representation. The adaptive SCgPC and adaptive HDMR methods both fail to converge more than a single data point, and the static methods all show no clear benefits over traditional Monte Carlo sampling.

6.7.1 3 Inputs

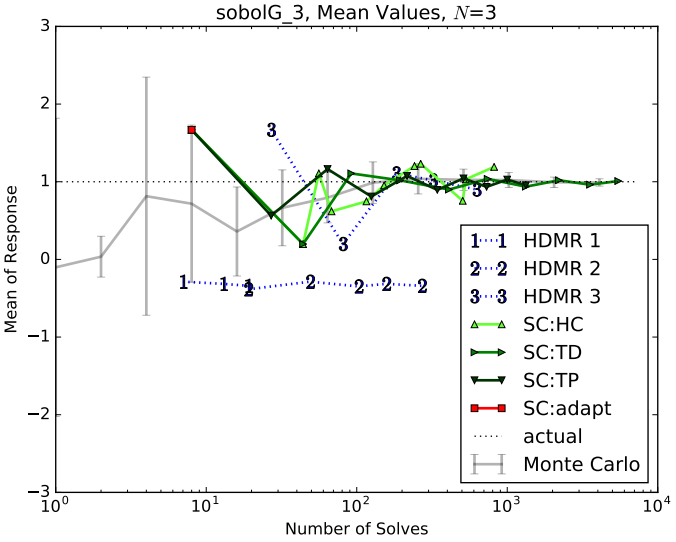

FIGURE 6.45: Sobol G-Function, $N = 3$, Mean Values

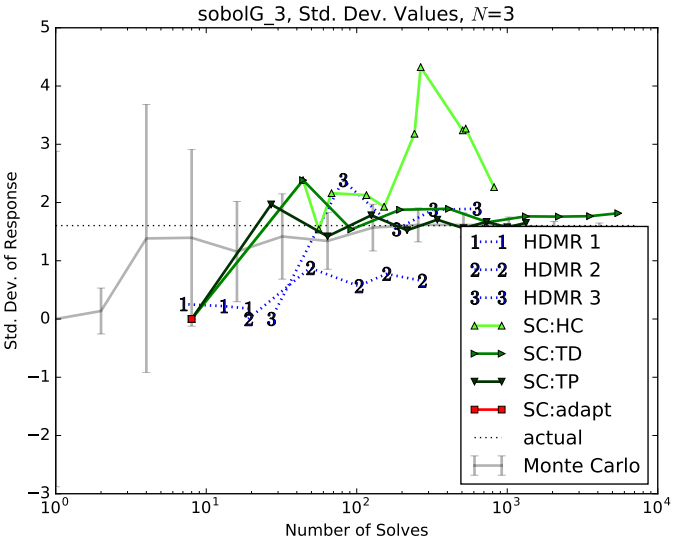

FIGURE 6.46: Sobol G-Function, $N = 3$, Std. Dev. Values

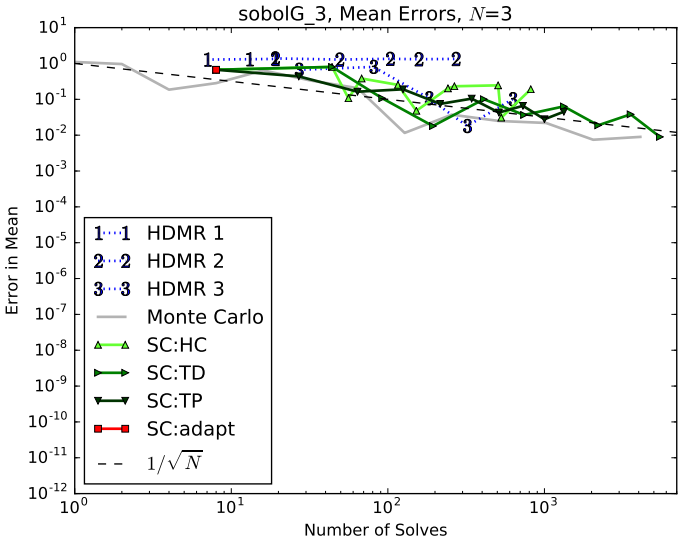

FIGURE 6.47: Sobol G-Function, $N = 3$, Mean Convergence

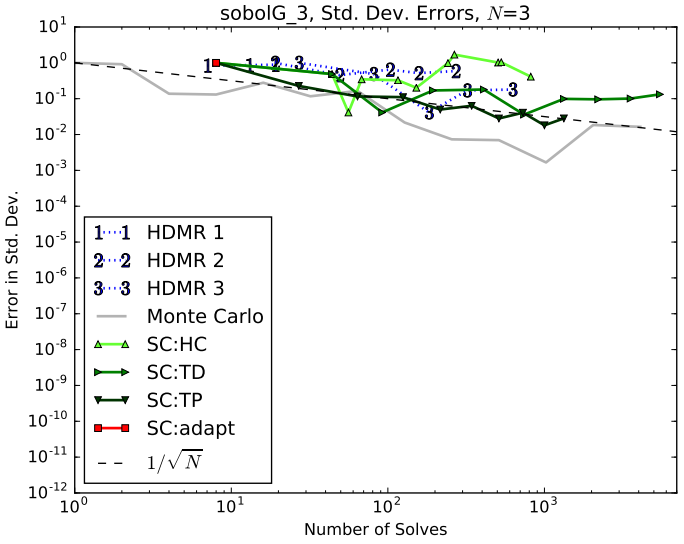

FIGURE 6.48: Sobol G-Function, $N = 3$, Std. Dev. Convergence

6.7.2 5 Inputs

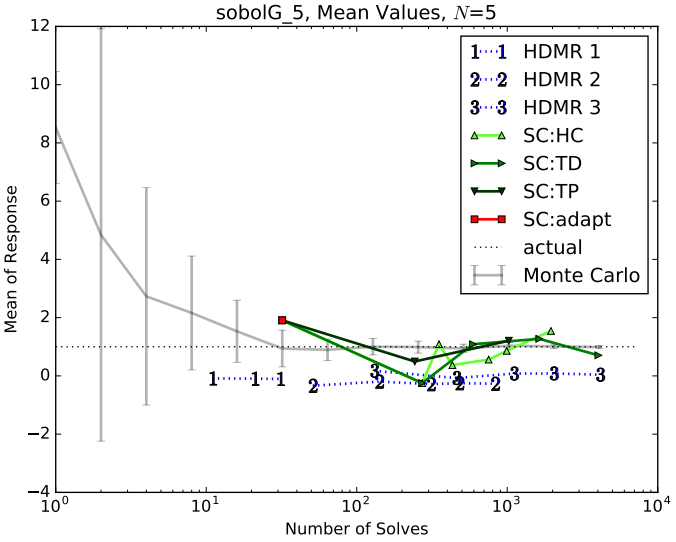

FIGURE 6.49: Sobol G-Function, $N = 5$, Mean Values

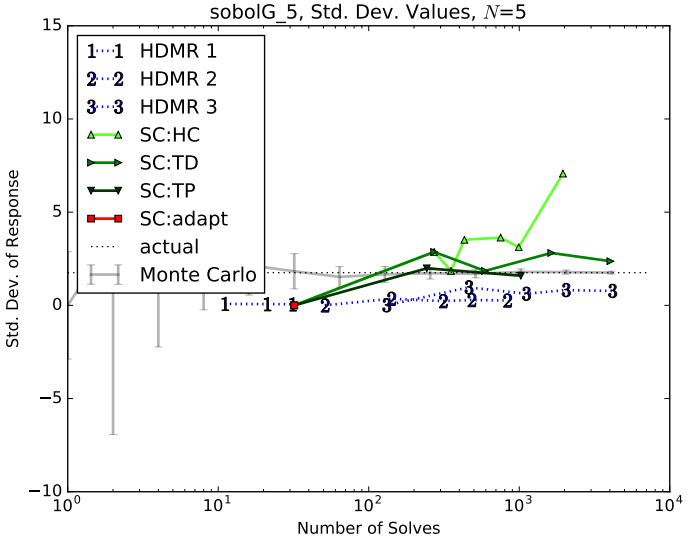

FIGURE 6.50: Sobol G-Function, $N = 5$, Std. Dev. Values

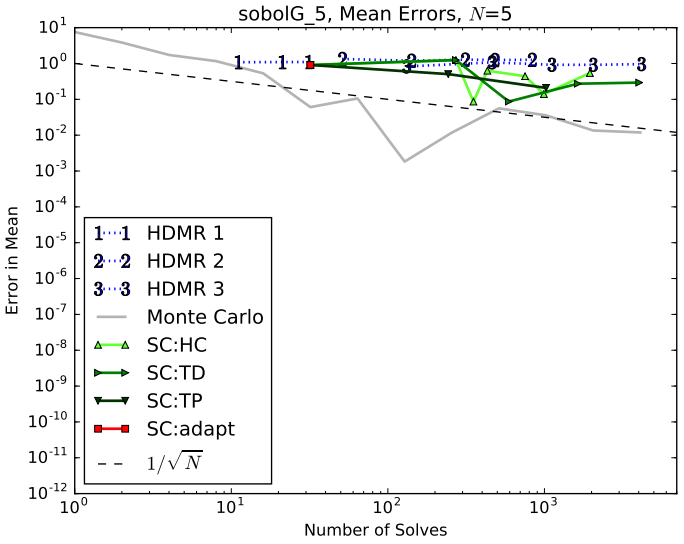

FIGURE 6.51: Sobol G-Function, $N = 5$, Mean Convergence

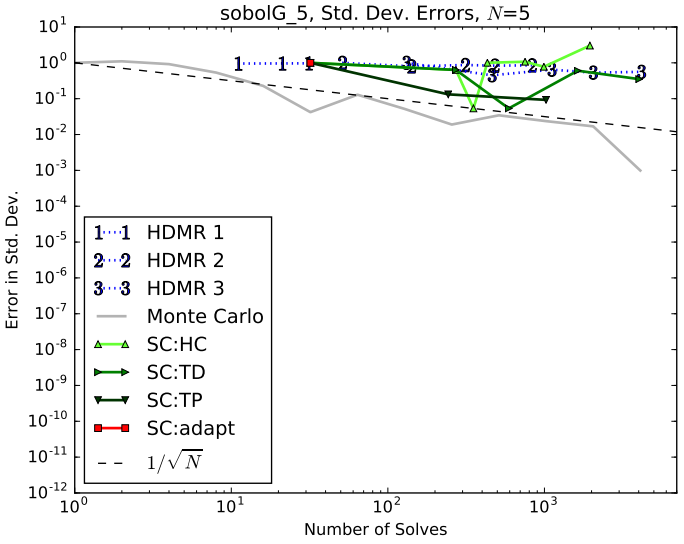

FIGURE 6.52: Sobol G-Function, $N = 5$, Std. Dev. Convergence

6.8 Conclusions

We have demonstrated the performance of HDMR methods using three different truncation orders as well as adaptive HDMR subset construction, all based on SCgPC expansions for the HDMR subsets. We observed good performance of the adaptive HDMR algorithm when the input space is limit in size and behaves predictably, and good performance in the static truncated HDMR methods when the interaction terms in the model

were limited. Additionally, we observed that adaptive HDMR and HDMR 1 could obtain UQ solutions with fewer evaluations than the other SCgPC and HDMR methods, which can be valuable when only very few realizations are practical.

However, in general SCgPC methods outperform HDMR when the input space is regular and isotropic. This leads to the conclusion that when computational resources are available, the SCgPC methods may be a better initial choice than the HDMR methods, while if resources are limited, the adaptive HDMR method can still be a good candidate.

Chapter 7

SCgPC and HDMR for Neutron Transport

7.1 Introduction

The analytic models used in chapters 4 and 6 are useful to analyze the behavior of the stochastic collocation for generalized polynomial chaos (SCgPC) and high-dimension model representation (HDMR) methods. Because they have simply-derived analytic expressions for statistical moments, convergence and behavior can be analyzed to a high degree of accuracy. Analytic demonstrations for SCgPC and HDMR have provided insight regarding when these methods converge quickly, and when they fail to perform better than traditional Monte Carlo (MC). In practice, however, the models on which we seek to apply advanced uncertainty quantification methods are seldom analytic. The response is often solved using nonlinear, iterative methods, and often involves many different physics in a single calculation.

In the next several chapters, we explore application of advanced uncertainty quantification methods to engineering-scale performance codes. In this chapter, we consider a well-understood problem with relatively few physics, but without an analytic solution. For this demonstration, we make use of a neutronics problem. Neutronics deals with the population and transport of neutrons through some geometry of materials using the Boltzmann neutron transport equation [60]. This model is selected for demonstration because it is more complex than the analytic models, in that there is not a simple analytic expression for the statistical moments of the responses. However, the model is still a single integro differential equation, and contains no direct nonlinearities in its input terms. This allows some clear analysis despite lack of analyticity.

7.1.1 Neutron Transport

Neutronics is a branch of nuclear engineering that involves the study, simulation, and design of neutral particle transport, particularly free neutron transport. The term *free neutron* describes a neutron that is not bound to a particular nucleus and moves about with some energy. The fundamental science behind nuclear engineering is the fission event, where a free neutron interacts with an atomic nucleus in a way that produces new free neutrons as well as high-energy fission fragments. The released energy can be converted to a variety of uses, most particularly the generation of electrical power.

In order to perform neutronics calculations, we desire to determine the number of neutrons traveling through a plane in the domain in a given period of time. The field variable used to describe this quantity is the *angular flux* $\psi(\mathbf{r}, E, \hat{\mathbf{\Omega}}, t)$, which should not be confused with ϕ used to describe polynomials in a gPC expansion. Angular flux generally is dependent on where in the domain it is quantified, the energy of neutrons considered, the direction of neutron travel, and time.

The controlling equation that approximates neutron transport is the Boltzmann transport equation [57] The Boltzmann equation is used to develop a balance equation for neutron conservation as in Eq. 7.1. In this equation, the time rate of change in the angular flux is balance on the left hand side by removal terms and on the right by source terms. Removal terms include neutrons traveling out of a differential area $d\diagdown$, and neutrons interacting with nuclei which potentially changes their position, energy, direction of travel, or free state. Source terms include new neutrons produced because of fission events, neutrons scattering from other directions of travel or energies into the direction of travel and energy of consideration, new neutrons emitted because of delayed precursors (or post-fission radioactive nuclei emitting neutrons), and other unspecified sources such as neutrons entering from outside the domain.

$$\begin{aligned}\left(\frac{1}{v(E)}\frac{\partial}{\partial t} + \hat{\mathbf{\Omega}}\cdot\nabla + \Sigma_t(\mathbf{r}, E, t)\right)\psi(\mathbf{r}, E, \hat{\mathbf{\Omega}}, t) = \frac{\chi_p(E)}{4\pi}\int_0^\infty \nu\Sigma_f(\mathbf{r}, E', t)\phi(\mathbf{r}, E', t)\ dE' \\ + \int_{4\pi}\int_0^\infty \Sigma_s(\mathbf{r}, E'\to E, \hat{\mathbf{\Omega}}'\to\hat{\mathbf{\Omega}}, t)\psi(\mathbf{r}, E', \hat{\mathbf{\Omega}}', t)\ dE'\ d\Omega' \\ + \sum_{i=1}^{I}\frac{\chi_{d,i}(E)}{4\pi}\lambda_i C_i(\mathbf{r}, t) \\ + s(\mathbf{r}, E, \hat{\mathbf{\Omega}}, t),\end{aligned} \tag{7.1}$$

where

- $\mathbf{r}$ is location in three-dimensional space,

- E is neutron energy,
- $\hat{\mathbf{\Omega}}$ is a unit vector solid angle parallel to neutron velocity,
- t is time,
- $v(E)$ is the magnitude of the neutron velocity,
- $\psi(\mathbf{r}, E, \hat{\mathbf{\Omega}}, t)$ is the *angular flux*,
- $\phi(\mathbf{r}, E, t)$ is the integral of ψ over angle (*scalar flux*),
- ν is the average number of neutrons produced per fission,
- $\chi_p(E)$ is the probability distribution function for neutrons produced by fission,
- $\chi_{d,i}(E)$ is the probability distribution function for neutrons with energy E produced by delayed neutron precursors,
- $\Sigma_t(\mathbf{r}, E, t)$ is the macroscopic total interaction cross section,
- $\Sigma_f(\mathbf{r}, E', t)$ is the macroscopic fission cross section,
- $\Sigma_s(\mathbf{r}, E' \to E, \hat{\mathbf{\Omega}}' \to \hat{\mathbf{\Omega}}, t)$ is the macroscopic scattering cross section for neutrons scattering from energy E' to energy E and from solid angle trajectory $\hat{\mathbf{\Omega}}'$ to $\hat{\mathbf{\Omega}}$,
- I is the number of delayed neutron precursors,
- λ_i is the decay constant for precursor i,
- $C_i(\mathbf{r}, t)$ is the total number of precursor i in $d\mathbf{r}$ at time t,
- $s(\mathbf{r}, E, \hat{\mathbf{\Omega}}, t)$ is an arbitrary source term,

and we assume

- free neutrons have no interaction with other free neutrons,
- total and fission cross sections are assumed angularly independent,
- fission neutrons are emitted isotropically after fission events, and
- delayed neutrons are emitted isotropically after precursor decay.

More details about the physical interpretation of these terms can be obtained in [57], [58], and [60].

One particular application of the neutron transport equation is reactor criticality. Criticality problems are chiefly concerned with the sustainability of a neutron reaction in

fissionable material. Given a particular set of materials in a given geometry, an analyst needs to determine if the number of neutrons is growing over time, diminishing over time, or remaining constant. This analysis is performed by reducing the neutron transport equation (Eq. 7.1) to steady-state operation and introducing the k-eigenvalue,

$$\Big(\hat{\mathbf{\Omega}}\cdot\nabla+\Sigma_t(\mathbf{r},E)\Big)\psi(\mathbf{r},E,\hat{\mathbf{\Omega}})=\frac{1}{k}\frac{\chi_p(E)}{4\pi}\int_0^\infty \nu\Sigma_f(\mathbf{r},E')\phi(\mathbf{r},E')\;dE' + \int_{4\pi}\int_0^\infty \Sigma_s(\mathbf{r},E'\to E,\hat{\mathbf{\Omega}}'\to\hat{\mathbf{\Omega}})\psi(\mathbf{r},E',\hat{\mathbf{\Omega}}')\;dE'\;d\Omega', \tag{7.2}$$

where we have removed the precursors and arbitrary source for this population-balancing problem. The k-eigenvalue is defined such that the following relationship between its value and the reaction sustainability exists:

- If $k>1$, the number of neutrons is growing in time.
- If $k<1$, the number of neutrons is diminishing in time,
- If $k=1$, the reaction is exactly sustained.

For design of commercial nuclear power plants, a k-eigenvalue near unity is often desired to maintain balanced plant operation.

In order to reduce Eq. 7.2 into a form suitable for numerical application, we discretize the energy space E into G distinct *energy groups*. For historical reasons, the energy group with the highest-energy neutrons are labeled with energy group $g=1$, while the lowest energy neutrons are in group $g=G$. This allows integrals to be approximated by sums, and introduces the subscript g to refer to the energy group for which each term applies. This discretization generates G equations coupled through the fission and scattering terms, each with the form

$$\Big(\hat{\mathbf{\Omega}}\cdot\nabla+\Sigma_{t,g}(\mathbf{r})\Big)\psi_g(\mathbf{r},\hat{\mathbf{\Omega}})=\frac{1}{k}\frac{\chi_{p,g}}{4\pi}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\mathbf{r})\phi_{g'}(\mathbf{r}) + \int_{4\pi}\sum_{g'=1}^{G}\Sigma_{s,g'\to g}(\mathbf{r},\hat{\mathbf{\Omega}}'\to\hat{\mathbf{\Omega}})\psi_{g'}(\mathbf{r},\hat{\mathbf{\Omega}}')\;d\Omega', \tag{7.3}$$

where cross sections and material properties are taken at their expected value within the energy group, such as

$$\Sigma_{t,g}\equiv\int_{E_{g+1}}^{E_g}\Sigma_t(\mathbf{r},E')\;dE', \tag{7.4}$$

where E_g is the maximum energy in group g and E_{G+1}=0. By discretizing the energy space, some error is introduced which is dependent on the number of energy groups G. In coarse problems, often the energy space is divided into two energy groups: those neutrons with energy less than or equal to equilibrium thermal neutron energy in the reactor, and those neutrons with greater energy. In more fine calculations, up to hundreds of energy groups can be used [60].

One dimension that still causes difficulty in Eq. 7.3 is the dependence on angular scattering; that is, there is no limit placed on the distribution dependence of scattered neutrons on the incident neutron angle. One approach to this numerical obstacle is to discretize angular space into discrete solid angles. This method is referred to as S_N or discrete ordinates. If the approximation is made that scattering is at most linearly anisotropic, the resulting simplification of Eq. 7.3 is the *diffusion* equation [57],

$$-D_g(\mathbf{r})\nabla^2\phi_g(\mathbf{r}) + \Sigma_{a,g}(\mathbf{r}) = \sum_{g'=1}^{G}\Sigma_{g'\to g}\phi_{g'}(\mathbf{r}) + \frac{\chi_{p,g}}{k}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\mathbf{r})\phi_{g'}(\mathbf{r}), \tag{7.5}$$

where we introduce the diffusion coefficient D,

$$D_g(\mathbf{r}) = \frac{1}{3\Sigma_{t,g}(\mathbf{r})}, \tag{7.6}$$

which comes from Fick's first law of diffusion and relates flux in a diffusive problem to steady-state concentration [58]. Because of integration over angle our field quantity of interest transitions from angular flux $\psi_g(\mathbf{r},\hat{\mathbf{\Omega}})$ to *scalar flux* $\phi_g(\mathbf{r})$, defined by

$$\phi_g(\mathbf{r}) \equiv \int_{4\pi} \psi_g(\mathbf{r},\hat{\mathbf{\Omega}})\, d\hat{\mathbf{\Omega}}, \tag{7.7}$$

where we introduce the macroscopic absorption cross section $\Sigma_{a,g}(\mathbf{r})$. Eq. 7.5 is well-suited to solution by numerical means by discretizing the domain mesh and applying a solution scheme such as finite elements. In some simple geometries Eq. 7.5 has analytic solutions.

We note that this diffusion approximation is most accurate when the dominant physics is isotropic neutron scattering. As a result, it performs poorly near strong absorbers, transition boundaries between materials, and on the boundaries of the problem. As long as the problem has large scattering cross sections and the domain is large with respect to the neutron mean free path, however, it is a reasonable approximation to the full neutron transport equation, and is much more conducive to numerical solution.

7.2 Problem

The problem we apply the diffusion equation and our advanced uncertainty quantification techniques is a benchmark for mixed-oxide (MOX) fuel assemblies in a pressurized water reactor. This benchmark is commonly referred to as C5G7 [61], and specifications are available for both two-dimensional and three-dimensional geometries. In our case, we consider the two-dimensional specifications.

7.2.1 Physical Problem

The geometry of this problem consists of a quarter-symmetric miniature core with four assemblies surrounded by a reflector. The geometry can be seen in Figure 7.1. In this figure, reflecting boundary conditions are imposed on the left and bottom, while vacuum boundaries are applied at the top and right. The mesh is fine triagonal elements, as shown for fuel elements in Figure 7.2, with a coarser mesh in the moderator. In Figure 7.1 we see four assemblies in a two-by-two array surrounded by a reflector. Two example scalar flux profiles from the reference input realization can be seen in Figs. 7.3 and 7.4. Seven materials make up the domain:

- UO2, the fuel for the bottom-left and upper-right assemblies;
- 4.3% enriched MOX fuel, the outer fuel for the bottom-right and upper-left assemblies;
- 7.0% enriched MOX fuel, the next-to-outer fuel for the MOX assemblies;
- 8.7% enriched fuel, the innermost fuel for the MOX assemblies;
- Guide tubes, interspersed points within the innermost fuel in all assemblies;
- Fission chambers, the central point in each assembly; and
- Reflector, the material outside the assemblies.

Each assembly is comprised of 17 by 17 square pin cells. Each pin cell is un-homogenized; that is, a pin cell is comprised of either a fuel pin, guide tube, or fission chamber surrounded by moderator. The pin cell is 1.26 centimeters on a side, while the radius of the fuel pin, guide tube, or fission chamber has a radius of 0.54 centimeters. The cladding and gap for fuel pins are homogenized into the fuel pin itself.

We discretize the energy space for this problem into eight energy groups. The discrete energy group boundaries are given in Table 7.1 [61].

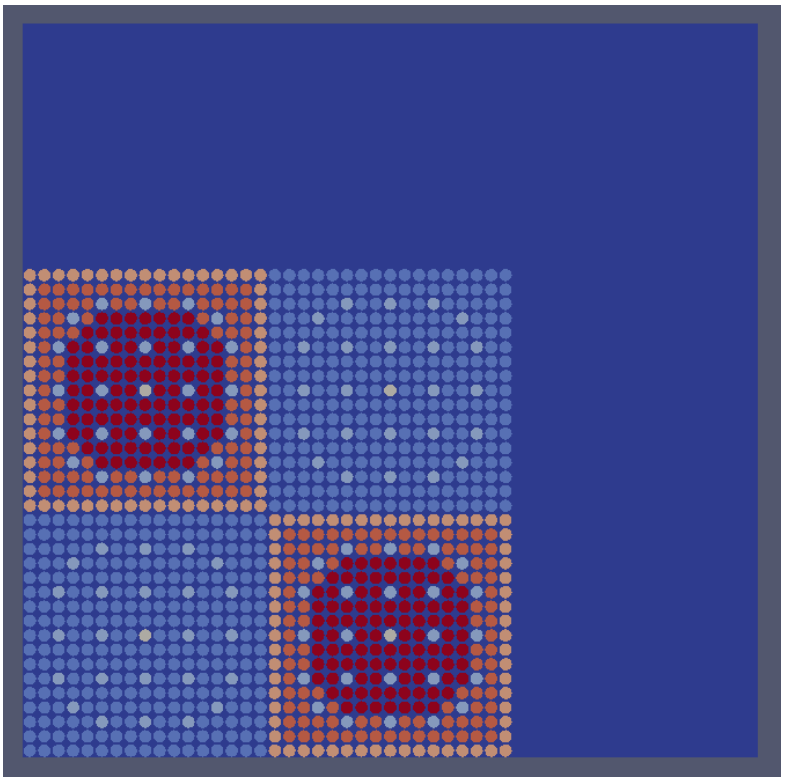

FIGURE 7.1: C5G7 Geometry

Group	Upper Energy Bound
7	0.02 eV
6	0.1 eV
5	0.625 eV
4	3 eV
3	500 keV
2	1 MeV
1	20 MeV

TABLE 7.1: C5G7 Energy Groups

This problem is solved using `MOOSE`-based application `RATTLESNAKE` [24] using the diffusion equation as the driving physics. `MOOSE`-based applications use continuous finite elements for mesh interpolation. The flux field parameter is set to use first-order Lagrange polynomials to interpret the finite element space. For this problem the `RATTLESNAKE` solver parallelizes quite effectively up to six parallel processes per evaluation. On the Idaho National Laboratory supercomputing framework `FALCON` each run takes approximately one minute to converge using `MOOSE`'s preconditioned Jacobian-free Newton Krylov solver [22] when parallelized with six processes.

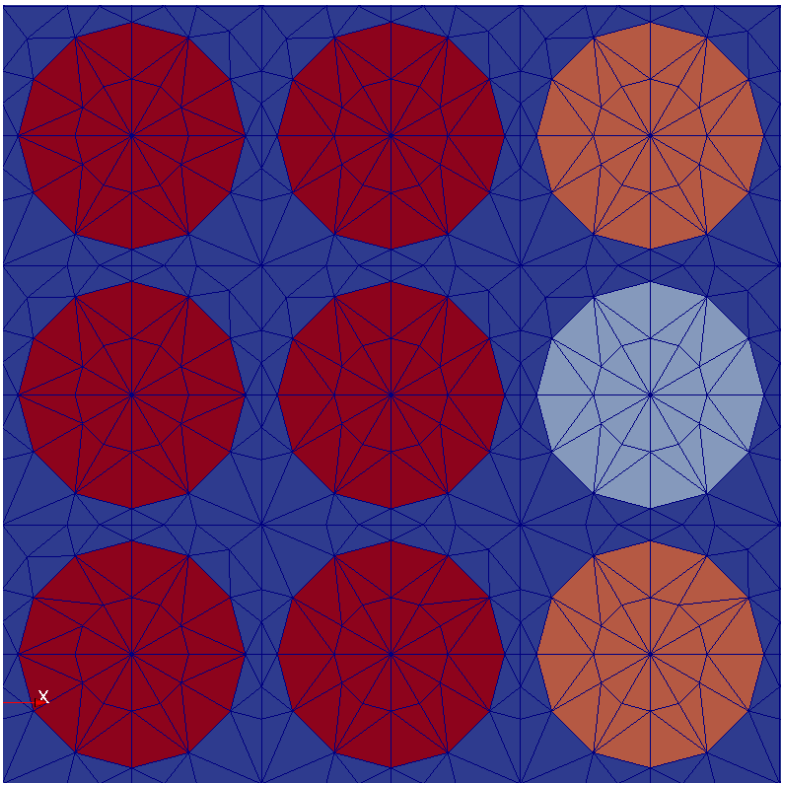

FIGURE 7.2: Partial C5G7 Mesh

7.2.2 Uncertainty

There are a total of 168 correlated macroscopic cross sections as uncertain inputs to this problem. To introduce uncertainty in these cross sections, we use the nominal reference case cross sections as the mean values of Gaussian normal distributions, and assign a standard deviation of five percent of the mean to all inputs. We also assign correlations between cross sections of the same type and material but different energy groups, as well as correlations between cross sections of the same energy group and material but different types. The correlation we assign in either case ten percent. This results in a symmetric covariance matrix that is sparse with blocks of nonzero entries. This approximate covariance matrix could be improved by using uncertainty propagation on a cross section generation tool to calculate actual covariances; however, for demonstration purposes, we make use of these assigned values.

We consider three responses for this model: the k-eigenvalue, and the thermal and fast ($g = 1, 5$) scalar flux measured in the center of the bottom-left element in the mesh, which is near to the center of the reactor. In order to provide an orthogonal and reasonably-sized uncertainty space, we first use `RAVEN` [2] to perform a Karhunen-Loevre (KL) [41] expansion. This results in a surrogate input space made up of orthogonal, standard normally-distributed variables. We refer to the collection of surrogate input variables

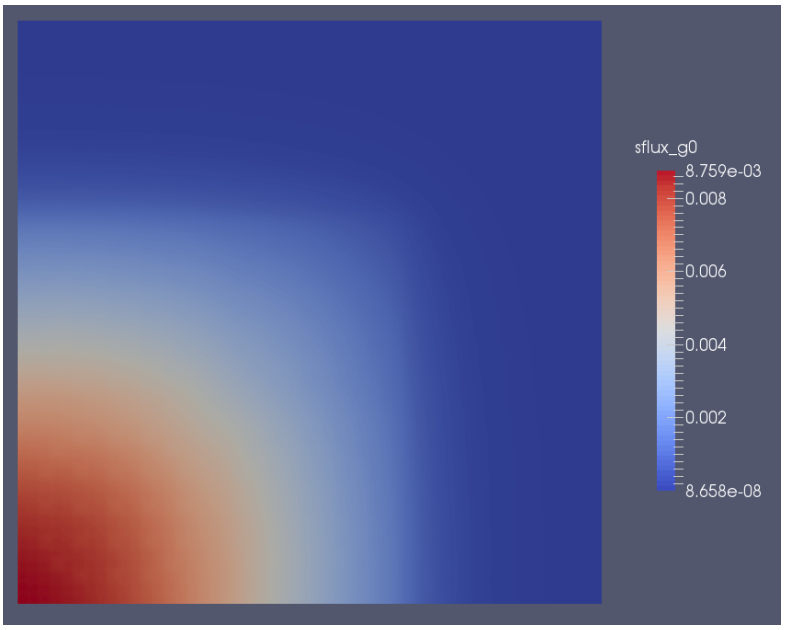

FIGURE 7.3: C5G7 Group 1 (Fast) Flux

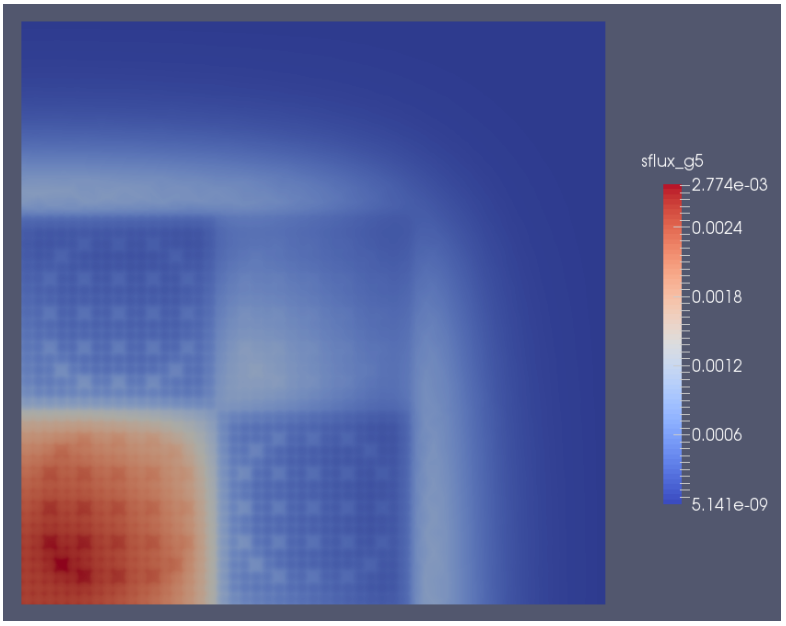

FIGURE 7.4: C5G7 Group 5 (Thermal) Flux

as *latent* variables, which are labeled only by their ranking in the KL expansion. For example, `latent_6` is the sixth dimension in the KL expansion. The original input space we designate the *manifest* input space, as these are the inputs manifested to the simulation model. This follows the naming convention in `RAVEN`.

We then perform a sensitivity analysis of the responses to the latent inputs using ten thousand Monte Carlo samples. The sensitivity values and KL eigenvalues are used together to construct an importance index, ranking the impact of each latent variable on both the input and response spaces. Importance rank is normalized to one, giving

	k-eigenvalue		Center Flux, $g = 1$		Center Flux, $g = 5$	
Rank	Dimension	Importance	Dimension	Importance	Dimension	Importance
1	24	0.09606	24	0.07231	24	0.07032
2	9	0.08555	9	0.06472	9	0.06648
3	0	0.06861	0	0.04856	100	0.06474
4	17	0.04737	116	0.03472	13	0.03396
5	23	0.03415	17	0.06470	0	0.03092
5	158	0.03047	10	0.02726	17	0.02716
6	164	0.02852	8	0.02468	10	0.02651
7	50	0.02695	164	0.02174	118	0.02600
7	6	0.02315	20	0.02157	117	0.02420

TABLE 7.2: C5G7 Importance Ranking

roughly an idea of the percent of the model retained due to truncation. The KL expansion and sensitivity ranking, along with importance determination, are utilities we use in `RAVEN`'s toolkit [42].

The first several importance-ranked eigenvalues for each response are shown in Table 7.2. There are many latent input dimensions common in the first several importance rankings for each variable; in particular, latent dimensions 24, 9, 0, and 17 are common to all three, while additionally 10 is common to both the flux terms. We elect to truncate the latent input space to include these common terms plus dimension 116 (important to the group 1 flux) and dimensions 100 and 13 (important to the group 5 flux). In total this gives our reduced input space a dimensionality of eight. This reduction is significant; we only keep 18 percent of the k-eigenvalue importance, and 16 percent of the two flux importance.

Figures 7.5 through 7.7 show the importance rank of each response. The red cross series corresponding to the left y-axis is the importance of each rank as a function of the ranking itself, as is shown in Table 7.2. The blue circle series corresponding to the right y-axis is the cumulative importance rank as a function of the number of dimensions kept in order of rank. A black line has been added to indicate at which level each series was truncated. The flux responses were both truncated at the end of the steep drop in importance rank values, while the eigenvalue was truncated somewhat more than that. Despite this steep truncation, we see much of the original response preserved.

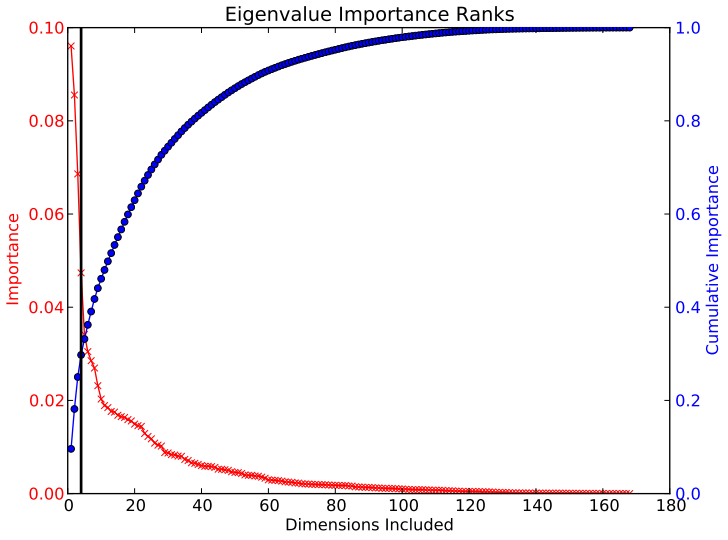

FIGURE 7.5: C5G7 Eigenvalue Importance Ranking

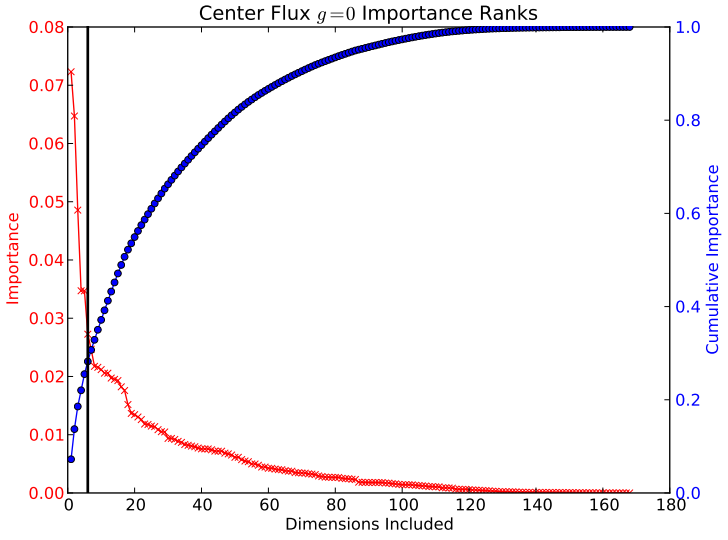

FIGURE 7.6: C5G7 Center Flux ($g = 1$) Importance Ranking

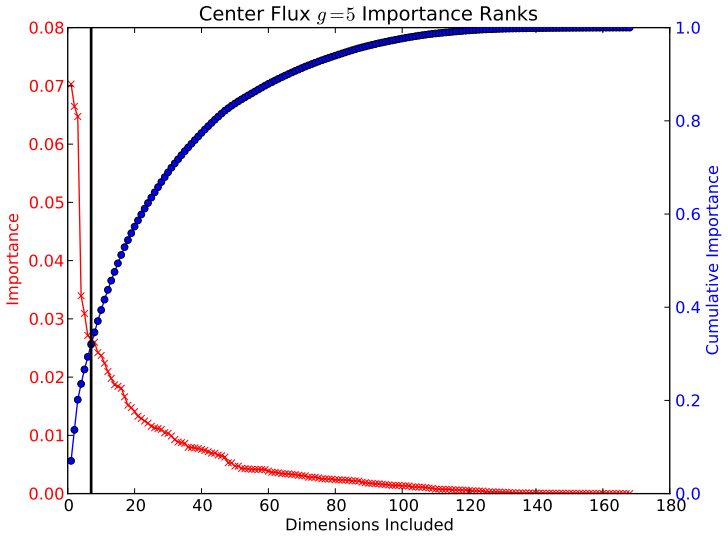

FIGURE 7.7: C5G7 Center Flux ($g = 5$) Importance Ranking

The agreement for the mean and standard deviation of the original full input space and the truncated latent space are shown in Figures 7.8 through 7.13. In each of these figures, the plot on the left shows the process of ten thousand Monte Carlo samples on either the mean or the standard deviation. The blue square series is MC on the original, untruncated space, while the red crosses is MC on the truncated space. The plot on the right shows the agreement level at ten thousand Monte Carlo runs. For the nearly-matching means, the plots are cropped near the calculated values. For the poorly-matching standard deviation, we scale the range to demonstrate the portion of the standard deviation retained. Error bars on both the left and the right plots are given using the mean-based approximation described in section 4.1.

For all three responses the agreement on the mean values has significant overlap, suggesting the mean was largely uncompromised by the input space truncation. The standard deviations, however, do not match as closely. For the two flux responses, roughly one quarter of the value was lost due to truncation, while for the eigenvalue response approximately one fifth of the standard deviation value was lost. This loss is expected, as the importance eigenvalues in Table 7.2 drop off somewhat slowly after the truncation performed. With another 12 inputs retained, it is possible to reduce the error in the k-eigenvalue truncation from roughly 25 percent to nearer 10 percent. However, this truncation is suitable for demonstration.

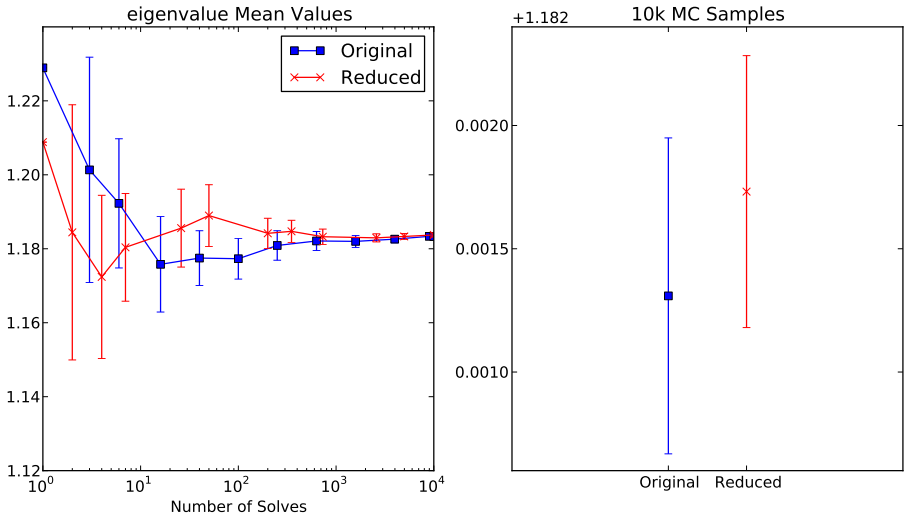

FIGURE 7.8: C5G7 Eigenvalue Input Reduction, Mean

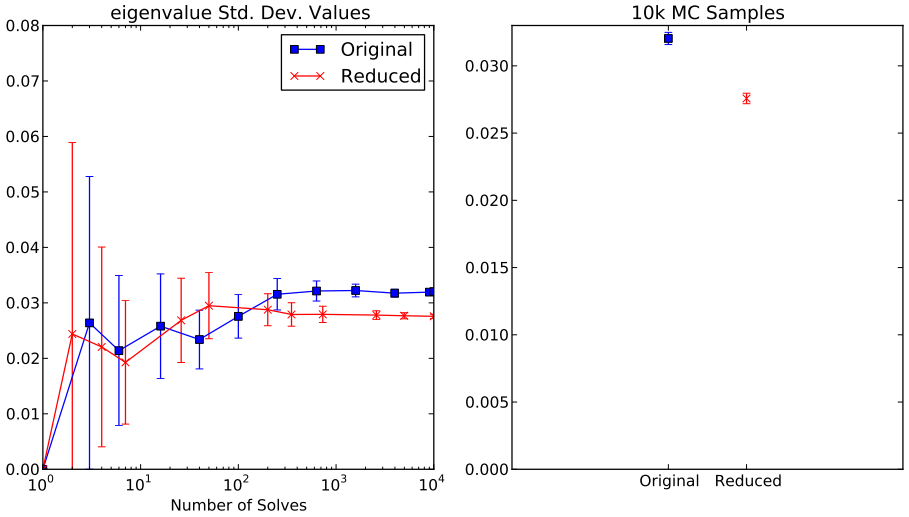

FIGURE 7.9: C5G7 Eigenvalue Input Reduction, Std. Dev.

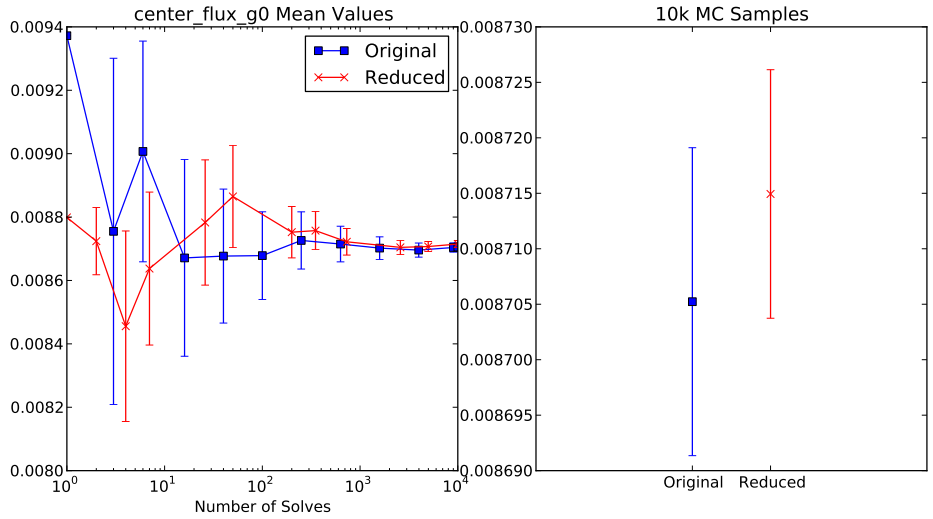

FIGURE 7.10: C5G7 Center Flux $g = 1$ Input Reduction, Mean

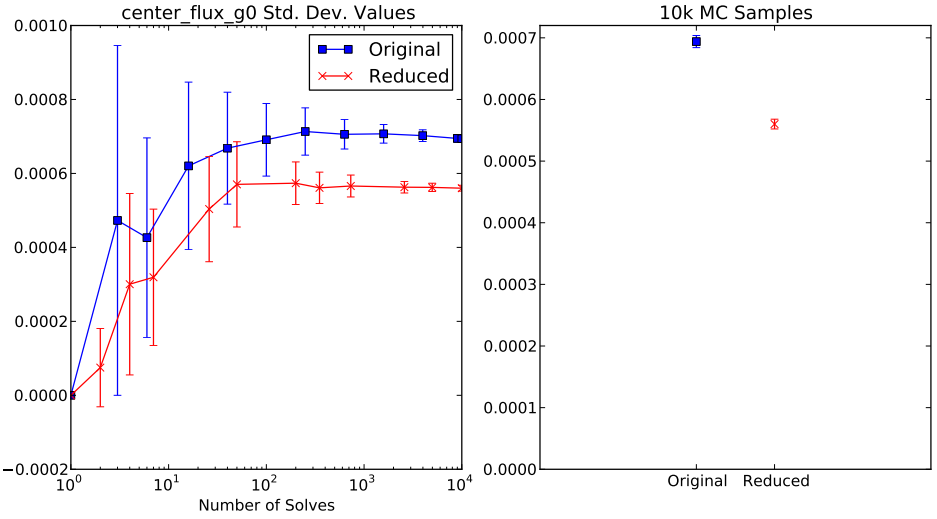

FIGURE 7.11: C5G7 Center Flux $g = 1$ Input Reduction, Std. Dev.

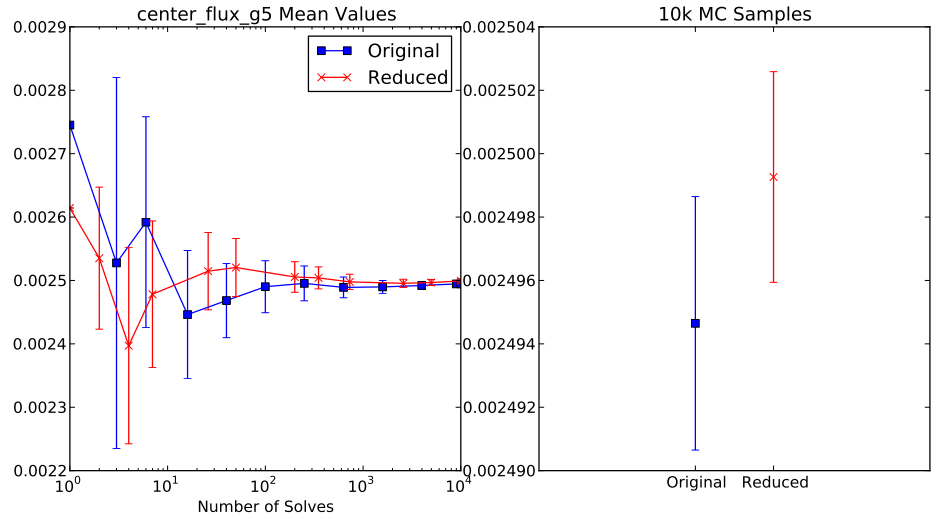

FIGURE 7.12: C5G7 Center Flux $g = 5$ Input Reduction, Mean

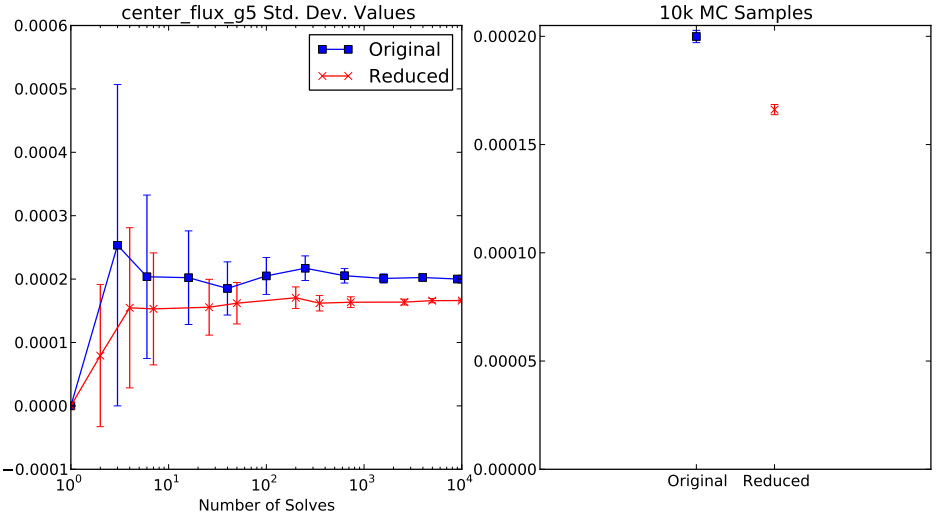

FIGURE 7.13: C5G7 Center Flux $g = 5$ Input Reduction, Std. Dev.

Since the efforts in this demonstration are to establish the effectiveness of collocation-based methods for this model, we consider agreement between collocation expansions and the reduced-space Monte Carlo statistics only. We ignore the original model statistics and instead consider agreement with the truncated input space. Comparison to the original model can be done by comparing to the reduced case MC, then comparing reduced space MC with the original model as we have shown here.

7.3 Results

Todo, intro to results

7.3.1 k-Eigenvalue

As can be seen in Figures TODO, even linear polynomials are sufficient to quite accurately capture the first two statistical moments of the k-eigenvalue for this model. This suggests a strongly linear dependence of k on cross sections, which can be justified through analyzing the first derivatives Eq. 7.5 with respect to each cross section independently. This means that any of the collocation methods are very well-suited to represent the original model, and a cost of far less computational solves than traditional Monte Carlo.

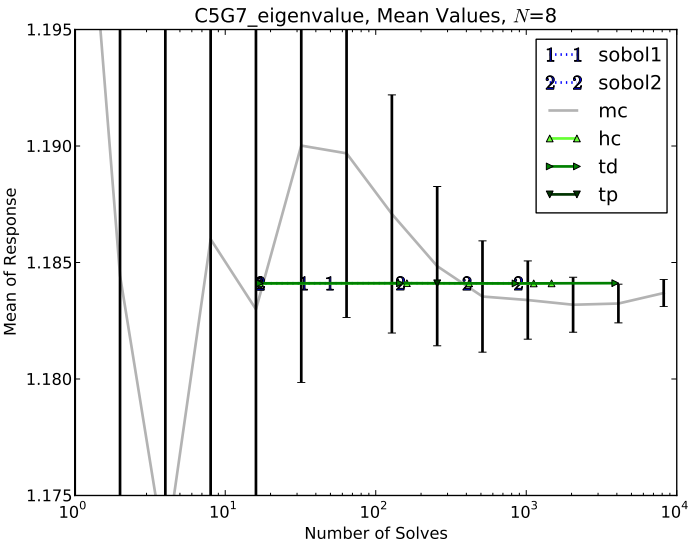

FIGURE 7.14: C5G7 k-Eigenvalue Mean Values

7.3.2 Center Flux, $g = 1$

Todo.

7.3.3 Center Flux, $g = 5$

Todo.

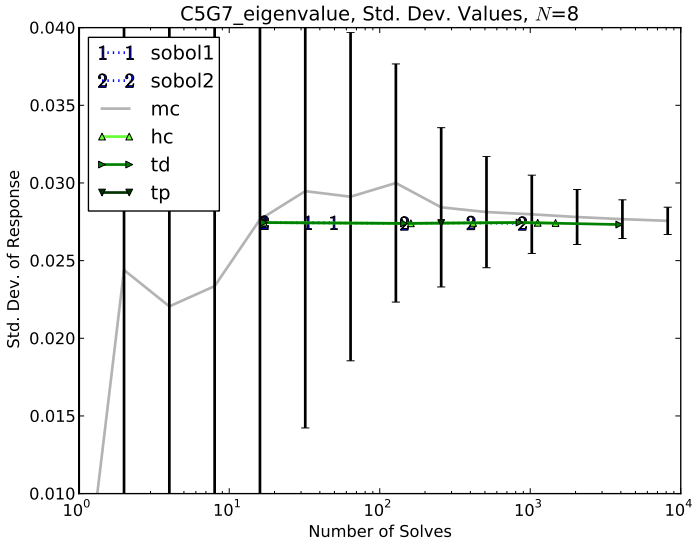

FIGURE 7.15: C5G7 k-Eigenvalue Std. Dev. Values

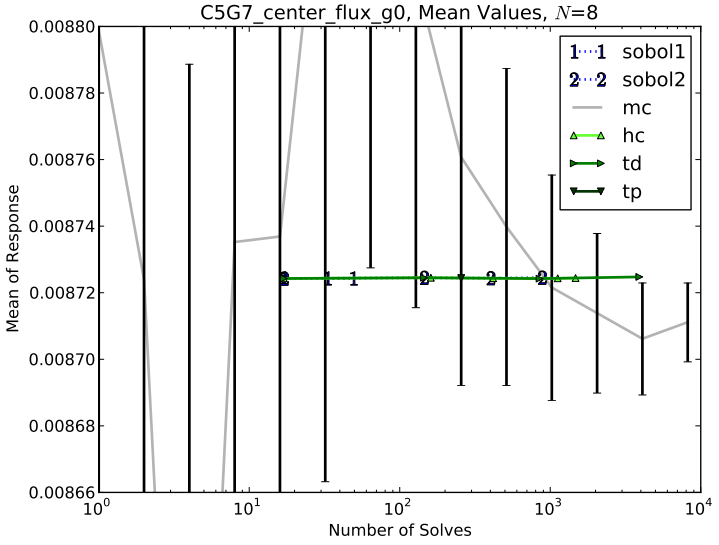

FIGURE 7.16: C5G7 Center Flux $g = 1$ Mean Values

7.4 Conclusion

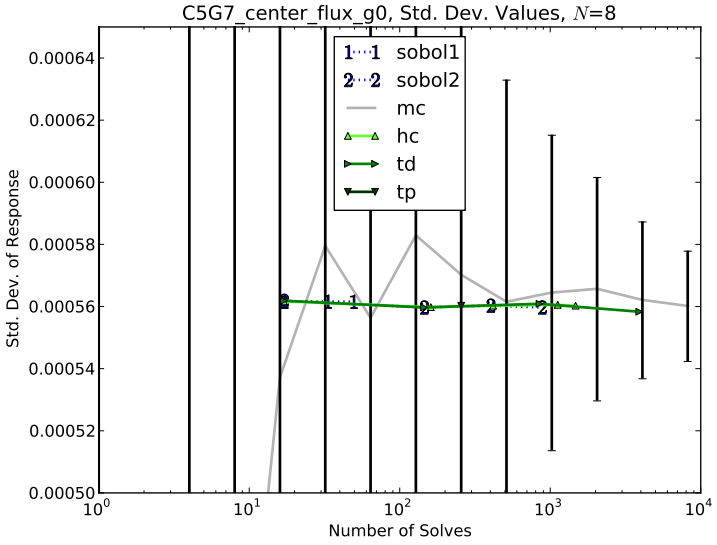

FIGURE 7.17: C5G7 Center Flux $g = 1$ Std. Dev. Values

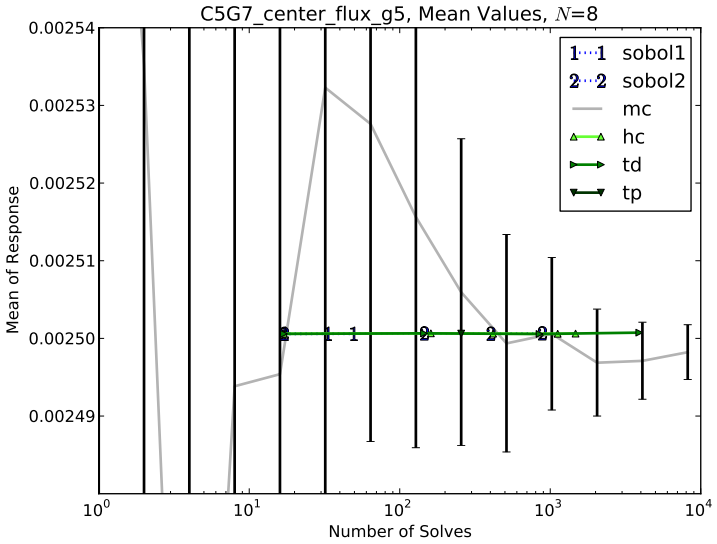

FIGURE 7.18: C5G7 Center Flux $g = 5$ Mean Values

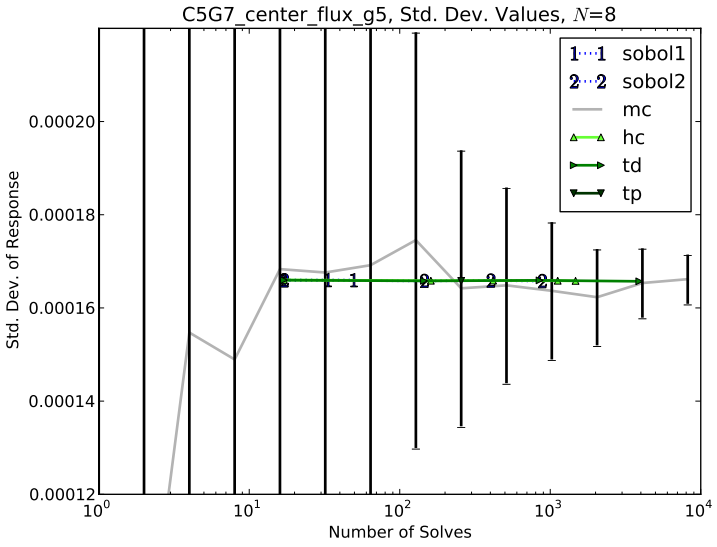

FIGURE 7.19: C5G7 Center Flux $g = 5$ Std. Dev. Values

Chapter 8

SCgPC and HDMR for Fuel Pin Cell

8.1 Introduction

While analytic models provide insight to the operation of stochastic collocation for generalized polynomial chaos and high-density model reduction methods, we are chiefly interested in applying these methods to engineering applications that lead to decision making in real-world activities. To this end, we selected multiphysics simulation code `MAMMOTH`, a `MOOSE`-based *MultiApp* that couples neutronics code `RATTLESNAKE`, fuel performance code `BISON`, and thermal hydraulics code `relap-7`. `MAMMOTH` solves these three sets of physics nonlinearly using picard iterations to feed back values until convergence is achieved.

8.2 Problem

The model we selected is a two-dimensional slice of a pressurized water reactor fuel pin, including the fuel, gap, clad, and moderator. The fuel is separated into 23 axial rings, each with the same material properties but independent input uncertainty. The response value of interest is the neutronics multiplication factor k-effective. The model is documented in [55].

TODO how many groups, etc. Or should that go in the MODELS chapter?

The uncertain inputs are group-, burnup-, and temperature-dependent cross sections, as well as three `BISON` inputs: fuel thermal expansion coefficient, fuel conductivity,

App	Git Version Hash
MOOSE	1fea13a34357a56c6fd049a239e57d597b1c277e
BISON	5552eca741fa30be0efdefd35fecd954b47c9586
RATTLESNAKE	2c892fad29ed7d1f7fb9833116d2b718f7b72055
MAMMOTH	be676b5974f990a0d2a7589ab2a2e58163a47b22
RAVEN	8f7c477740a8277c536d9bd6734614615a8b5cb7

TABLE 8.1: Application Versions Used

and clad conductivity. The macroscopic cross sections are calculated using Scale [56] along with a covariance matrix which provides the correlation between cross sections. Karhunen-Loevre [41] is used to decouple the resulting 671 inputs and reduce them to 20 representative independent inputs. This reduction is informed using RAVEN algorithms considering both the KL expansion as well as input-output sensitivity, together referred to as the *importance rank*. The cutoff was selected to ... TODO to MODELS chapter?

The simulation only considers neutronics (RATTLESNAKE) and fuel performance (BISON), so the thermal hydraulics is neglected. The feedback from RATTLESNAKE to BISON is the power shape, and the feedback from BISON to RATTLESNAKE is the temperature, which in turn affects the cross sections. For performance, the number of Picard iterations between the separate models is limited to 6 per time step.

TODO time steps, other input parameters, input file in appendix?, commit of MAMMOTH used

MOOSE and its applications including RATTLESNAKE, BISON, and MAMMOTH do not generally have a versioning system or release schedule; instead, it is tracked by Git [54] commit hashes. This computation was performed with the application versions listed in Table 8.1. There is nothing particularly special about these versions, except that they were concurrent and compatible at the time calculations were begun.

8.3 Limitations

During the collection of data, it was discovered that the performance of BISON can fluctuate depending on the way is is parallelized. There were instances where BISON would fail to converge, but report an unconverged temperature as a converged solution. As a result, there is artifical numerical error that is difficult to track or account for. Regardless, we demonstrate the performance of various methods on this model, as this behavior indicates true simulation behavior.

Method	Degree	Runs
Total Degree	1	47
Total Degree	2	1105
Total Degree*	3	17389
HDMR (1)	1	47
HDMR (1)	2	47
HDMR (1)	3	93
HDMR (1)	4	93
HDMR (1)	5	139
HDMR (2)	1	47
HDMR (2)	2	1105
HDMR (2)†	3	3221
HDMR (2)†	4	7361
HDMR (2)*	5	13571

TABLE 8.2: Evaluations Required for 23 Input Pin Cell Model

8.4 Results

Figures 8.1 and 8.3 show the values obtained for the mean and variance of this model for a selection of uncertainty quantification methods, including traditional Monte Carlo, static stochastic collocation for generalized polynomial chaos expansion using the total degree polynomial construction indices, first- and second-order static high-density model reduction (HDMR) and adaptive HDMR using both adaptive cut-HDMR and adaptive generalized polynomial chaos. For additional clarity, we provide graphs centered more especially on the non-Monte Carlo data in Figures 8.2 and 8.4. Because the number of Monte Carlo runs necessary to obtain a well-resolved benchmark is prohibitive, we do not present any error convergence plots for this model.

Table **??** summarizes the number of calculations required for each collocation method. Entries marked with a † indicate results that were not obtained because of `MAMMOTH` simulations that failed to converge. Entries marked with a $*$ indicate results that were not attempted because of the number of samples required. We note that in Table 8.2 for first-order static HDMR method successive runs have the same number of evaluations required despite constructing higher-order polynomials. This is because we enforced a floor function for quadrature, requiring a minimum number of quadrature points for a polynomial despite its low order. This artificially increases the points for odd-numbered sets for this particular method, but prevents abnormally poor integration.

We note that for both the mean and the variance, the collocation-based methods all converge within the estimated Monte Carlo value single-standard deviation band with only first-order results. This suggests a high degree of linearity in the response. In both the mean and the standard deviation, there appears to be some convergence towards

increasing the magnitude from the first-order expansions, but without a near-analytic benchmark it is difficult to be certain if this is convergence to the true solution. TODO more comments on this? Comment on how HDMR2 matches TD, and how that suggests linearity in the second-order interactions.

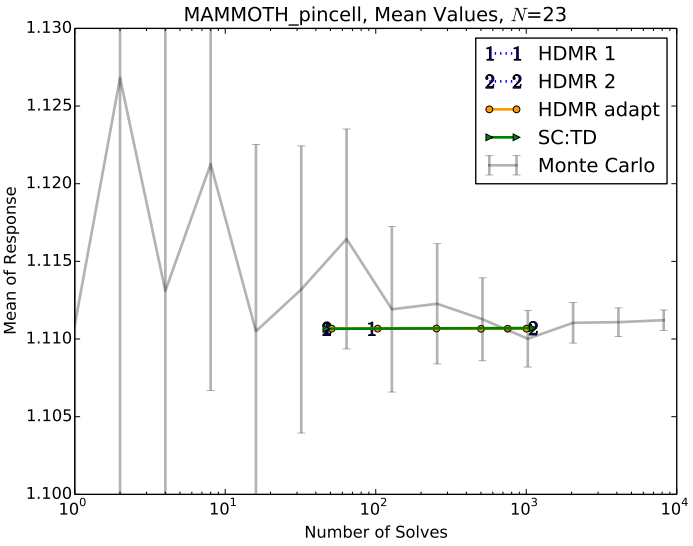

FIGURE 8.1: MAMMOTH Pin Cell, Mean Values

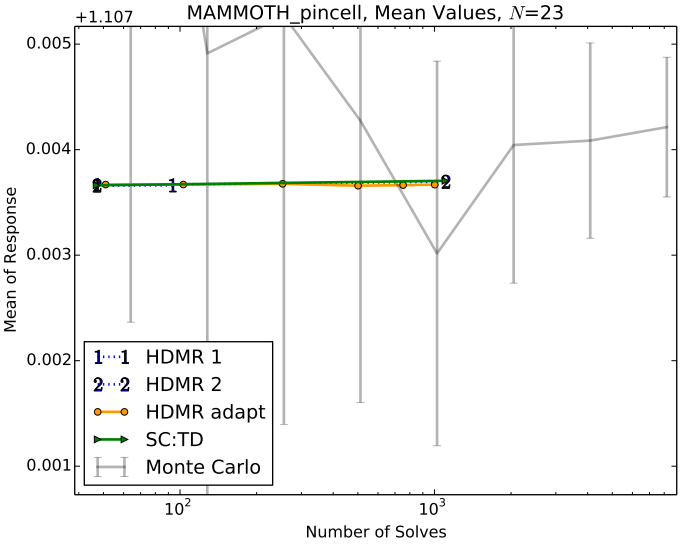

FIGURE 8.2: MAMMOTH Pin Cell, Mean Values (Zoomed)

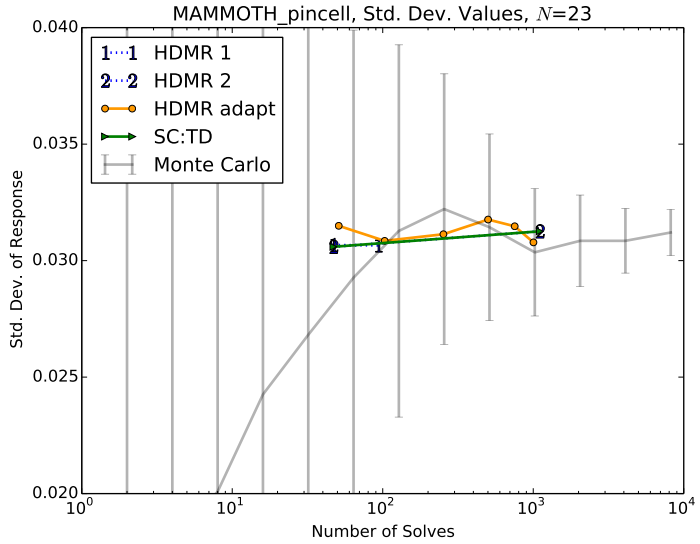

FIGURE 8.3: MAMMOTH Pin Cell, Variance Values

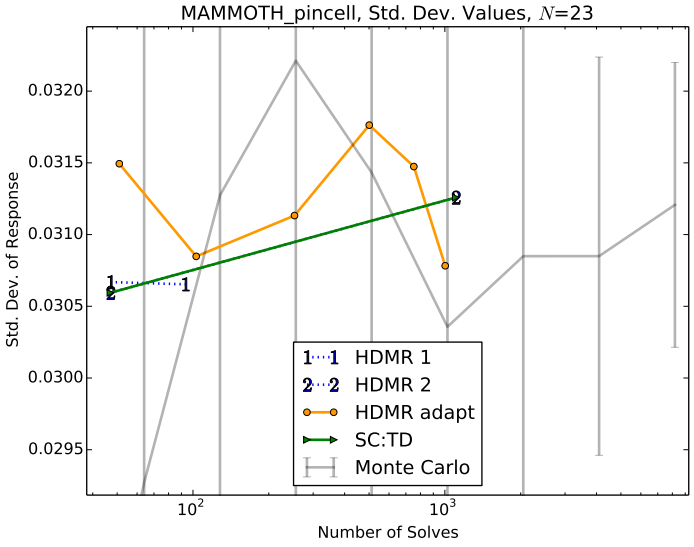

FIGURE 8.4: MAMMOTH Pin Cell, Variance Values (Zoomed)

8.5 Conclusions

While the lack of an analytic benchmark makes it difficult to be certain how much better the collocation-based methods are performing than traditional Monte Carlo, it is clear they are no worse even with first-order approximations. Additionally, with these first-order approximations only requiring roughly 47 evaluations instead of ten thousand, we are prepared to conclude that all the collocation-based methods considered here are more efficient for this model than traditional analog Monte Carlo when it comes to determining second-order statistics.

Chapter 9

SCgPC and HDMR for Time-Dependent Analysis

9.1 Introduction

Up to now in this work we have restricted ourselves to models with a set of single-valued responses. One of the strengths of the `RAVEN` [53] framework is its innate ability to extend reduced-order models (such as the stochastic collocation for generalized polynomial chaos and high-density model reduction expansions) to include time-dependent analysis. `RAVEN` does this by taking snapshots in time and interpolating between them to evaluate the reduced-order model at any time. As part of this work we added the algorithms necessary to do this snapshot-based time-dependent analysis for both the stochastic collocation for generalized polynomial chaos and the high-density model reduction methods. Conveniently, no additional quadrature points are required to perform transient instead of static uncertainty quantification using SCgPC or HDMR methods. It should be noted, however, that adaptive SCgPC and adaptive HDMR are not well-suited to time-dependent analysis because of the plethora of responses; effectively, there is a full set of responses for each snapshot in time, making it difficult for the adaptive algorithms to determine the ideal polynomials to add.

To demonstrate performance of this feature, we sought a time-dependent response with transient behavior. Because none of the analytical models nor the `MAMMOTH` pincell problem have notable transient behavior, we consider a `BISON` [52] simulation of an OECD benchmak [50] where the performance of light-water reactor fuel through several power transients is analyzed. A first pass at uncertainty quantification for this benchmark is performed in [51], and we use the same input variables and uncertainty distributions here.

9.2 Problem Description

The problem considered here is Case 2a defined in Chapter 2 of the benchmark report [50]. It involves several different steady-state fuel behaviors after transitioning power levels for a pressurized water reactor fuel pin. The `BISON` mesh used for this problem is a smeared-pellet mesh, as was also used in [51]. This allows the mesh to be generated directly through the `BISON` input rather than coupling a mesh generation code to `BISON` in `RAVEN`. During the simulation the pellet expands to make contact with the cladding, and modeling persists before and after this phenomenon. The mesh is axisymmetric 2-D in R-Z geometry, using 4290 QUAD8 finite elements or 324 thousand degrees of freedom [51]. Because `RAVEN` couples with `BISON` natively, the input-output processing was handled without any need for user involvement.

The uncertain inputs to this model are given in Table **??**, which is based on the data in [51]. The input parameters are all distributed normally, with the exception of the inlet temperature which was instead distributed uniformly. In addition, there were several dependent inputs in the `BISON` input file that had to be perturbed based on the independent inputs; these are provided in Table 9.2. The responses of interest for this model are the following:

- maximum cladding temperature (`max_clad_surf_temp`),
- percent fission gas released (`fgr_percent`),
- elongation of the cladding, and (`max_clad_creep_strain`)
- maximum creep strain on the cladding (`clad_elongation`).

When we use the term *maximum*, we refer to the maximum value obtained from 13 axial positional along the length of the fuel. There is a separate maximum value for each burnup time step.

9.3 Results

Of particular interest in this problem is analysis of sensitivity coefficients as they devolop in time. As the simulation progresses, there is a shift in the dominant physics behind response values. For example, one of the more dramatic physics transitions occurs as the fuel expands enough to make contact with the cladding. To produce this results, first-order Sobol using first-order polynomials were used. While a more advanced analysis

RAVEN Name	Uncertain Parameter	Mean	Std. Dev.
clad_cond	Clad Thermal Conductivity	16.0	2.5
clad_thick	Cladding Thickness	6.7e-4	8.3e-6
clad_rough	Cladding Roughness	5.0e-7	1.0e-7
creep_rate	Clad Creep Rate	1.0	0.15
fuel_cond	Fuel Thermal Conductivity	1.0	0.05
fuel_dens	Fuel Density	10299.24	51.4962
fuel_exp	Fuel Thermal Expansion	1.0e-5	7.5e-7
fuel_rad	Fuel Pellet Radius	4.7e-3	3.335e-6
fuel_rough	Fuel Pellet Roughness	2.0e-6	1.6667e-7
fuel_swell	Solid Fuel Swelling	5.58e-5	5.77e-6
gas_cond	Gas Conductivity	1.0	0.025
gap_thick	Gap Thickness	9.0e-5	8.33e-6
mass_flux	Mass Flux	3460	57.67
rod_press	Rod Fill Pressure	1.2e6	40000.0
sys_press	System Pressure	1.551e7	51648.3
sys_power	System Power	1.0	0.016667
RAVEN Name	Uncertain Parameter	Lower Bound	Upper Bound
inlet_temp	Inlet Temperature	558.0	564.0

TABLE 9.1: OECD Benchmark Independent Inputs

Raven Name	Bison Path	Calculation
clad_inner	Kernals.heat_source_clad.inner_diameter	2*(fuel_rad + gap_width)
outer_diam_heat	Kernals.heat_source_clad.outer_diameter	2*(fuel_rad + gap_width + clad_thick)
sys_press_cool	CoolantChannel.convective_clad_surface.inlet_pressure	sys_press
outer_diam_cool	CoolantChannel.convective_clad_surface.rod_diameter	2*(fuel_rad + gap_width + clad_thick)
porosity_thermal	Materials.fuel_thermal.intial_porosity	1 - fuel_dens/10980
fuel_diam	Materials.fuel_relocation.diameter	2*fuel_rad
gap_diam	Materials.fuel_relocation.gap	2*gap_thick
porosity_sifgr	Materials.fission_gas_release.initial_porosity	1 - fuel_dens/10980

TABLE 9.2: OECD Benchmark Dependent Inputs

could be performed, we found many of the realizations in the input space needed adjustments that could not realistically be automated in order to be solved by `BISON`, such as preconditioning and executioner parameters. Despite the low order representation, however, there is significant physics demonstrated in the sensitivity results.

Figures 9.3 through 9.6 show the development of Sobol indices over the burnup of the fuel, expressed in percent FIMA (fissions per initial metal atom). In each plot, the power history shape as a function of burnup is superimposed in dotted red, providing insight to some of the sensitivity behaviors. Because of the large number of input parameters, we only show those parameters on each plot that have significant impact on the response considered. For clarity, we use a consistent scheme for coloring and marking throughout the plots, where green is fuel parameters, black is system parameters, blue is gap parameters, and magenta is clad parameters. In each set of parameters, different symbols are used to differentiate the various related inputs. We consider each plot

separately. For reference, we also include the burnup-dependent mean and standard deviation in Figure 9.1 and 9.2 respectively. While we show the max centerline fuel mean and variance shapes, the max cladding temperature follows the same shape. In each figure, the magnitude of the values are scaled for each parameter by a factor shown in the legend, which allows all the shapes to be seen clearly.

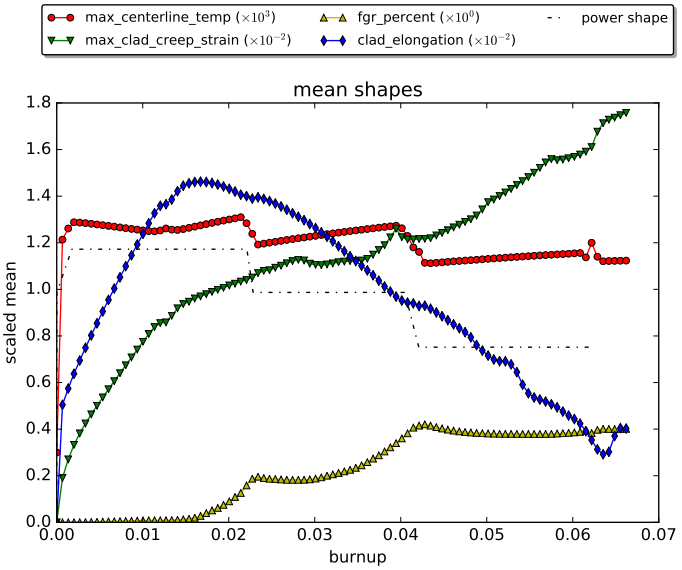

FIGURE 9.1: OECD Response Mean Values over Burnup

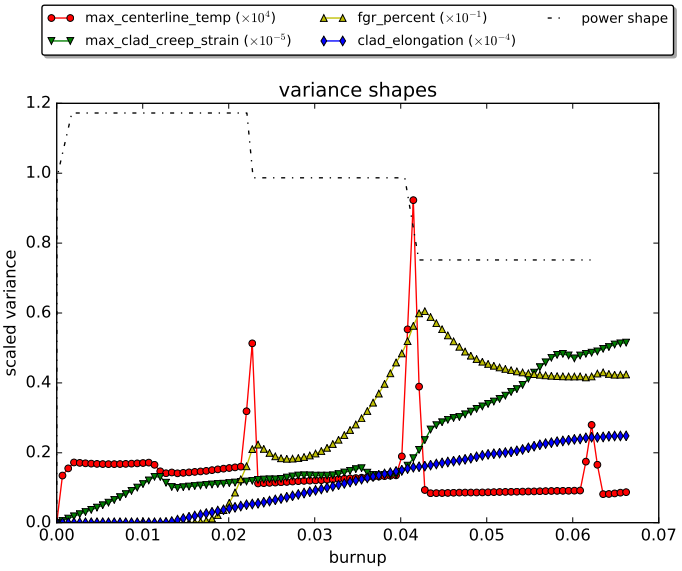

FIGURE 9.2: OECD Response Variance Values over Burnup

9.3.1 Maximum Clad Surface Temperature

One of the key design parameters for fuel in nuclear power plants, the maximum clad surface temperature is used to quantify margin to clad melting points, at which point radioactive fuel and gas could escape into the primary moderator loop. Understanding the behavior of this parameter is key to the design of nuclear fuel.

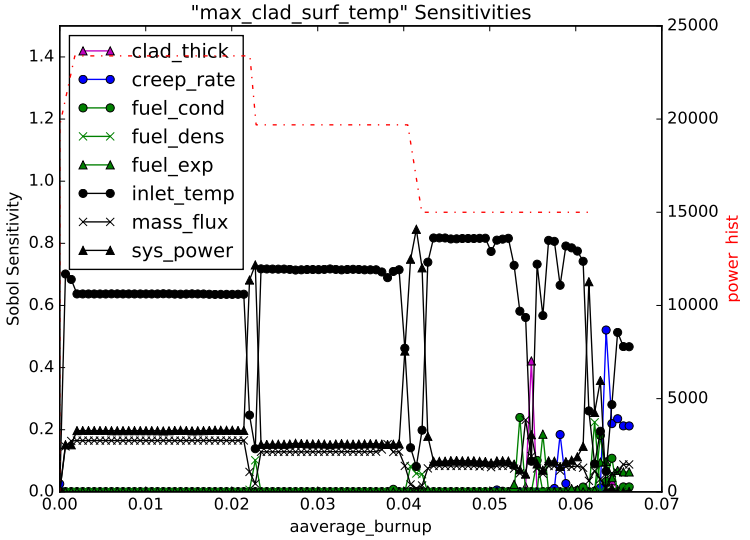

FIGURE 9.3: Maximum Clad Surface Temperature Sensitivities

As can be seen in Figure 9.3, the variance in this parameter is dominated throughout the simulation by the variance in the inlet temperature of the moderator. Immediately around power changes, however, there are spikes where the variance in peak clad temperature is instead dominated for a very short time by the system power, instead. This is reasonable, since the inlet moderator temperature determines the amount of heat that can be transferred out of the clad and into the moderator. However, near changes in the system power, and before steady-state operation is achieved, variance in the system power itself will drive the amount of heat transferred from the fuel to the clad, and dominates the variance in the clad temperature.

In a similar fashion, we see a trade off between two less-impacting variables. The mass flux, or the amount of moderator passing over the cladding, and fuel density share a similar relationship as the inlet temperature and the system power. During steady-state operation the clad temperature is more sensitive to the mass flux, but this exchanges with fuel density near power transients, for the same reasons as the inlet temperature and system power.

Interestingly, we see several new parameters demonstrating impact near the end of the simulation at high burnup. The fuel conductivity, clad thickness, fuel expansion coefficient, and creep rate all exhibit stronger influence toward the end of life, though the inlet temperature continues to be dominant except for a peak around 0.055 FIMA, where there is a transition in physics that emphasizes the clad thickness over other inputs. This is likely because the peak clad temperature reaches quite low values towards the end of its life, as the fuel produces less heat.

9.3.2 Percent Fission Gas Released

During fission events on the atomic scale in the fuel, some of the products are fission gases. These are often radioactive themselves and can escape the confines of the fuel and cladding more easily than pieces of fuel, making them another design concern for mitigating contamination of the moderator.

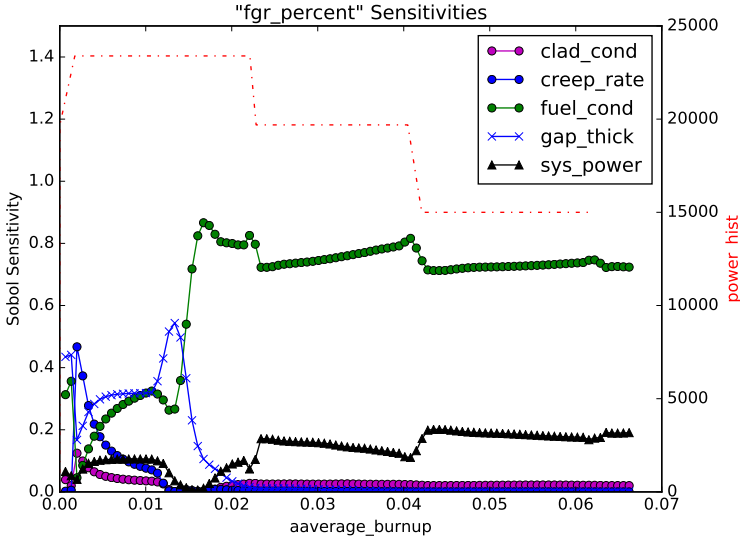

FIGURE 9.4: Percent Fission Gas Released Sensitivities

As can be seen in Figure 9.4, the variance in this parameter is split into two regions, with some effects crossing between the two. The dividing phenomenon appears to be when the fuel has expanded through the gap to contact the cladding, which occurs around 0.015 FIMA. Prior to this, there is an interesting interplay between the sensitivities to gap thickness, the creep rate, and the fuel conductivity, with lesser impact from the system power and clad conductivity. Clearly the ability to remove heat effectively from the fuel has a large impact on how likely fission gas is released. After contact, the variance in the fuel conductivity and system power dominate variance in the fission gas released.

9.3.3 Maximum Cladding Creep Strain

Cladding *creep* describes the physics of pressurized water outside the fuel cladding pressing onto the cladding while the cladding experiences changes in temperature. Clad creep strain is the measure of the strain on the cladding as a result of creep.

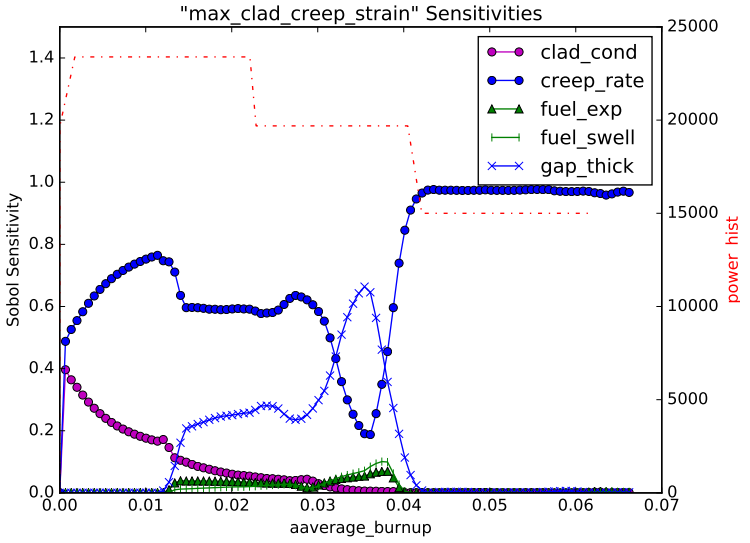

FIGURE 9.5: Maximum Cladding Creep Strain Sensitivities

As can be seen in Figure 9.5, the variance in this parameter has three distinct regions: before fuel-clad contact, high-power contact, and low-power contact. Nearly all the way through the simulation, the variance in the creep strain is unsurprisingly dominated by variance in the creep rate. Once contact is made, however, the gap thickness plays a more important role, and grows in importance until the second drop in power. This is likely because the creep strain is mitigated by the expanding fuel pushing back onto the cladding, and the size of the gap determines how early that strain begins to be relieved.

Also interestingly, the variance in the clad conductivity is initially important, but tails off as physics besides the moderator pressure begin influencing the creep strain. The fuel expansion and fuel swelling parameters, which describe the fuel expansion as a result of both thermal expansion and cracking and expansion due to fission gas buildup, are important after contact and before the second power drop, but insignificant in the first and third sections.

9.3.4 Clad Elongation

Clad elongation measures the changing length of the clad as temperatures and pressures act on it throughout the simulation. As seen in Figure 9.1, the clad tends to elongate

quickly under high power, but tails off as power diminishes.

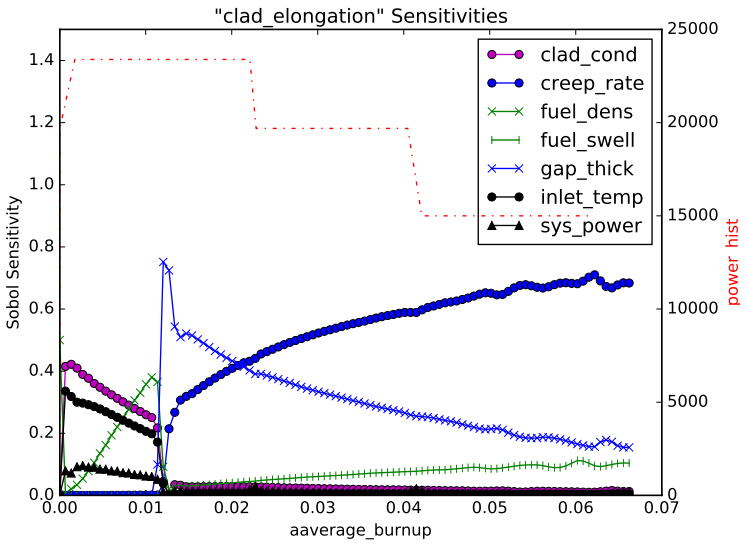

FIGURE 9.6: Clad Elongation Sensitivities

As can be seen in Figure 9.6, the variance in this parameter has two remarkably different sets of physics, one before fuel-clad contact and one after. Before contact, the variance in the elongation is determined by variance in the clad conductivity, inlet temperature, and system power, with growing importance from the fuel density. The first three parameters all deal with the amount of heat the clad is receiving and its ability to remove it, suggesting during this phase thermal expansion is the main source of elongation. As the fuel expands, however, the fuel density becomes important up until contact is made.

After contact, the elongation variance is initially determined almost solely by the gap thickness, which determines when exactly the fuel begins to push on to the clad. This slowly trades places with the creep rate as power levels diminish and the expanding fuel provides less variance than the pressure of the moderator. Interestingly, even as the gap thickness diminishes in importance, the fuel swelling parameter grows during the power reduction, showing how after contact the gap thickness becomes less important than the swelling of the fuel itself.

9.4 Conclusion

Time-dependent sensitivity analysis provides means to better understand uncertainty propagation throughout a transient simulation. As physics shift throughout the simulation, so too does the sensitivity of the response to the input parameters. If this same simulation were performed using only static analysis on time-averaged responses, the

ability to make clear decisions would be reduced because of the lack of time-dependent information. The addition of time-dependent analysis is quite beneficial to analysts considering time-dependent simulations.

Chapter 10

Conclusions

10.1 Introduction

In this work we have explored advanced uncertainty quantification methods and their application to a variety of models. In addition to implementing existing algorithms, new predictive methods for adaptive algorithms have been introduced and demonstrated. We have compared convergence performance to traditional analog Monte Carlo, and observed cases both when collocation-based methods are desirable and when Monte Carlo is preferable. We have also demonstrated performance of these methods on a multiphyics engineering problem, as well as in time-dependent analysis of and OECD benchmark. Here we summarize the obsesrved results and generalize them, as well as discuss limitations discovered during this work.

10.2 Performance Determination

As seen in several analytic models as well as engineering performance models, the convergence rate of both stochastic collocation for generalized polynomial chaos as well as high-density model reduction (using stochastic collocation for subsets) depend primarily on two factors. First, despite many tools to combat the curse of dimensionality, grid-based collocation methods still degrade significantly as the size of the input space increases. For any more than approximately 10 independent inputs, collocation-based methods perform little better than Monte Carlo until many thousands of samples are taken. Second, SCgPC and HDMR perform much better for models with a high level of continuity than discontinuous models. As seen in the Sobol G-Function, the collocation methods have great difficulty representing the absolute value function. However, for

models with high levels of continuity and low dimensionality, collocation methods prove very effective in comparison to traditional Monte Carlo methods.

Between different collocation-based methods, we also see several trends. First, static HDMR methods never outperform their corresponding SCgPC methods; that is, second-order polynomial expansions in SCgPC always match or outperform HDMR methods that are limited to second-order polynomials. This is expected because HDMR at any truncation is a subset of the SCgPC expansion. However, the static HDMR method is still valuable, as even in larger input dimensionality problems some solution can be obtained with few runs. For example, for first-order HDMR using first-order polynomials, only three times the input dimensionality samples are required to obtain a solution. For very costly models, it may not be possible to use SCgPC without HDMR.

Second, we observe for all continuous functions, the total degree polynomial construction method significantly outperforms the hyperbolic cross method, as expected by its design. Since polynomial expansion methods struggle to perform well for discontinuous models anyway, total degree is a good method to use if the response is expected to be smooth.

Finally, we note that in general the adaptive methods seldom completely outperform all the other collocation methods. Because the prediction algorithm is imperfect, there will always be a static choice of polynomials that is more effecient. However, if the nature of the response in polynomial representation is not well-known, the adaptive algorithms can be effective tools in exploring the response polynomial space.

10.3 Limitations Discovered

One limitation that has not yet been discussed is the reliablity of model algorithms. Because many engineering codes are complicated and involve a great number of opttions to assure particular realizations can be solved, they are also often somewhat fragile. Changes in the input space can require changes in other solution options, such as pre-conditioning tools, spatial and temportal step sizes, and so on. The changes required are often not predictable, and if not applied, can result in regular failure to converge a solution. Traditional Monte Carlo methods overcome this issue by rejecting failed points and chosing a new sample. This introduces some bias, but hopefully a small amount relative to the overall sample size. For collocation-based methods, however, re-sampling is not a valid option, and failure to converge results in a failure of the method. In the process of searching for a suitable engineering demonstration model, many problems were considering using a variety of codes; however, after extensive collaboration, it was determined many of these codes accepted as much as a 10% failure rate in random

perturbations of the input space. This failure rate almost surely renders the collocation-based methods unusable. Thus, in addition to considering the dimnsionality of the input space and regularity of the response, the robustness of the algorithms used to solve the model responses must be considered before applying stochastic collocation for generalized polynomial chaos expansion or high-density model reductions methods.

10.4 Future Work

TODO

Appendix A

Quadratures, Polynomials, and Distributions

A.1 Introduction

Thanks to the work of Xiu and Karniadakis [25], many distributions have corresponding polynomials and integrating quadratures that are ideally suited for generalized Polynomial Chaos (gPC) expansion construction. In this appendix, we consider the four continuous distributions with corresponding ideal polynomials and quadratures, as well as an arbitrary case for other distributions. In general, domain transformations must be performed to shape a general distribution into a form that matches the quadrature integration scheme; we discuss those transformations for each distribution here as well. For clarity, we define x as the variable for standard distributions and domains, while y is the variable for generic distributions and domains. Transformation involves the relationship between x and y.

A.1.1 General Syntax

We use the following syntax when describing polynomials, distributions, normalizations, and quadratures.

- x: Argument of standard distribution with specific domain.
- y: Argument of general distribution with generic domain.
- $h(y)$: Generic function with dependence on distributed variable y.
- $\rho(y)$: Probability measure, or the functional part of the probability distribution.

- A: Normalization factor, or scalar part of the probability distributions.
- μ: Distribution mean.
- σ: Distribution standard deviation.

Note that because of the definitions above, for each distribution we require

$$A \int_a^b \rho(y)dy = 1, \tag{A.1}$$

where a and b are the extreme values of the distribution, and might be infinite. We also make use of the two standard functions, the Gamma function (not to be confused with the Gamma distribution),

$$\Gamma(x) = \int_0^\infty z^{x-1} \exp(-z)dz, \tag{A.2}$$

and the Beta function (not to be confused with the Beta distribution),

$$\mathrm{B}(z_1, z_2) = \int_0^1 t^{z_1-1}(1-t)^{z_2-1}dt. \tag{A.3}$$

In each example the quadrature nodes and weights x_i, w_i will use subscripts that relate to the symbol of the associated polynomials, for increased clarity.

A.2 Uniform Distributions and Legendre Polynomials

The uniform distribution is a single value between finite extrema a and b and zero everywhere else so that $x \in [a, b]$. The uniform distribution has the following characteristics.

$$\mu = \frac{a+b}{2}, \tag{A.4}$$

$$\sigma = \frac{b-a}{2} \tag{A.5}$$

$$A = \frac{1}{2\sigma}, \tag{A.6}$$

$$\rho(y) = 1, \tag{A.7}$$

$$1 = \frac{1}{2\sigma} \int_a^b dy. \tag{A.8}$$

Legendre polynomials $P_n(x)$ are defined on the range $x \in [-1, 1]$ with polynomial order $n \in \mathbb{N}$. They are defined by the contour integral [64]

$$P_n(x) = \frac{1}{2\pi i} \oint \left(1 - 2tx + t^2\right)^{-1/2} t^{-n-1} dt, \tag{A.9}$$

and are made orthonormal as

$$\frac{2n-1}{2} \int_{-1}^{1} P_m(x) P_n(x) dx = \delta_{mn}. \tag{A.10}$$

Legendre quadrature approximates the following integrals:

$$\int_{-1}^{1} h(x) d(x) = \sum_{\ell=1}^{\infty} w_\ell h(x_\ell) \tag{A.11}$$

In order to convert a general uniform distribution to share the domain of Legendre polynomials and quadrature, the following conversion is necessary:

$$y = \sigma x + \mu, \tag{A.12}$$

$$x = \frac{y - \mu}{\sigma}. \tag{A.13}$$

As a result, Legendre quadrature integrates arbitrary uniform distributions as

$$\int_a^b h(y) \rho(y) dy = \frac{1}{2} \sum_{\ell=1}^{\infty} w_\ell h(\sigma x_\ell + \mu). \tag{A.14}$$

A.3 Normal Distribution and Hermite Polynomials

The normal distribution is a symmetric, bell-shaped distribution with domain $y \in (-\infty, \infty)$. The normal distribution has the following characteristics.

$$\rho(y) = \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right), \tag{A.15}$$

$$A = \frac{1}{\sigma\sqrt{2\pi}}, \tag{A.16}$$

$$1 = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) dy. \tag{A.17}$$

Hermite polynomials $\mathrm{He}_n(x)$ are defined on the range $x \in (-\infty, \infty)$ with polynomial order $n \in \mathbb{N}$. They can be defined through the contour integral [64]

$$\mathrm{He}_n(x) = \frac{n!}{2\pi i} \oint \exp\left(-t^2 + 2tx\right) t^{-n-1} dt, \tag{A.18}$$

and are made orthonormal as

$$\frac{1}{\sqrt{2\pi} n!} \int_{-\infty}^{\infty} \mathrm{He}_m(x) \mathrm{He}_n(x) \exp\left(\frac{-x^2}{2}\right) dx = \delta_{m,n}. \tag{A.19}$$

Hermite quadrature approximates the following integrals:

$$\int_{-\infty}^{\infty} h(x) \exp\left(\frac{-x^2}{2}\right) dy = \sum_{h=1}^{\infty} w_h h(x_h). \tag{A.20}$$

In order to convert a general normal distribution to share the domain of Hermite polynomials and quadrature, the following conversion is necessary:

$$y = \sigma x + \mu, \tag{A.21}$$

$$x = \frac{y - \mu}{\sigma}, \tag{A.22}$$

As a result, Hermite quadrature integrates arbitrary normal distributions as

$$\int_{-\infty}^{\infty} h(y) \rho(y) dy = \frac{1}{\sqrt{2\pi}} \sum_{h=1}^{\infty} w_h h(\sigma x_h + \mu). \tag{A.23}$$

A.4 Gamma Distribution and Laguerre Polynomials

The Gamma distribution has a finite lower bound a and infinite upper bound so that $y \in [a, \infty)$. This distribution also takes as an argument shape factors α and β. The Gamma distribution has the following characteristics:

$$\rho(y) = y^{\alpha - 1} e^{-\beta y}, \tag{A.24}$$

$$A = \frac{\beta^\alpha}{\Gamma(\alpha)}, \tag{A.25}$$

$$1 = \frac{\beta^\alpha}{\Gamma(\alpha)} \int_a^{\infty} (y - a)^{\alpha - 1} e^{-\beta(y-a)} dy. \tag{A.26}$$

Laguerre polynomials $\mathcal{L}_n(x)^{(\tilde{\alpha})}$ are defined on the range $x \in [0, \infty]$ with polynomial order $n \in \mathbb{N}$. An additional argument $\tilde{\alpha}$ is required to specify the polynomial family.

Note that we use $\tilde{\alpha}$ for the polynomial parameter, and α for the distribution parameter. Laguerre polynomials can be defined through the contour integral [64]

$$\mathcal{L}_n^{(\tilde{\alpha})}(x) = \frac{1}{2\pi i} \oint \frac{1}{(1-t)^{\tilde{\alpha}+1)} t^{n+1}} \exp\left(-\frac{xt}{1-t}\right) dt, \tag{A.27}$$

and are made orthonormal as

$$\frac{n!}{\Gamma(n+\tilde{\alpha}+1)} \int_0^\infty x^\alpha e^{-x} \mathcal{L}_m^{(\tilde{\alpha})} \mathcal{L}_n^{(\tilde{\alpha})} dx = \delta_{mn}. \tag{A.28}$$

Laguerre quadrature approximates the following integrals:

$$\int_0^\infty h(x) x^{\tilde{\alpha}} e^{-x} dx = \sum_{g=1}^{\infty} w_g h(x_g). \tag{A.29}$$

In order to convert a general Gamma distribution to share the domain of Laguerre polynomials and quadrature, the following conversion is necessary:

$$y = \frac{x}{\beta} + L, \tag{A.30}$$

$$x = (y - L)\beta. \tag{A.31}$$

As a result,

$$\int_L^\infty h(y)\rho(y) dy = \frac{1}{(\alpha-1)!} \sum_{g=1}^{\infty} w_g h\left(\frac{x_g}{\beta} + L\right), \tag{A.32}$$

where points and weights are obtained using $\tilde{\alpha} = \alpha - 1$ for Laguerre polynomials and quadrature.

A.5 Beta Distribution and Jacobi Polynomials

The Beta distribution is a flexible distribution with finite range $y \in [a, b]$ and shaping parameters α and β. In the event $\alpha = \beta$, the distribution is symmetric. If $\alpha = \beta = 0$, the uniform distribution is recovered. The Beta distribution has the following characteristics:

$$\rho(y) = y^{\alpha-1}(1-y)^{\beta-1}, \tag{A.33}$$

$$A = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}, \tag{A.34}$$

$$1 = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_a^b y^{\alpha-1}(1-y)^{\beta-1} dy. \tag{A.35}$$

Jacobi polynomials $J_n(x)^{(\tilde{\alpha},\tilde{\beta})}$ are defined on the range $[-1,1]$ with polynomial order $n \in \mathbb{N}$. Two shaping arguments $\tilde{\alpha}$ and $\tilde{\beta}$ are used to uniquely define the polynomial family. Note that we use $\tilde{\alpha}, \tilde{\beta}$ for the polynomial parameters, and α, β for the distribution parameters. Jacobi polynomials can be defined through solution of the recurrence relation as

$$J_n^{(\tilde{\alpha},\tilde{\beta})}(x) = \frac{(-1)^n}{2^n n!}(1-x)^{-\tilde{\alpha}}(1+x)^{-\tilde{\beta}}\frac{d^n}{dx^n}\Big[(1-x)^{\tilde{\alpha}+n}(1+x)^{\tilde{\beta}+n}\Big], \tag{A.36}$$

where both $\tilde{\alpha}$ and $\tilde{\beta}$ are greater than -1. Jacobi polynomials are made orthonormal as

$$\xi(\tilde{\alpha},\tilde{\beta},n)\int_{-1}^{1}(1-x)^{\tilde{\alpha}}(1+x)^{\tilde{\beta}}J_m^{(\tilde{\alpha},\tilde{\beta})}(x)J_n^{(\tilde{\alpha},\tilde{\beta})}(x)dx = \delta_{mn}, \tag{A.37}$$

where

$$\xi(\tilde{\alpha},\tilde{\beta},n) = \frac{2n+\tilde{\alpha}+\tilde{\beta}+1}{2^{\tilde{\alpha}+\tilde{\beta}+1}}\frac{\Gamma(n+\tilde{\alpha}+1)\Gamma(n+\tilde{\beta}+1)}{\Gamma(n+\tilde{\alpha}+\tilde{\beta}+1)n!}. \tag{A.38}$$

Jacobi quadrature approximates the following integrals:

$$\int_{-1}^{1} h(x)(1-x)^{\tilde{\alpha}}(1+x)^{\tilde{\beta}}dx = \sum_{j=1}^{\infty} w_j h(x_j) \tag{A.39}$$

Transforming general Beta distributions to compatible domain with Jacobi polynomials and quadrature will be done in two steps, first to standard Beta distribution (using variable argument z) and then to Jacobi quadrature domain. To convert to standard Beta:

$$z = \frac{y-a}{b-L}, \quad y = (b-a)z + a, \quad dy = (b-a)dz, \tag{A.40}$$

$$1 = \frac{1}{\mathrm{B}(\alpha,\beta)}\int_0^1 z^{\alpha-1}(1-z)^{\beta-1}dz, \tag{A.41}$$

To convert to same form as the Jacobi probability weight,

$$z = \frac{1+x}{2}, \quad x = 2z-1, \quad dz = \frac{1}{2}dx, \tag{A.42}$$

so that

$$1 = \frac{1}{2^{\alpha+\beta-1}\mathrm{B}(\alpha,\beta)}\int_{-1}^{1}(1+x)^{\alpha-1}(1-x)^{\beta-1}dx. \tag{A.43}$$

Combining the transformations,

$$y = \frac{b-a}{2}x + \frac{b+a}{2}, \quad x = \left(y - \frac{b+a}{2}\right)\left(\frac{2}{b-a}\right) \tag{A.44}$$

In a potentially confusing twist, the Jacobi polynomial characteristic $\tilde{\alpha}$ is related to the Beta distribution parameter β, and the Jacobi polynomial characteristic $\tilde{\beta}$ is related to

the Beta distribution parameter α, as

$$\tilde{\alpha} = \beta - 1, \quad \tilde{\beta} = \alpha - 1. \tag{A.45}$$

As a result,

$$\int_a^b h(y)\rho(y)dy = \frac{1}{2^{\tilde{\alpha}+\tilde{\beta}-1}\mathrm{B}(\tilde{\alpha},\tilde{\beta})} \sum_{B=1}^{\infty} w_B h\left(\frac{b-a}{2}x_B + \frac{b+a}{2}\right). \tag{A.46}$$

A.6 Arbitrary Distributions and Legendre Polynomials

In general there is not a family of polynomials and quadratures that corresponds nicely for every probabilistic distribution. However, many continuous distributions have a CDF and inverse CDF that map the distribution to and from the domain $[0, 1]$. As a result, we can apply Legendre quadrature to the converted space. This does not perfectly preserve the integration properties of Gaussian quadrature, but does allow for general distributions to be covered. We require for arbitrary distributions that y is finite, or $-\infty < a \leq y \leq b < \infty$. In this case, the distribution has the following properties:

$$\rho(y) = \rho(y), \tag{A.47}$$

$$F(y) = \int_a^y \rho(y')dy', \tag{A.48}$$

$$1 = \int_a^b \rho(y)dy. \tag{A.49}$$

We now consider transforming to the domain $[0, 1]$ using the CDF. Let $u \in [0, 1]$, and note $F(y) \in [0, 1]$. Let

$$du = dF(y) = \rho(y)dy, \tag{A.50}$$

then

$$F(y) = u \quad \therefore \quad y = F^{-1}(u), \tag{A.51}$$

$$dy = \frac{1}{\rho(y)}du, \tag{A.52}$$

$$\int_a^b h(y)\rho(y)dy = \int_0^1 h\big(F^{-1}(u)\big)\rho\big(F^{-1}(u)\big)\frac{1}{\rho(F^{-1}(u))}du, \tag{A.53}$$

$$= \int_0^1 h\big(F^{-1}(u)\big)du. \tag{A.54}$$

$$x = \frac{u-\mu}{\sigma} \therefore u = \sigma x + \mu, \tag{A.55}$$

$$u = \frac{\hat{b} - \hat{a}}{2}x + \frac{\hat{b} + \hat{a}}{2}, \quad \hat{b} = 1, \hat{a} = 0, \tag{A.56}$$

$$u = \frac{1}{2}(x + 1), \tag{A.57}$$

$$\int_a^b h(y)\rho(y)dy = \int_0^1 h\big(F^{-1}(u)\big)du, \tag{A.58}$$

$$= \frac{1}{2}\sum_{\ell=1}^{\infty} w_\ell h\left(F^{-1}\left(\frac{1}{2}(x_\ell + 1)\right)\right)du. \tag{A.59}$$

In this manner, arbitrary distributions with a continuous CDF and inverse CDF can be expanded using Legendre polynomials and integrated using Legendre quadrature. This work was presented in [25], and we have added the transformation algorithms explicitly.

Appendix B

Recovering ANOVA from cut-HDMR

B.1 Introduction

When using SCgPC to represent individual cut-HDMR subsets (see 5.3), it is simple to recover analytic ANOVA statistics for a cut-HDMR expansion, despite the lack of orthogonality in cut-HDMR terms. This is because the gPC components of each subset term are replete with orthogonal relationships. Note that while the following algorithm will obtain ANOVA results for cut-HDMR terms, the statistics gathered are for the cut-HDMR expansion, not for the original model. When the cut-HDMR expansion is truncated, the ANOVA terms will only be as accurate to the original model as the cut-HDMR expansion itself is.

To reconstruct the ANOVA decomposition of a cut-HDMR expansion, we simply apply ANOVA to the cut-HDMR expansion, which results in significant reduction of terms due to gPC orthogonalities. We begin at the cut-HDMR expansion with subsets determined by generalized polynomial chaos expansions by repeating Eq. 5.20,

$$T(Y) \approx t_r + \sum_{n=1}^{N} \left(\sum_{k' \in \Lambda'_n(L')} t_{n;k'} \Phi_{k'}(Y_n) \right) + \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} \left(\sum_{k' \in \Lambda'_{m,n}(L')} t_{m,n;k'} \Phi_{k'}(Y_m, Y_n) \right), \tag{B.1}$$

and recall the definition of ANOVA in Eq. 5.1, 5.2, 5.4, and 5.7. For demonstration, note we truncate the cut-HDMR to second-order effects in Eq. B.1, but the concepts extend to higher-order truncations. To further simplify, we consider a three-dimension input

space for $T(Y) = T(x, y, z)$, which again can be extended to higher dimensions. Further, to simplify some notation, we express the generalized polynomial chaos expansion of a subset with respect to an input variable y_n as $G(y_n)$,

$$T(y_n, \hat{\bar{Y}}_n) \approx G(y_n) = \sum_{k' \in \Lambda'_n(L')} t_{n;k'} \Phi_{k'}(Y_n), \tag{B.2}$$

so that for example

$$t_n(y_n) = T(y_n, \hat{\bar{Y}}_n) - t_r \approx G(y_n) - t_r. \tag{B.3}$$

Eq. B.1 then becomes

$$T(x, y, z) = t_r + t_x + t_y + t_z + t_{xy} + t_{xz} + t_{yz}, \tag{B.4}$$

with the following definitions:

$$t_r = T(\bar{x}, \bar{y}, \bar{z}), \tag{B.5}$$

$$t_x = T(x, \bar{y}, \bar{z}) - t_r \approx G(x) - t_r, \tag{B.6}$$

$$t_y = T(\bar{x}, y, \bar{z}) - t_r \approx G(y) - t_r, \tag{B.7}$$

$$t_z = T(\bar{x}, \bar{y}, z) - t_r \approx G(z) - t_r, \tag{B.8}$$

$$t_{xy} = T(x, y, \bar{z}) - t_x - t_y - t_r \approx G(x, y) - t_x - t_y - t_r, \tag{B.9}$$

$$t_{xz} = T(x, \bar{y}, z) - t_x - t_z - t_r \approx G(x, z) - t_x - t_z - t_r, \tag{B.10}$$

$$t_{yz} = T(\bar{x}, y, z) - t_y - t_z - t_r \approx G(y, z) - t_y - t_z - t_r. \tag{B.11}$$

Substituting and collecting terms,

$$T(x, y, z) \approx t_r - G(x) - G(y) - G(z) + G(x, y) + G(x, z) + G(y, z), \tag{B.12}$$

where the approximation primarily depends on the ability of SCgPC to represent each subset space. In the limit that infinite polynomials are available, the equation becomes exact.

For the purposes of derivations in this section, we continue to implicitly assume all integrations over an input space Ω_n are with respect to $\rho_n(y_n)$,

$$\int_{\Omega_n} f(y_n) dy_n = \int_{a_n}^{b_n} \rho_n(y_n) f(y_n) dy_n, \tag{B.13}$$

$$\int_{\Omega} f(Y) dY = \int_{a_1}^{b_1} \cdots \int_{a_N}^{b_N} \rho(y_1, \cdots, y_N) f(y_1, \cdots, y_N) dy_1 \cdots, dy_N. \tag{B.14}$$

We now apply ANOVA to $T(x,y,z)$ as $H[T](x,y,z)$ by considering each ANOVA subset term individually. The first term in ANOVA, the expectation value h_0, is given as

$$h_0 = \int_{\Omega} T(Y)dY, \tag{B.15}$$

which expands into the sum of individual integrals

$$\begin{aligned} h_0 =& t_r \\ &- \int_{\Omega_x} G(x)dx - \int_{\Omega_y} G(y)dy - \int_{\Omega_z} G(z)dz \\ &+ \int_{\Omega_{x,y}} G(x,y)dxdy + \int_{\Omega_{x,z}} G(x,z)dxdz + \int_{\Omega_{y,z}} G(y,z)dydz, \end{aligned} \tag{B.16}$$

recalling that by definition

$$\int_{\Omega_n} dy_n = 1. \tag{B.17}$$

Also recalling the nature of the orthonormal polynomials families in SCgPC,

$$\int_{\Omega_n} \phi_{k_n}(y_n)dy_n = 0 \quad \forall \quad k_n > 0, \tag{B.18}$$

all nonzero polynomial terms integrate to zero,

$$\int_{\Omega_x} G(x)dx = \int_{\Omega_x} \sum_{k' \in \Lambda'} c_{k'} \Phi_{k'}(x)dx = c_{\varnothing}^{(x)}, \tag{B.19}$$

$$\int_{\Omega_{x,y}} G(x,y)dxdy = \int_{\Omega_{x,y}} \sum_{k' \in \Lambda'} c_{k'} \Phi_{k'}(x,y)dxdy = c_{\varnothing}^{(x,y)}, \tag{B.20}$$

where we use the parenthetical superscript to denote the subset origin of the scalar coefficients and the subscript $\varnothing$ to indicate $k = \{0\}^{N_s}$, where N_s is the dimensionality of the expansion subset. Because of model symmetry, the same process applies for subsets (y) and (z) as for subset (x), and the same process applies for subsets (x,z) and (y,z) as for subset (x,y). As a result, the zeroth-order ANOVA term is

$$h_0 = t_r - c_{\varnothing}^{(x)} - c_{\varnothing}^{(y)} - c_{\varnothing}^{(z)} + c_{\varnothing}^{(x,y)} + c_{\varnothing}^{(x,z)} + c_{\varnothing}^{(y,z)}, \tag{B.21}$$

or simply the zeroth polynomial order contribution terms from each subset expansion subset in Eq. B.12.

For first-order ANOVA terms (first-order interactions), we consider first h_x.

$$\begin{aligned} h_x &= \int_{\Omega_{y,z}} T(x,y,z)\ dy\ dz - h_0, \\ &= t_r - G(x) - \int_{\Omega_y} G(y)\ dy - \int_{\Omega_z} G(z)\ dz + \int_{\Omega_y} G(x,y)\ dy + \int_{\Omega_z} G(x,z)\ dz \\ &\quad + \int_{\Omega_{y,z}} G(y,z)\ dy\ dz - h_0. \end{aligned} \tag{B.22}$$

Performing the integrals, for example

$$\begin{aligned} \int_{\Omega_y} G(x,y)\ dy &= \int_{\Omega_y} \sum_{k\in\Lambda} c_k \Phi_k(x,y)\ dx, \\ &= \left\{ \begin{array}{ll} 0, & k_y \geq 1, \\ c_{(k_x,0)}\phi_{k_x}(x), & k_y = 0, \end{array} \right\} \\ &= \sum_{\substack{k\ \in\ \Lambda \\ k_y\ =\ 0}} c_k \phi_{k_x}(x). \end{aligned} \tag{B.23}$$

Evaluating all integrations and simplifying, we have an expression for h_x,

$$h_x = t_r - G(x) - c_\varnothing^{(y)} - c_\varnothing^{(z)} + \sum_{\substack{k\ \in\ \Lambda \\ k_y\ =\ 0}} c_k^{(xy)} \phi_{k_x}(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_z\ =\ 0}} c_k^{(xz)} \phi_{k_x}(x) + c_\varnothing^{(yz)} - h_0, \tag{B.24}$$

$$= c_\varnothing^{(x)} - G(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_y\ =\ 0}} c_k^{(xy)} \phi_{k_x}(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_z\ =\ 0}} c_k^{(xz)} \phi_{k_x}(x) - c_\varnothing^{(xy)} - c_\varnothing^{(xz)},$$

$$= - \sum_{\substack{k\ \in\ \Lambda \\ k_x\ >\ 0}} c_k^{(x)} \Phi_k(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_x\ >\ 0 \\ k_y\ =\ 0}} c_k^{(xy)} \phi_{k_x}(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_x\ >\ 0 \\ k_z\ =\ 0}} c_k^{(xz)} \phi_{k_x}(x).$$

Note that all the terms in Eq. B.24 are elements from each polynomial set where the only nonzero polynomial orders are those with respect to x. Because of the model symmetry in the cut-HDMR expansion, the procedure for h_y and h_z will be identical to h_x.

For second-order ANOVA terms (second-order interactions), we consider $h_{x,y}$.

$$\begin{aligned} h_{x,y} &= \int_{\Omega_z} T(x,y,z)\ dz - h_x - h_y - h_0, \\ &= t_r - G(x) - G(y) - c_{\varnothing}^{(z)} + G(x,y) + \sum_{\substack{k\,\in\,\Lambda \\ k_z\,=\,0}} c_k^{(xz)}\phi_{k_x}(x) + \sum_{\substack{k\,\in\,\Lambda \\ k_z\,=\,0}} c_k^{(yz)}\phi_{k_y}(y) \\ &\quad - h_x - h_y - h_0, \\ &= \sum_{\substack{k\,\in\,\Lambda \\ k_x\,>\,0 \\ k_y\,>\,0}} c_k \Phi_k(x,y). \end{aligned} \tag{B.25}$$

As with the first-order case, the second-order case contains only those polynomials whose order is greater than zero in all of its dependencies. Model symmetry dictates the same procedure for both $h_{x,z}$ and $h_{y,z}$ as $h_{x,y}$.

With all of the ANOVA terms in Eq. 5.1 calculated, it is possible to obtain moments of the cut-HDMR expansion using them. The expected value is simply the zeroth-order ANOVA term. The only subset gPC polynomials that do not integrate to zero are those that are entirely zeroth-order polynomials.

$$\mathbb{E}[H[T](x,y,z)] = h_0. \tag{B.26}$$

The second moment is the integral of the sum of the square of the terms, because each ANOVA term is orthogonal with respect to the remainder of the ANOVA terms,

$$\mathbb{E}[H[T](x,y,z)^2] = h_0^2 + \int_{\Omega} h_x^2 + h_y^2 + h_z^2 + h_{x,y}^2 + h_{x,z}^2 + h_{y,z}^2 dxdydz. \tag{B.27}$$

Because of the orthonormal properties of the polynomials within each expansion term,

$$\int_{\Omega} \sum_{\ell\in\Lambda_1} \sum_{k\in\Lambda_2} \Phi_\ell(x,y,z)\Phi_k(x,y,z)\ dx\ dy\ dz = \delta_{\ell,k}, \tag{B.28}$$

and because lower-dimensional polynomials are subsets of higher-dimensional polynomials,

$$\Phi_{k_x=1}(x) = \Phi_{k_x=1,k_y=0,k_z=0}(x,y,z) = \Phi_{1,0,0}(x,y,z), \tag{B.29}$$

$$\Phi_{k_y=1}(y) = \Phi_{k_x=0,k_y=1,k_z=0}(x,y,z) = \Phi_{0,1,0}(x,y,z), \tag{B.30}$$

$$\Phi_{k_z=1}(y) = \Phi_{k_x=0,k_y=0,k_z=1}(x,y,z) = \Phi_{0,0,1}(x,y,z), \tag{B.31}$$

and so on, The integral of the square of each term is the sum of the squares of each applicable polynomial coefficient. For h_x^2,

$$\int_{\Omega_x} h_x^2 dx = \sum_{\substack{k \in \Lambda \\ k_x > 0}} \left(c_k^{(x)}\right)^2 + \sum_{\substack{k \in \Lambda \\ k_x > 0 \\ k_y = 0}} \left(c_k^{(xy)}\right)^2 + \sum_{\substack{k \in \Lambda \\ k_x > 0 \\ k_z = 0}} \left(c_k^{(xz)}\right)^2, \tag{B.32}$$

and by symmetry we obtain h_y^2 and h_z^2 as well. For $h_{x,y}^2$,

$$\int_{\Omega} h_{xy}^2 \; dx \; dy = \sum_{\substack{k \in \Lambda \\ k_x > 0 \\ ky > 0}} \left(c_k^{(xy)}\right)^2, \tag{B.33}$$

and similarly for $h_{x,z}^2$ and $h_{y,z}^2$.

Note that implementing cut-HDMR to ANOVA algorithms is more straightforward than the derivation; ultimately, the Sobol coefficients, which are equivalent to the second moment of each ANOVA subset term, are simply a sum of the square of all the coefficients in all the constituent cut-HDMR subset SCgPC terms for whom the only nonzero polynomials are those that the Sobol coefficient term is with respect to. Because the terms in both the expected value and the variance are only scalar values, there are efficient to obtain computationally with a high degree of accuracy and with little effort to implement.

Bibliography

[1] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press, 2010.

[2] 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.

[3] Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments, 2015. URL `http://www.sfu.ca/~ssurjano/index.html`. Accessed: 2016-09-01.

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

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

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

[7] 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.

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

[9] 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.

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

[11] 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.

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

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

[14] 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.

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

[16] Herschel Rabitz, Ömer F Aliş, Jeffrey Shorter, and Kyurhee Shim. Efficient input—output model representations. *Computer Physics Communications*, 117(1): 11–20, 1999.

[17] Herschel Rabitz and Ömer F Aliş. General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, 25(2-3):197–233, 1999.

[18] 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.

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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

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

[27] 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.

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

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

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

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

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

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

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

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

[36] 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.

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

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

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

[40] 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.

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

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

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

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

[45] 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.

[46] 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.

[47] A Markov. *On certain applications of algebraic continued fractions*. PhD thesis, PhD thesis, St. Petersburg, 1884. in Russian, 1884.

[48] Saltelli and Sobol. About the use of rank transformation in sensitivity analysis of model output. *Reliablility Engineering & System Safety*, 50(3):225–239, 1995.

[49] Marrel, Iooss, van Dorpe, and Volkova. An efficient methodology for modeling complex computer codes with gaussian processes. *Computational Statistics and Data Analysis*, 52(10):4731–4744, 2008.

[50] Blyth, Porter, Avramova, Ivanov, Royer, Sartori, Cabellow, 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)*, 2, 2014.

[51] Swiler, Kyle Gamble, Rodney C Schmidt, and Richard Williamson. Sensitivity analysis of oecd benchmark tests in bison. Technical report, Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), 2015.

[52] 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.

[53] 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.

[54] Jon Loeliger and Matthew McCullough. *Version Control with Git: Powerful tools and techniques for collaborative software development.* " O'Reilly Media, Inc.", 2012.

[55] Gleicher, Ortensi, Spencer, Wang, Novascone, Hales, Gaston, Williamson, and Martineau. The coupling of the neutron transport application "rattlesnake" to the nuclear fuels performance application "bison" under the "moose" framework. In *International Topical Meeting on Advances in Reactor Physics*, Kyoto, Japan, 2014.

[56] Steven M Bowman. Scale 6: comprehensive nuclear safety analysis code system. *Nuclear technology*, 174(2):126–148, 2011.

[57] James J Duderstadt and Louis J Hamilton. Nuclear reactor analysis. 1976.

[58] Elmer E Lewis. *Fundamentals of nuclear reactor physics.* Academic Press, 2008.

[59] John R Lamarsh. *Nuclear Reactor Theory.* Addison-Wesley, 1966.

[60] Elmer Eugene Lewis and Warren F Miller. Computational methods of neutron transport. 1984.

[61] EE Lewis, MA Smith, N Tsoulfanidis, G Palmiotti, TA Taiwo, and RN Blomquist. Benchmark specification for deterministic 2-d/3-d mox fuel assembly transport calculations without spatial homogenization (c5g7 mox). *NEA/NSC*, 2001.

[62] HH Ku. Notes on the use of propagation of error formulas. *Journal of Research of the National Bureau of Standards*, 70(4), 1966.

[63] AA Clifford. Multivariate error analysis: A handbook of error propagation and calculation in many-parameter system, 1973.

[64] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation, 1964.

[65] WG Cochran and Gertrude M Cox. Notes on the statistical analysis of the results. 1992.