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

Abstract

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

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

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

Contents

List of Figures

List of Tables

Chapter 1

Introduction

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

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

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

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

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

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

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

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

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

The second method we consider is stochastic collocation for generalized polynomial chaos (SCgPC)[23, 34–36], whereby deterministic collocation points are used to develop a polynomial-interpolated reduced-order model of the response as a function of the inputs. This method algorithmically expands the solver as the sum of orthogonal multi-dimensional polynomials with scalar coefficients. The scalar coefficients are obtained by

numerical integration using multidimensional collocation (quadrature) points. The chief distinction between SCgPC and Monte Carlo methods is that SCgPC is deterministic, in that the realizations required from the solver are predetermined instead of randomly sampled. There are two major classes of deterministic uncertainty quantification methods: intrusive and unintrusive. Like Monte Carlo, SCgPC is unintrusive and performs well without any need to access the operation of the solver. This behavior is desirable for construction black-box approach algorithms for uncertainty quantification. Other intrusive methods such as stochastic Galerkin exist [17], but require solver modification to operate. This makes them solver-dependent and undesirable for an independent uncertainty quantification framework.

The other methods we present here expand on SCgPC. First, we introduce non-tensor-product methods for determining the set of polynomial bases to use in the expansion. Because a tensor product grows exponentially with increasing cardinality of the input space, we combat this curse of dimensionality using the alternative polynomial set construction methods[25]. These bases will then be used to construct Smolyak-like sparse grids [26] to provide collocation points that in turn calculate the coefficients in the polynomial expansion. Second, we consider anisotropic sparse grids, allowing higher-order polynomials for particular input parameters. We also consider methods for obtaining weights that determine the level of anisotropic preference to give parameters, and explore the effects of a variety of anisotropic choices.

The second method group we consider is high-dimension model representation (HDMR), which correlates with Sobol decomposition [32]. This method is useful both for developing sensitivities of the quantity of interest to subsets of the input space, as well as constructing a reduced-order model itself. We demonstrate the strength of HDMR as a method to inform anisotropic sensitivity weights for SCgPC.

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

We implement all these methods in Idaho National Laboratory's `RAVEN`[21] uncertainty quantification framework. `RAVEN` is a Python-written framework that non-intrusively provides tools for analysts to quantify the uncertainty in their simulations with minimal development. To demonstrate the application of the method developed, we use a complex non-linear multiphysics system solver simulating the operation of a fuel pin within a nuclear reactor core, including both neutronics and fuel performance physics kernals. For this solver, we use the coupled `RATTLESNAKE`[22] and `BISON` [18, 20] production

codes. Both of these codes are developed and maintained within the MOOSE[19] environment. The multiphysics nonlinear system provides a challenge with unknown response properties for the uncertainty quantification methods discussed in this proposal.

The remainder of this work will proceed as follows:

- Chapter 2: We describe the analytic test problems and engineering-scale problem solved by the simulations we will be running, along with their properties and inferences about the algorithms developed. We discuss potential approaches to model solving and applications of the models.
- Chapter 3: We describe methods for uncertainty quantification, including Monte Carlo (MC), stochastic collocation for generalized Polynomial Chaos (SCgPC), and high-dimension model reduction (HDMR). We additionally describe adaptive methods for SCgPC and HDMR.
- Chapter 4: We analyze results obtained for the various UQ methods on analytic models, and contrast them with traditional Monte Carlo convergence on statistical moments.
- Chapter 5: We perform analysis on the engineering-scale multiphysics coupled problem, and analyze results.
- Chapter 6: We consider application of collocation-based methods to time-dependent sensitivity analysis.
- Chapter 7: We draw conclusions from the evaluations performed, and offer some suggestions for applicability and limitations discovered.
- Chapter 8: We consider new research and future development uncovered by the UQ methods demonstrated here.

Chapter 2

Methods: Stochastic Collocation for Generalized Polynomial Chaos

2.1 Introduction

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

Many simulation models are algorithms constructed to solve partial differential equations, often in two or three dimensions and possibly time. The inputs to these models include boundary conditions, material properties, tuning parameters, and so forth. The outputs are quantities of interest, either data fields or scalar values. The outputs are used to inform decision-making processes. 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.

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

the output is a vector or field quantity, each element can be considered as a distinct scalar response. Our generic model takes the form

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

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, etc.

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

$$\int_a^b \rho_n(y_n) d\ y_n = 1, \tag{2.2}$$

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 almost surely lies between a and b.

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

$$\int_{a_1}^{b_1} \cdots \int_{a_N}^{b_N} \rho(Y) dy_1 \cdots dy_N = 1. \tag{2.3}$$

For clarity, we define multidimensional integral operator

$$\int_{\Omega}(\cdot)dY \equiv \int_{a_1}^{b_1}\cdots\int_{a_N}^{b_N}(\cdot)dy_1\cdots dy_N, \tag{2.4}$$

so that Eq. 2.3 can be written

$$\int_{\Omega}\rho(Y)dY = 1. \tag{2.5}$$

The function $u(Y)$ maps realizations ω from the input space Ω to a real-valued response Q. 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 $Y(\omega)$, which can be used as inputs for $u(Y)$ to obtain a realization of the response $u(Y(\omega)) = Q(\omega)$. To simplify notation, in general the dependency of a realization on ω will be omitted and implicit in equations.

2.1.1 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 here. For example, in simulating a projectile traveling under the influence of gravity, the initial release velocity magnitude $|v|$ along with the horizontal v_x and vertical v_y components of that velocity might all be inputs to a projectile model. However, because there is a relationship between the horizontal and vertical velocity components and the magnitude of the total velocity, one cannot perturb the horizontal, vertical, and total velocity independently; they are correlated through the equations

$$v_x = |v|\cos\theta, \tag{2.6}$$

$$v_y = |v|\sin\theta, \tag{2.7}$$

where θ is the angle between horizontal and the initial trajectory of the projectile.

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. For example, a transformation matrix can be constructed that represents coupled (v_x, v_y, v_z) with independent variables (ℓ_1, ℓ_2, ℓ_3). For this trivial case, the transformation matrix is TODO.

$$todo \tag{2.8}$$

This transformation allows us to sample realizations using the independent variables (ℓ_1, ℓ_2, ℓ_3) and then provide the model $u(v_x, v_y, |v|)$ with the coupled parameter values it expects. While the example here is quite simple, there are mathematical approaches to decoupling input parameters through surrogate spaces. In particular, using principle component analysis (sometimes known as Karhunen-Loeve expansion [39]), the covariance matrix for the distributed input parameters can be used to construct a multidimensional standard Gaussian normal distribution, whose components are all orthogonal. TODO more here! 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 problem through the model and comprehend their effects on the output responses. 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 mean provides the expected value of the response, or generally the most probable value for the response. The variance 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 [45] 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 expanding the range by multiples of the standard deviation. 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. 2.1 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

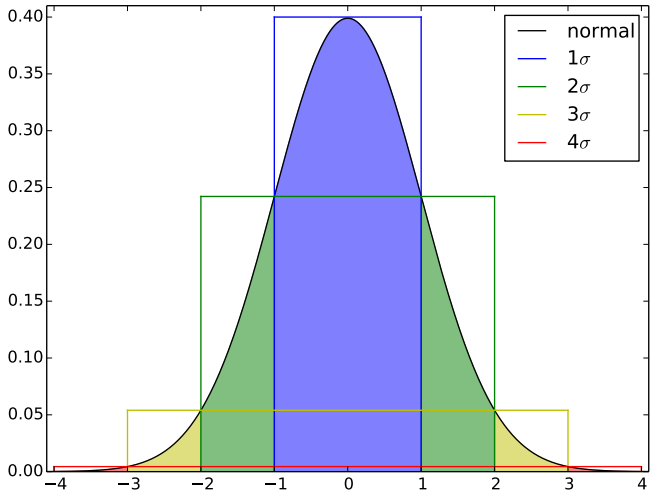

FIGURE 2.1: Standard Deviations of Normal Gaussian Distribution

Higher order moments, skewness and kurtosis, 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.

While both the skewness and kurtosis provide insight as to the distribution of reponses, 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,

$$\text{mean} = \mathbb{E}[u(Y)] = \int_{\Omega} \rho(Y)u(Y)dY, \tag{2.9}$$

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

$$\text{variance} = \mathbb{E}[u(Y)^2] - \mathbb{E}[u(Y)]^2 = \int_{\Omega} \rho(Y)u(Y)^2dY - \text{mean}^2. \tag{2.10}$$

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

In addition, 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 reponse as a function of changes in the input. This metric is most useful what attempting to maximize or minimize a response value by changing input parametrs. 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 occuring.

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 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 leads to failures, and what portion to successes, and define the hypersurfaces dividing successes and failures. 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 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 naieve cost-effective way to control variance are to decrease margins in fuel manufacturing instead of reflector construction. While this example seems readily evident, often engineering intuition can be surprised by uncertainty quantification and sensitiviy analysis.

2.2.3 Uncertainty Quantification Techniques

There are several common tools used for uncertainty quantification when analytic analysis is not possible. These include stochastic methods such as Monte Carlo sampling, deterministic methods such as Grid sampling, and mixed methods such as Latin Hypercube sampling (LHS). After describing these, we also consider advanced uncertainty quantification techniques: generalized polynomial chaos expansion and high-dimension model reduction.

2.2.4 Monte Carlo

The Monte Carlo method [10] 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 [11]. In its simplest form, Monte Carlo involves randomly picking realizations from a set of possibilities, then statistically collecting the results. In uncertainty quantification, Monte Carlo can be used to sample points in the input space based on the joint probability distribution. The collection of points is analyzed to determine the moments of the response.

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

$$\mathbb{E}[u(Y)] = \frac{1}{N}\sum_{m=1}^{M}\left(u(Y_m)\right) + \epsilon_M^{\mathrm{MC}}, \tag{2.11}$$

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

$$\epsilon_M^{\text{MC}} \propto \frac{1}{\sqrt{M}}. \tag{2.12}$$

The second moment is similarly approximated as

$$\mathbb{E}[u(Y)^2] \approx \frac{1}{N} \sum_{m=1}^{M} \big(u(Y_m)^2\big). \tag{2.13}$$

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 [12][13]; we restrict our discussion to traditional analog Monte Carlo sampling.

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

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

TODO 2d, 3d grid example

2.2.5 Grid

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

and CDF spacing, plus a normal distribution for comparison. In higher dimensions, a grid is constructed as a tensor product of each grid.

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

$$\mathbb{E}[u(Y)] = \int_{\Omega} \rho(Y)u(Y)dY \approx \sum_{m=1}^{M} w_m u(Y_m), \tag{2.14}$$

where m iterates over each node in the grid, Y_m is the multidimensional input point as node m, and w_m is a probability weight determined by the volume of probability represented by the node. In grids constructed by CDF, all w_m are of the same value, while in grids spaced equally by value, w_m can vary significantly. Similarly, the second moment is approximated by

$$\mathbb{E}[u(Y)^2] = \int_{\Omega} \rho(Y)u(Y)^2dY \approx \sum_{m=1}^{M} w_m u(Y_m)^2. \tag{2.15}$$

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

2.2.6 LHS

A cross between Monte Carlo and Grid sampling strategies, the Latin Hypercube Sampling (LHS) strategy has long been a sampling tool used to reduce the total samples needed without significantly sacrificing integration quality [15]. In LHS, the input space is also divided into a grid just as in the Grid sampling strategy. However, unlike Grid sampling, only one sample is taken per hyperplane; that is, for any of the input variables, there is only one sample taken between each of the one-dimensional nodes. Once a hypervolume is selected to take a sample, the exact point is selected by random sampling in the probability space within the hypervolume. See for example the sampling in Figure TODO.

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

2.3 Generalized Polynomial Chaos

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

Originally Wiener proposed expanding in Hermite polynomials for Gaussian-normal distributed variables [16]. Askey and Wilson generalized Hermite polynomials to include Jacobi polynomials, including Legendre and Laguerre polynomials [24]. Xiu and Karniadakis combined these concepts to perform PCE for a range of Gaussian-based distributions with corresponding polynomials, including Legendre polynomials for uniform distributions, Laguerre polynomials for Gamma distributions, and Jacobi polynomials for Beta distributions [23].

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) [23]. Two significant methodologies have grown from gPC application. The first makes use of Lagrange polynomials to expand the original function or simulation code, as Lagrange polynomials can be made orthogonal over the same domain as the distributions [31]; the other uses the Wiener-Askey polynomials [23]. We consider the latter in this work.

We consider a simulation code that produces a quantity of interest u as a function $u(Y)$ whose arguments are the uncertain, distributed input parameters $Y = (Y_1, \ldots, Y_n, \ldots, Y_N)$. A particular realization ω of Y_n is expressed by $Y_n(\omega)$, and a single realization of the entire input space results in a solution to the function as $u(Y(\omega))$. We acknowledge obtaining a realization of $u(Y)$ may take considerable computation time and effort, and may be solved nonlinearly. There also may be other input parameters that contribute to the solution of $u(Y)$; we neglect these, as our interest is in the uncertainty space; all parameters without uncertainty are held at their nominal values. In addition, it is possible

that the quantity of interest $u(Y)$ is an integrated quantity or some norm of a value that is temporally or spatially distributed. We restrict $u(Y(\omega))$ to a single scalar output, but the same principles apply to a multidimensional response. Further, a quantity of interest may be time-dependent in a transient simulation. In this case, the PCE can be constructed at several selected points in time throughout the simulation, which can then be interpolated between. In effect, the polynomial coefficients become time-dependent scalar values. For now, we consider a static case with no time dependence.

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

$$\Phi_k(Y) = \prod_{n=1}^{N} \phi_{k_n}(Y_n), \tag{2.16}$$

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

$$u(Y) \approx \sum_{k \in \Lambda(L)} u_k \Phi_k(Y), \tag{2.17}$$

where u_k is a scalar weighting polynomial coefficient. The polynomials used in the expansion are determined by the set of multi-indices Λ, which can be selected in a variety of ways we will discuss in section 2.3.1 and are the essence of this work. In the limit that Λ contains all possible combinations of polynomials of any order, Eq. 2.16 is exact. Practically, however, Λ is truncated to some finite set of combinations, discussed in section 2.3.1.

Using the orthonormal properties of the Wiener-Askey polynomials,

$$\int_{\Omega} \Phi_k(Y) \Phi_{\hat{k}}(Y) \rho(Y) dY = \delta_{k\hat{k}}, \tag{2.18}$$

where $\rho(Y)$ is the combined PDF of Y, Ω is the multidimensional domain of Y, and δ_{nm} is the Dirac delta, we can isolate an expression for the polynomial expansion coefficients. We multiply both sides of Eq. 2.16 by $\Phi_{\hat{k}}(Y)$, integrate both sides over the probability-weighted input domain, and sum over all $\hat{k}$ to obtain the coefficients, sometimes referred to as polynomial expansion moments,

$$u_k = \frac{\langle u(Y) \Phi_k(Y) \rangle}{\langle \Phi_k(Y)^2 \rangle}, \tag{2.19}$$

$$= \langle u(Y) \Phi_k(Y) \rangle, \tag{2.20}$$

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

$$\langle f(Y)\rangle \equiv \int_{\Omega} f(Y)\rho(Y)dY. \tag{2.21}$$

When $u(Y)$ has an analytic form, these coefficients can be solved by integration; however, in general other methods must be applied to numerically perform the integral. While tools such as Monte Carlo integration can be used to evaluate the integral, we can harness the properties of Gaussian quadratures because of the probability weights and domain. This stochastic collocation method is discussed in section 2.4.

2.3.1 Polynomial Index Set Construction

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

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

$$\Lambda_{\mathrm{TP}}(L) = \left\{\bar{p} = (p_1, \cdots, p_N) : \max_{1\leq n\leq N} p_n \leq L\right\}. \tag{2.22}$$

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 2.3, 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 2.3: Tensor Product Index Set, $N = 2, L = 3$

It is evident there is some inefficiencies in this index set. First, it suffers dramatically from the *curse of dimensionality*; that is, the number of polynomials required grows exponentially with increasing dimensions. Second, the total order of polynomials is not considered. Assuming the contribution of each higher-order polynomial is smaller than lower-order polynomials, the (3,3) term is contributing sixth-order corrections that are likely smaller than the error introduced by ignoring fourth-order corrections (4,0) and (0,4). This leads to the development of the *total degree* (TD) and *hyperbolic cross* (HC) polynomial index set construction strategies [25].

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

$$\Lambda_{\mathrm{TD}}(L) = \left\{ \bar{p} = (p_1, \cdots, p_N) : \sum_{n=1}^{N} p_n \leq L \right\}. \tag{2.23}$$

The cardinality of this index set is $|\Lambda_{\mathrm{TD}}(L)| = \binom{L+N}{N}$, which grows with increasing dimensions much more slowly than TP. For the same $N = 2, L = 3$ case above, the TD index set is given in Table 2.4.

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

TABLE 2.4: 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

$$\Lambda_{\mathrm{HC}}(L) = \left\{ \bar{p} = (p_1, \ldots, p_N) : \prod_{n=1}^{N} p_n + 1 \leq L + 1 \right\}. \tag{2.24}$$

The cardinality of this index set is bounded by $|\Lambda_{\mathrm{HC}}(L)| \leq (L+1)(1+\log(L+1))^{N-1}$. It grows even more slowly than TD with increasing dimension, as shown in Table 2.5 for $N = 2, L = 3$.

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

TABLE 2.5: Hyperbolic Cross Index Set, $N = 2, L = 3$

It has been shown that the effectiveness of TD and HC as index set choices depends strongly on the regularity of the responce [25]. TD tends to be most effective for infinitely-continuous response surfaces, while HC is more effective for surfaces with limited smoothness or discontinuities.

2.3.2 Anisotropy

While using TD or HC to construct the polynomial index set combats the curse of dimensionality present in TP, it is not eliminated and continues to be an issue for problems of large dimensionality. Another method that can be applied to mitigate this issue is index set anisotropy, or the unequal treatment of various dimensions. In this

strategy, weighting factors $\alpha = (\alpha_1, \ldots, \alpha_n, \ldots, \alpha_N)$ are applied in each dimension to allow additional polynomials in some dimensions and less in others. This change adjusts the TD and HC construction rules as follows, where $|\alpha|_1$ is the one-norm of α.

$$\tilde{\Lambda}_{\mathrm{TD}}(L) = \left\{ \bar{p} = (p_1, \cdots, p_N) : \sum_{n=1}^{N} \alpha_n p_n \leq |\alpha|_1 L \right\}, \tag{2.25}$$

$$\tilde{\Lambda}_{\mathrm{HC}}(L) = \left\{ \bar{p} = (p_1, \cdots, p_N) : \prod_{n=1}^{N} (p_n + 1)^{\alpha_n} \leq (L+1)^{|\alpha|_1} \right\}. \tag{2.26}$$

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

$$|\alpha|_1 = \frac{\sum_{n=1}^{N} \alpha_n}{N}. \tag{2.27}$$

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 2.6 (TD) and 2.7 (HC).

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

TABLE 2.6: Anisotropic Total Degree Index Set, $N = 2, L = 3$

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

TABLE 2.7: Anisotropic Hyperbolic Cross Index Set, $N = 2, L = 3$

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

2.3.3 Polynomial Expansion Features

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

constructed. The error in this representation will be of the same order as the truncation error of the expansion.

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

$$u(Y) \approx \tilde{u}(Y) \equiv \sum_{k \in \lambda} u_k \Phi_k(Y), \tag{2.28}$$

the mean is simply

$$\begin{aligned} \mathbb{E}[\tilde{u}(Y)] &= \int_\Omega \rho(Y) \sum_{k \in \Lambda} u_k \Phi_k(Y) dY, \\ &= u_{(0,\cdots,0)}. \end{aligned} \tag{2.29}$$

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

$$\begin{aligned} \mathbb{E}[\tilde{u}(Y)^2] &= \int_\Omega \rho(Y) \left[\sum_{k \in \Lambda} u_k \Phi_k(Y) \right]^2 dY, \\ &= \int_\Omega \rho(Y) \sum_{k_1 \in \Lambda} \sum_{k_2 \in \Lambda} u_{k_1} \Phi_{k_1}(Y) \cdot u_{k_2} \Phi_{k_2}(Y) dY, \\ &= \sum_{k_1 \in \Lambda} \sum_{k_2 \in \Lambda} u_{k_1} \cdot u_{k_2} \delta_{k_1,k_2}, \\ &= \sum_{k \in \Lambda} u_k^2. \end{aligned} \tag{2.30}$$

2.4 Stochastic Collocation

Having outlined the PCE 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. Quadrature integration takes the form

$$\int_a^b f(x)\rho(x) = \sum_{\ell=1}^{\infty} w_\ell f(x_\ell), \tag{2.31}$$

$$\approx \sum_{\ell=1}^{\hat{L}} w_\ell f(x_\ell), \tag{2.32}$$

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

$$q^{(\hat{L})}[f(x)] \equiv \sum_{\ell=1}^{\hat{L}} w_\ell f(x_\ell). \tag{2.33}$$

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

$$Q^{(\mathbf{L})} = q_1^{(\hat{L}_1)} \otimes q_2^{(\hat{L}_2)} \otimes \cdots, \tag{2.34}$$

$$= \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}. \tag{2.35}$$

It is worth noting each quadrature may have distinct points and weights; they need not be constructed using the same quadrature rule. In general, one-dimensional Gaussian quadrature excels in exactly integrating polynomials of order $2p-1$ using p points and weights; equivalently, it requires $(p+1)/2$ points to integrate an order p polynomial. For convenience we repeat here the coefficient integral we desire to evaluate, Eq. 2.19.

$$u_k = \langle u(Y)\Phi_k(Y)\rangle. \tag{2.36}$$

We can approximate this integral with the appropriate Gaussian quadrature as

$$u_k \approx Q^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)], \tag{2.37}$$

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.

$$u_k \approx q^{(\hat{L})}[u(Y)\Phi_k(Y)], \tag{2.38}$$

$$= \sum_{\ell=1}^{\hat{L}} w_\ell u(Y_\ell)\Phi_k(Y_\ell). \tag{2.39}$$

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. 2.16 and replace it in the sum,

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell \Phi_k(Y_\ell) \sum_{\hat{k} \in \Lambda(L)} u_{\hat{k}} \Phi_{\hat{k}}(Y_\ell). \tag{2.40}$$

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

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell u_k \Phi_k(Y_\ell)^2. \tag{2.41}$$

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

$$p = \frac{2k+1}{2} = k + \frac{1}{2} < k + 1, \tag{2.42}$$

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

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

2.4.1 Smolyak Sparse Grids

Smolyak sparse grids [26] are an attempt to discover the smallest necessary quadrature set to integrate a multidimensional integral with varying orders of predetermined quadrature sets. In our case, the polynomial index sets determine the quadrature orders each one needs in each dimension to be integrated accurately. For example, the polynomial index set point (2,1,3) requires three points in Y_1, two in Y_2, and four in Y_3,or

$$Q^{(2,1,3)} = q_1^{(3)} \otimes q_2^{(2)} \otimes q_3^{(4)}. \tag{2.43}$$

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

$$Q^{(\Lambda(L))} = \bigotimes_{k \in \Lambda} Q^{(k)}. \tag{2.44}$$

Smolyak sparse grids consolidate this tensor form by adding together the points from tensor products of subset quadrature sets. Returning momentarily to a one-dimensional problem, we introduce the notation [35]

$$\Delta_k^{(\hat{L})}[f(x)] \equiv \left(q_k^{(\hat{L})} - q_{k-1}^{(\hat{L})}\right)[f(x)], \tag{2.45}$$

$$q_0^{(\hat{L})}[f(x)] = 0. \tag{2.46}$$

A Smolyak sparse grid is then defined and applied to the desired integral in Eq. 2.19,

$$S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)] = \sum_{k\in\Lambda(L)} \left(\Delta_{k_1}^{(\hat{L}_1)} \otimes \cdots \otimes \Delta_{k_N}^{(\hat{L}_N)}\right)[u(Y)\Phi_k(Y)]. \tag{2.47}$$

Equivalently, and in a more algorithm-friendly approach,

$$S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)] = \sum_{k\in\Lambda(L)} c(k) \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}[u(Y)\Phi_k(Y)] \tag{2.48}$$

where

$$c(k) = \sum_{\substack{j=\{0,1\}^N, \\ k+j\in\Lambda}} (-1)^{|j|_1}, \tag{2.49}$$

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

$$u_k = \langle u(Y)\Phi_k(Y)\rangle, \tag{2.50}$$

$$\approx S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)]. \tag{2.51}$$

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

TODO expand to include the matlab algorithm from Motamed's class

2.5 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 [28] and used in [8]. The algorithm proceeds generally as follows:

- Begin with the mean (zeroth-order) polynomial expansion.
- While not converged:

- Collect a list of the polynomial index set whose predecessors have all been evaluated.
- Predict the impact of adding each polynomial to the existing polynomial index set.
- If the total impact of all indices is less than tolerance, convergence is reached.
- Otherwise, add the predicted highest-impact polynomial and loop back.

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

Figures 2.2 and 2.3 show a single 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, 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.

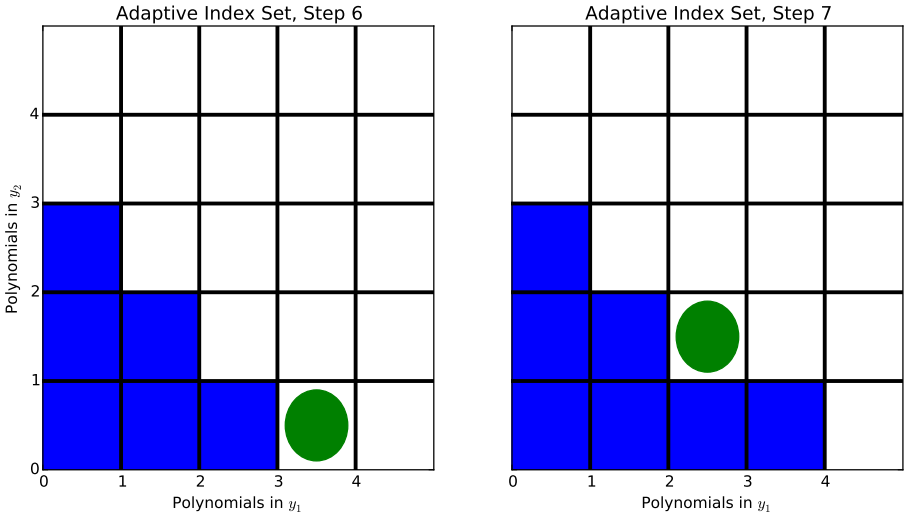

FIGURE 2.2: Adaptive Sparse Grid Step

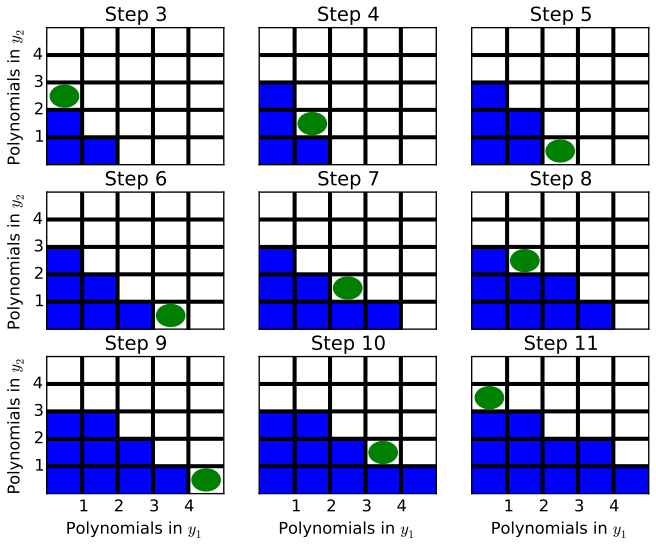

FIGURE 2.3: Adaptive Sparse Grid Progression

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; instead, it evaluates each potential index and selects the one with the most impact [8]. This is somewhat inefficient, because of SCgPC representations created that are not used in the final product. One improvement we make to this algorithm is to predict the impact of un-evaluated polynomials based on the impact of predecessors.

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

$$\eta_k = u_k^2. \tag{2.52}$$

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

$$\tilde{\eta}_k = \frac{1}{N - j} \sum_{n=1}^{N} \eta_{k - e_n}, \tag{2.53}$$

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

prediction algorithm may be misled, in general it saves significantly over the previous algorithm.

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

$$f(a, b) = a^3 b^3, \tag{2.54}$$

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

Chapter 3

Results: Stochastic Collocation for Generalized Polynomial Chaos

TODO get rid of the HDMR results from this section and rewrite results analysis!

3.1 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 approach. 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 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. The Monte Carlo performance is taken at a few representative points. Error bars are provided for the Monte Carlo method and are estimated using the population variance,

$$\epsilon_{95} = \frac{1.96\bar{\sigma}_N}{\sqrt{N}}, \tag{3.1}$$

$$\bar{\sigma}_N^2 = \frac{N}{N-1}\sigma_N^2 = \frac{N}{N-1}\left(\frac{1}{N}\sum_{i=1}^{M} u(Y(\omega_i))^2 - \bar{u}(Y)_N^2\right), \tag{3.2}$$

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

Convergence figures are log-log error graphs with the number of computational solves on the x-axis and error with respect to the analytical solution on the y-axis. The distinct lines demonstrate series of results obtained for each UQ method. The series we show here are analog traditional Monte Carlo; static stochastic collocation for generalized polynomial chaos expansion 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 stochastic collocation for polynomial chaos method (SC:adapt). Each series obtains additional values by increasing the refinement of the method. For Monte Carlo, 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 this is useful, 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 on 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.

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

3.2 Tensor Monomials

3.2.1 Description

The simplest model we make use of is a first-order tensor polynomial (tensor monomial) combination [8]. 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

$$u(Y) = \prod_{n=1}^{N} (y_n + 1). \tag{3.3}$$

For example, for $N = 3$ we have

$$u(Y) = y_1 y_2 y_3 + y_1 y_2 + y_1 y_3 + y_2 y_3 + y_1 + y_2 + y_3 + 1. \tag{3.4}$$

For this model we 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 3.1. The two-dimensional representation of this function is given in Figure 3.1.

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

TABLE 3.1: 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.

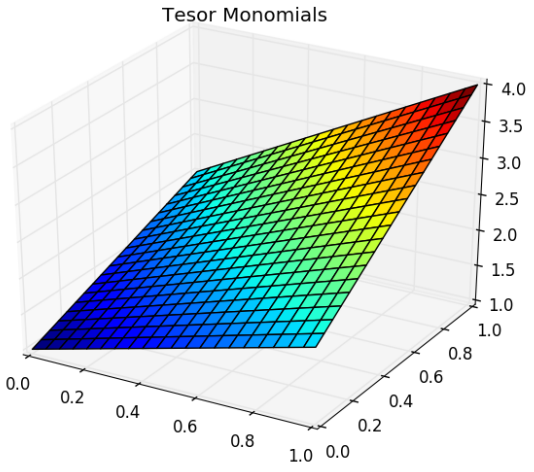

FIGURE 3.1: Tensor Monomials

3.2.2 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 magnitude. 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 5.1 through 5.4), five (Figures 5.5 through 5.8), and ten (Figures 5.9 through 5.12) input parameters. Note specially that TP exactly reproduces the original model with expansion order 1, so no convergence is observed past the initial sampling point.

3.2.3 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.

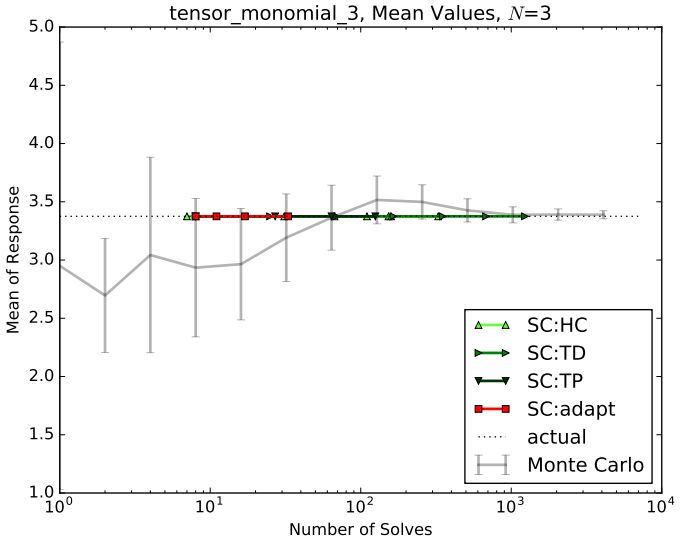

FIGURE 3.2: Tensor Monomial, $N = 3$, Mean Values

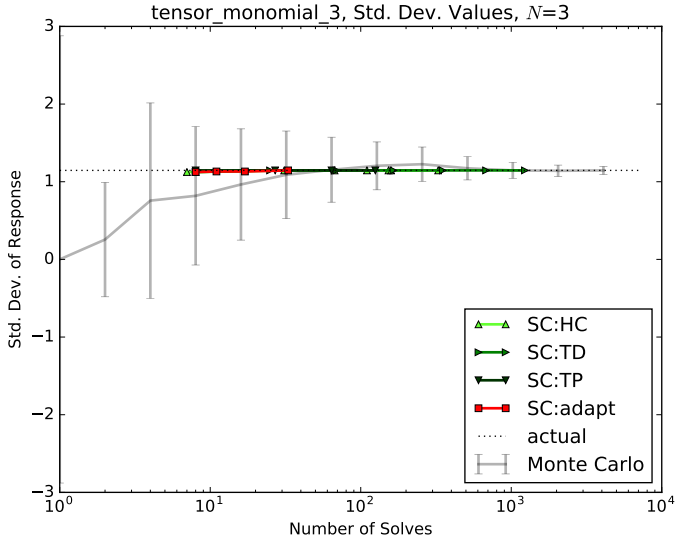

FIGURE 3.3: Tensor Monomial, $N = 3$, Std. Dev. Values

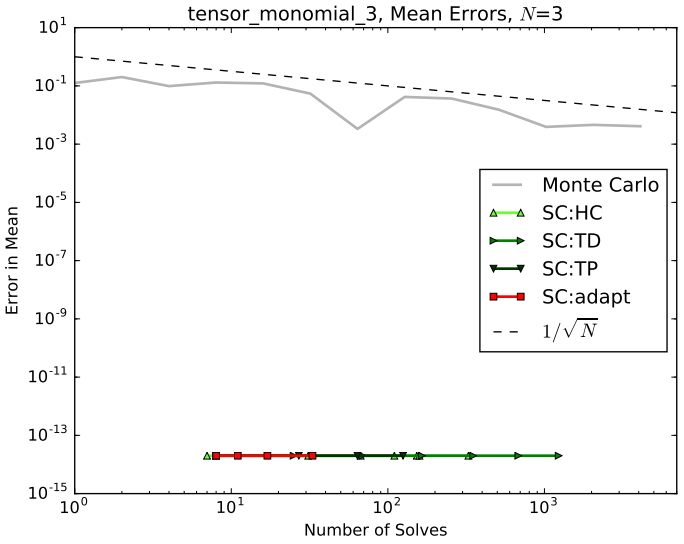

FIGURE 3.4: Tensor Monomial, $N = 3$, Mean Convergence

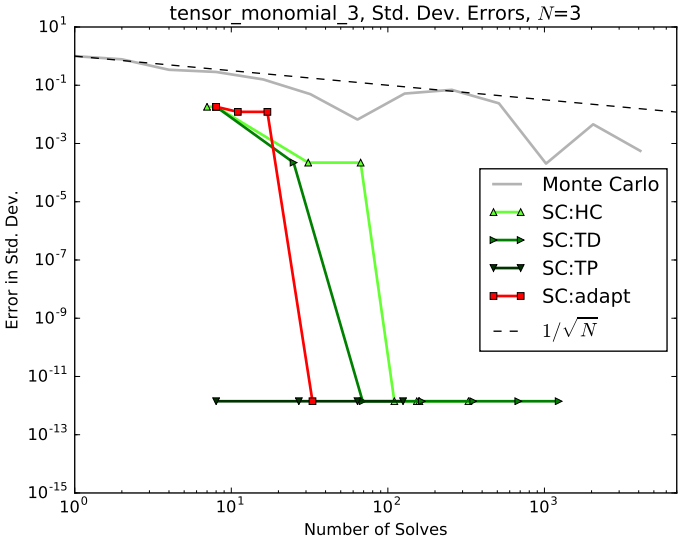

FIGURE 3.5: Tensor Monomial, $N = 3$, Std. Dev. Convergence

3.2.4 5 Inputs

While the convergence on the mean is still direct for the five-dimensional input problem, we begin to see degradation in the convergence of collocation-based methods. 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 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.

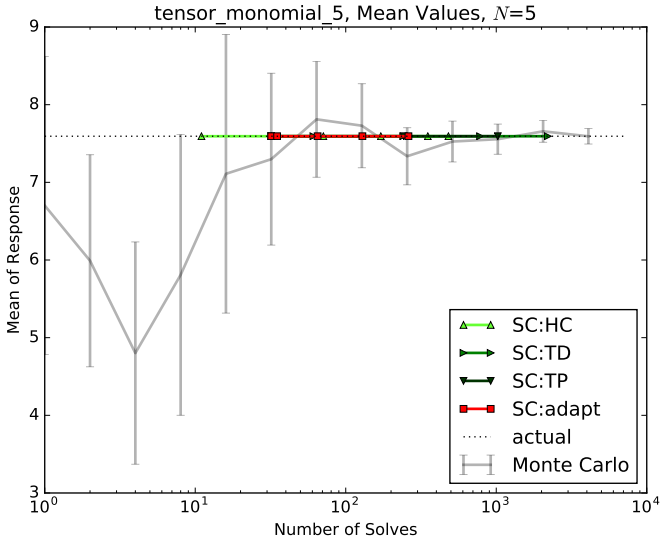

FIGURE 3.6: Tensor Monomial, $N = 5$, Mean Values

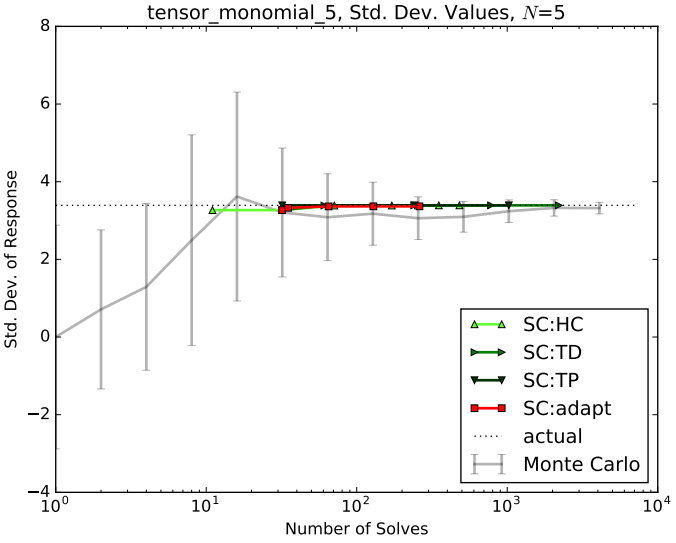

FIGURE 3.7: Tensor Monomial, $N = 5$, Std. Dev. Values

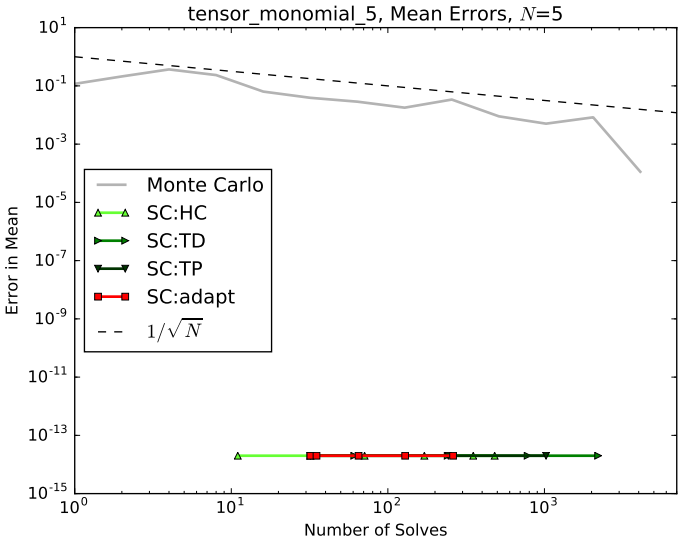

FIGURE 3.8: Tensor Monomial, $N = 5$, Mean Convergence

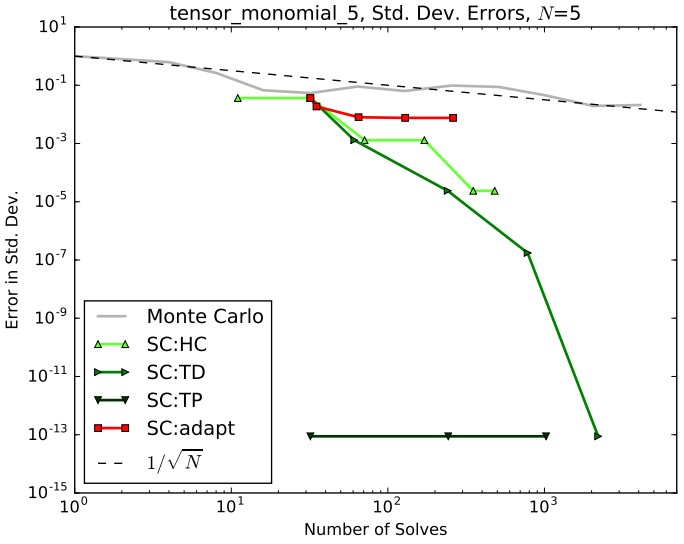

FIGURE 3.9: Tensor Monomial, $N = 5$, Std. Dev. Convergence

3.2.5 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 of the inputs.

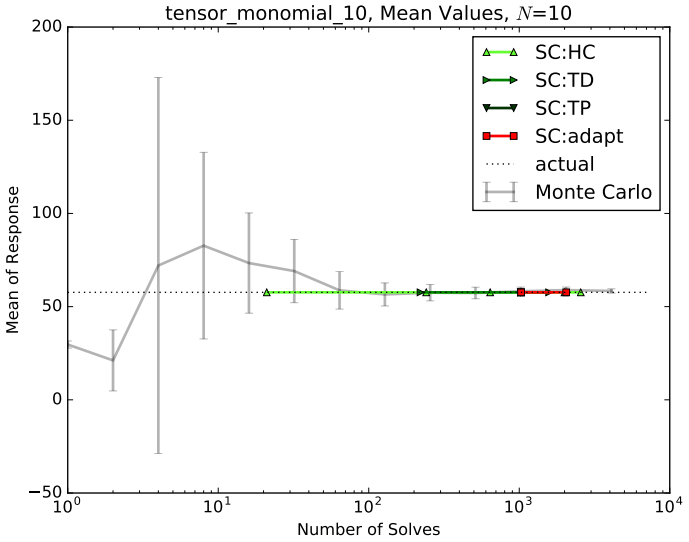

FIGURE 3.10: Tensor Monomial, $N = 10$, Mean Values

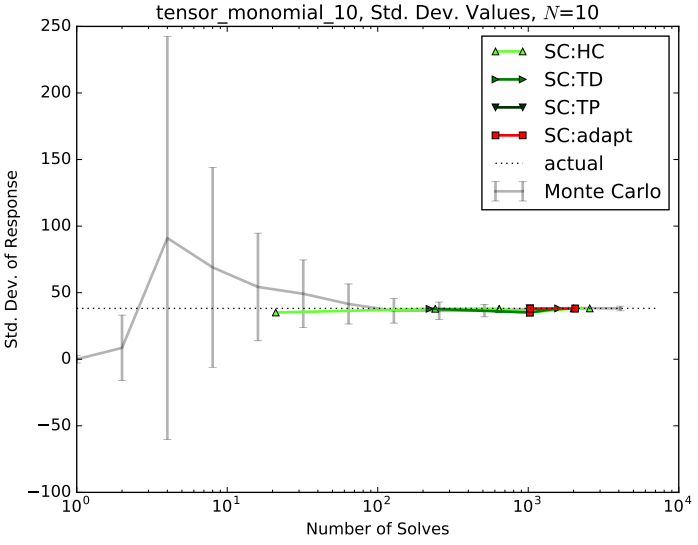

FIGURE 3.11: Tensor Monomial, $N = 10$, Std. Dev. Values

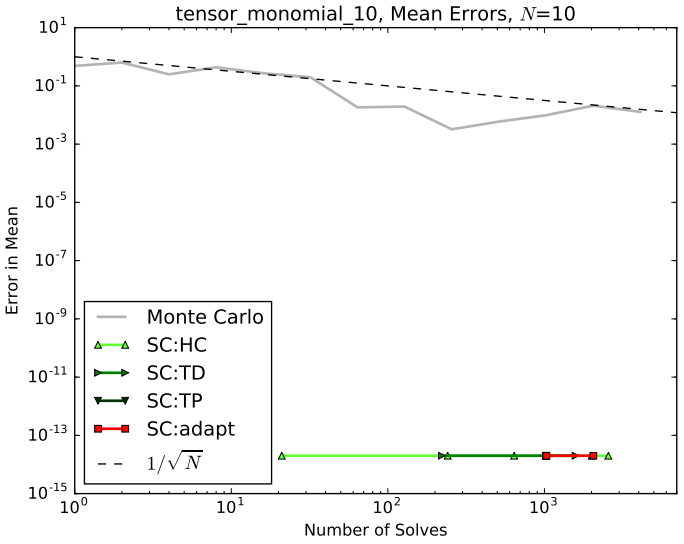

FIGURE 3.12: Tensor Monomial, $N = 10$, Mean Convergence

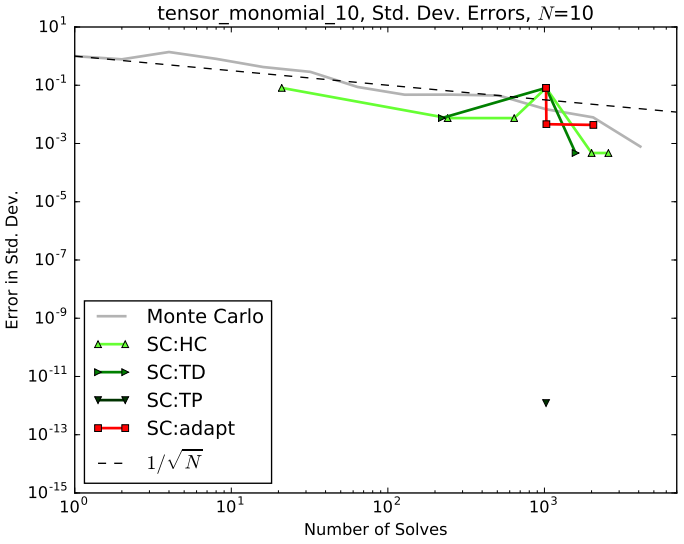

FIGURE 3.13: Tensor Monomial, $N = 10$, Std. Dev. Convergence

3.3 Sudret Polynomial

3.3.1 Description

The polynomial used by Sudret in his work [41] 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

$$u(Y) = \frac{1}{2^N} \prod_{n=1}^{N} (3y_n^2 + 1). \tag{3.5}$$

The two-dimensional representation of this function is given in Figure 3.14. The variables

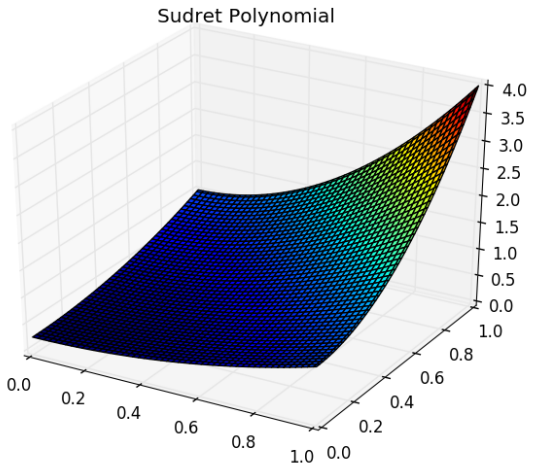

FIGURE 3.14: Sudret Polynomial

are distributed uniformly on [0,1].

The statistical moments and sensitivities are given in Table 3.2, 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 3.2: Analytic Expressions for Sudret Case

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

3.3.2 Discussion

The Sudret polynomial model is a near neighbor to the tensor monomials model; however, it includes only even-ordered polynomials, which provides more of a challenge to

the adaptive method. 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 standard deviation for this model is that the first-order expansions tend to estimate the variance more accurately than the second-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 is exhibits some cancellation of errors to obtains such an "accurate" result.

3.3.3 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.

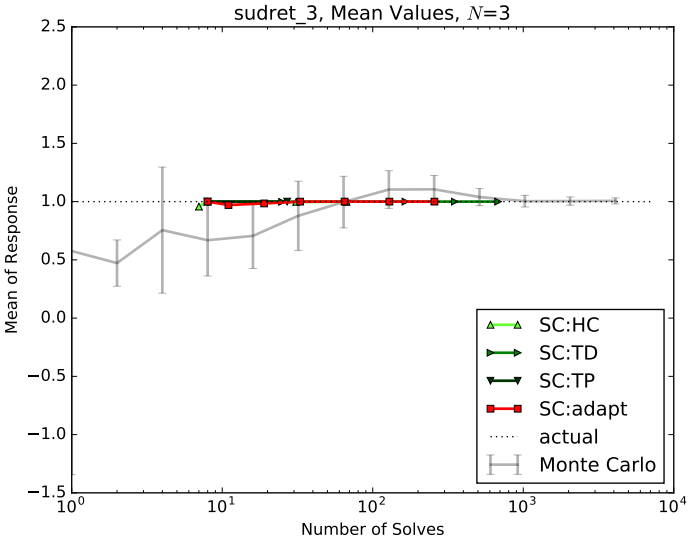

FIGURE 3.15: Sudret Polynomial, $N = 3$, Mean Values

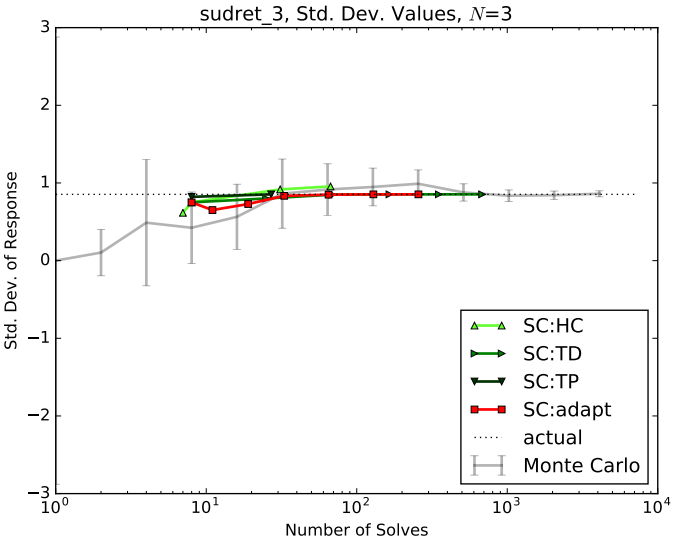

FIGURE 3.16: Sudret Polynomial, $N = 3$, Std. Dev. Values

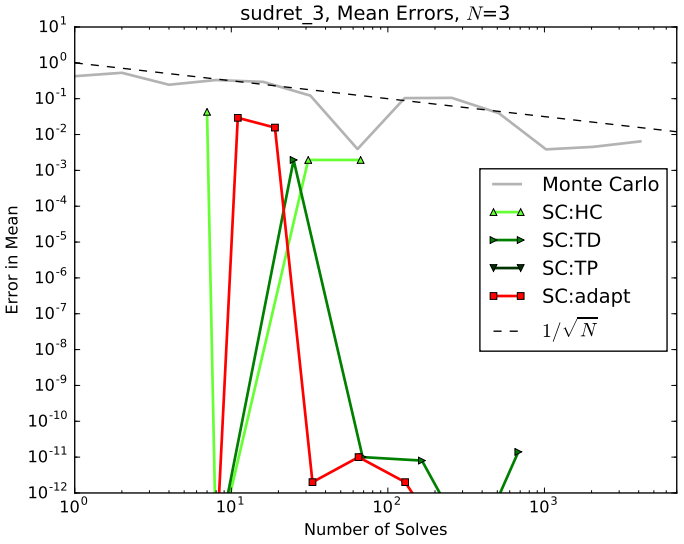

FIGURE 3.17: Sudret Polynomial, $N = 3$, Mean Convergence

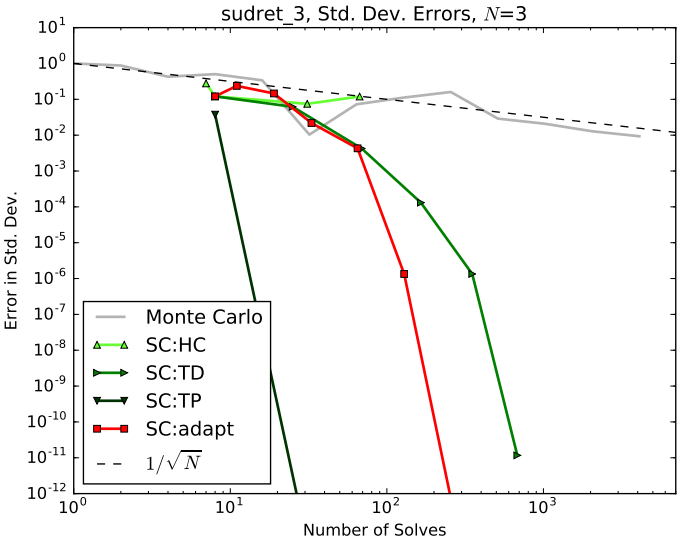

FIGURE 3.18: Sudret Polynomial, $N = 3$, Std. Dev. Convergence

3.3.4 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 Monte Carlo.

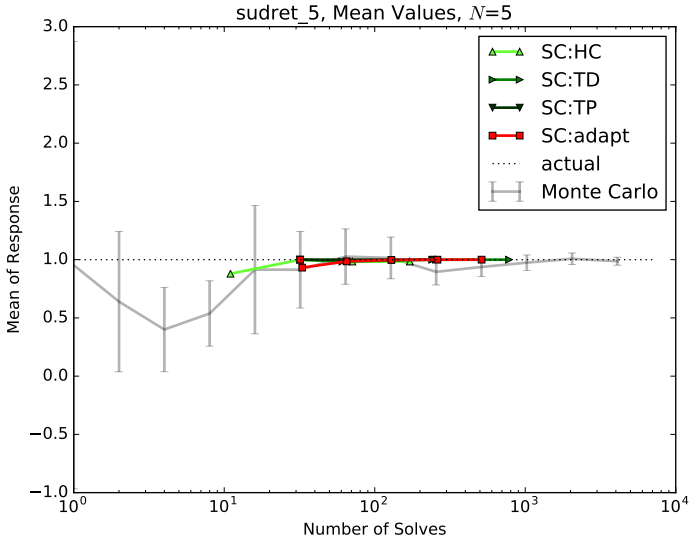

FIGURE 3.19: Sudret Polynomial, $N = 5$, Mean Values

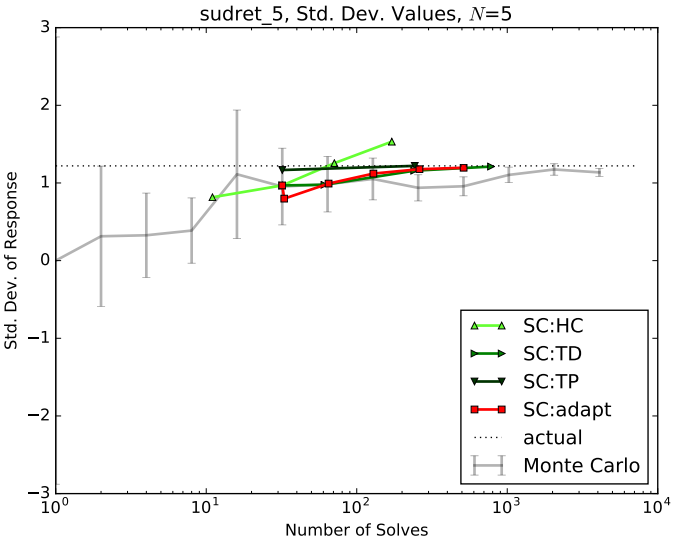

FIGURE 3.20: Sudret Polynomial, $N = 5$, Std. Dev. Values

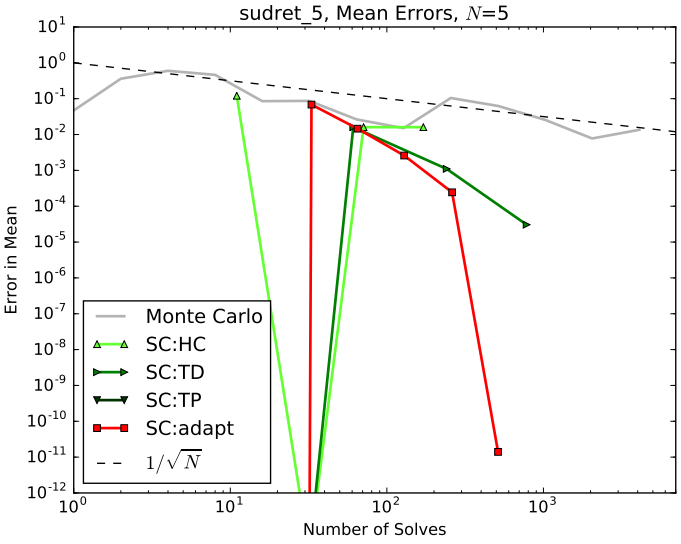

FIGURE 3.21: Sudret Polynomial, $N = 5$, Mean Convergence

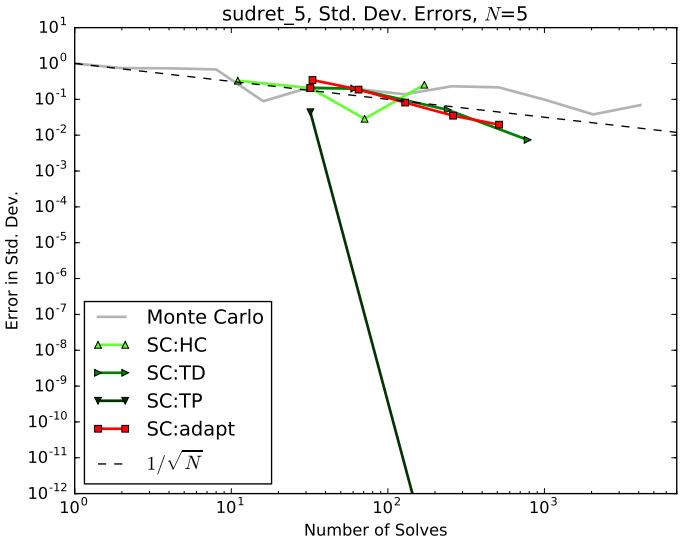

FIGURE 3.22: Sudret Polynomial, $N = 5$, Std. Dev. Convergence

3.4 Attenuation

3.4.1 Description

While this model is a tensor product and analytic, it also is the solution to a 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. 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 boundary on the right boundary.

This model represents an idealized single-dimension system where an beam of particles impinges on a purely-absorbing material with total scaled length of 1. The 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 solution takes the form

$$u(Y) = \prod_{n=1}^{N} \exp(-y_n/N). \tag{3.6}$$

The two-dimensional representation of this function is given in Figure 3.23. Because

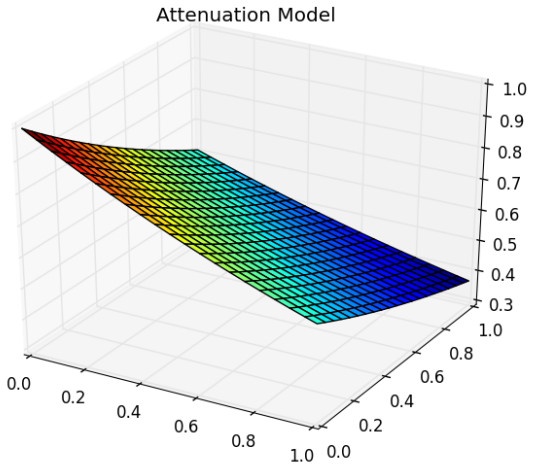

FIGURE 3.23: Attenuation Model

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 3.3.

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 3.3: 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 orders,

$$e^{ay} = 1 - ay + \frac{(ay)^2}{2} - \frac{(ay)^3}{6} + \frac{(ay)^4}{24} - \frac{(ay)^5}{120} + \mathcal{O}(y^6). \tag{3.7}$$

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 $a_1^2 a_2^2/4$, while the coefficient for y_1^4 is $a^4/24$.

3.4.2 Discussion

Similar to the tensor monomial model and Sudret polynomial model, the attenuation model can be expanded as the infinite sum of ever-increasing polynomials, making it tensor product in shape. However, unlike the previous two models, the magnitude of the contribution from the higher-order polynomials decreases swiftly, making lower-order polynomials more characteristic of the model in general. Because this model is still a tensor model and the response is infinitely continuous, we expect to see a similar trend in the most effective methods for polynomial expansion.

3.4.3 2 Inputs

With only two input parameters, we see excellent convergence for all methods for the 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.

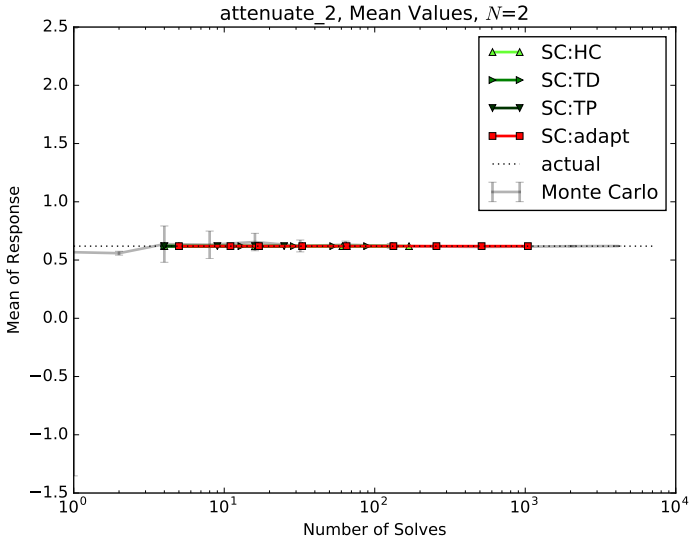

FIGURE 3.24: Attenuation, $N = 2$, Mean Values

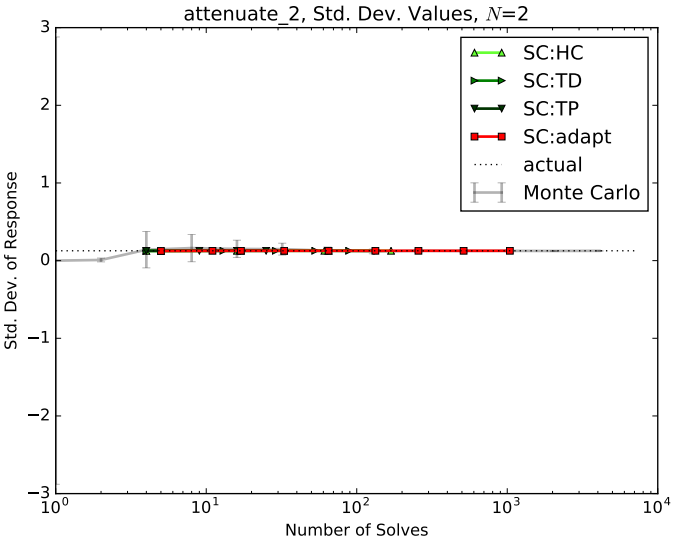

FIGURE 3.25: Attenuation, $N = 2$, Std. Dev. Values

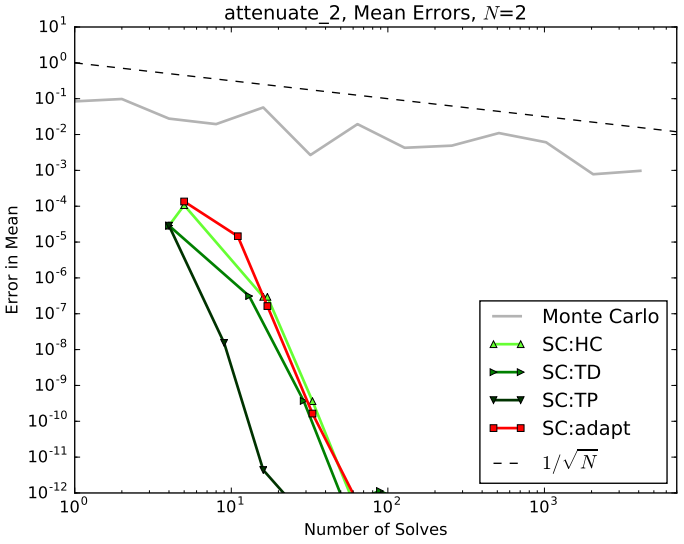

FIGURE 3.26: Attenuation, $N = 2$, Mean Convergence

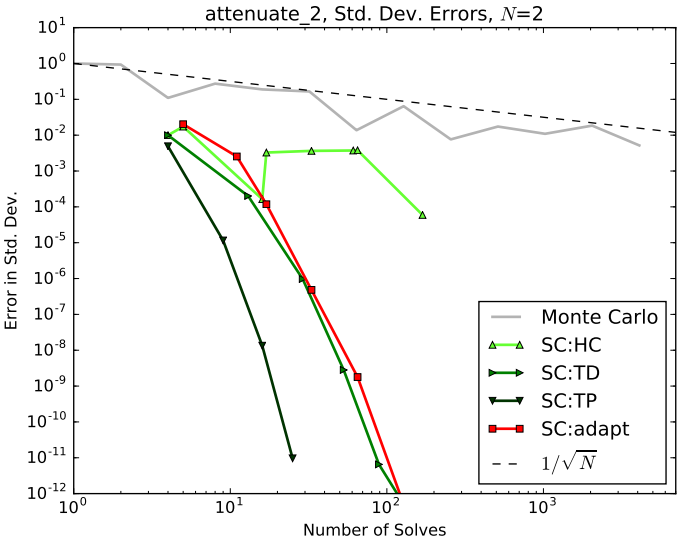

FIGURE 3.27: Attenuation, $N = 2$, Std. Dev. Convergence

3.4.4 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 similarly and converge quite well.

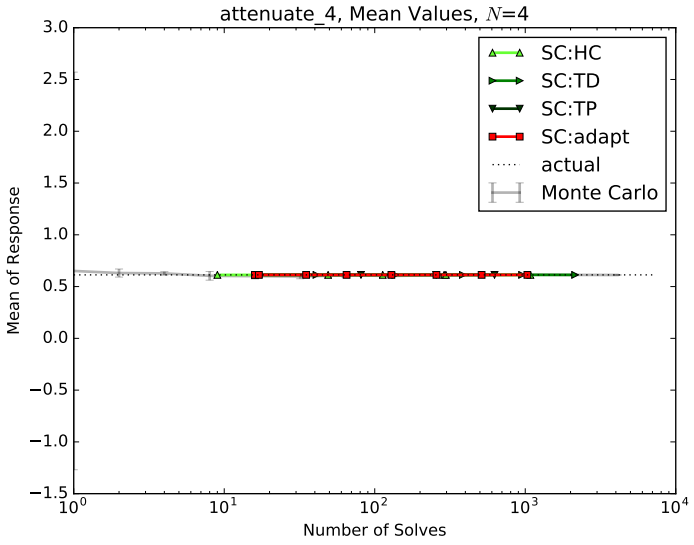

FIGURE 3.28: Attenuation, $N = 4$, Mean Values

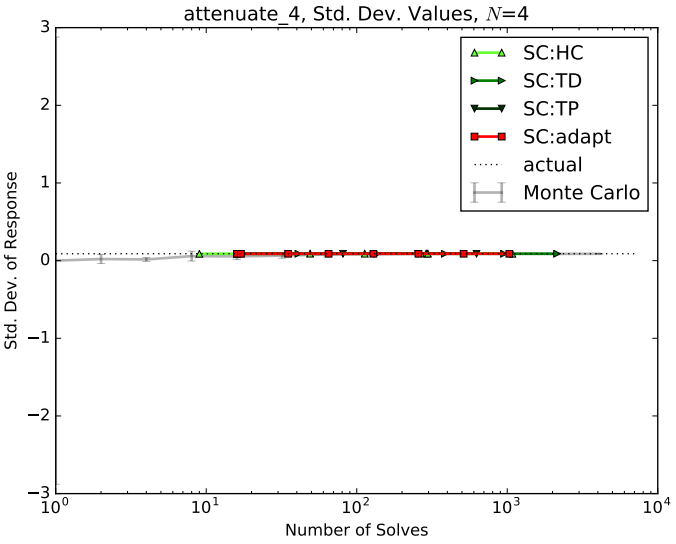

FIGURE 3.29: Attenuation, $N = 4$, Std. Dev. Values

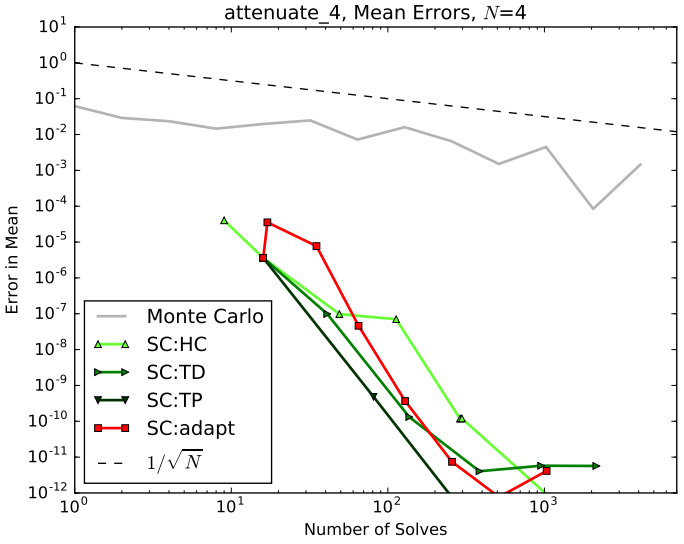

FIGURE 3.30: Attenuation, $N = 4$, Mean Convergence

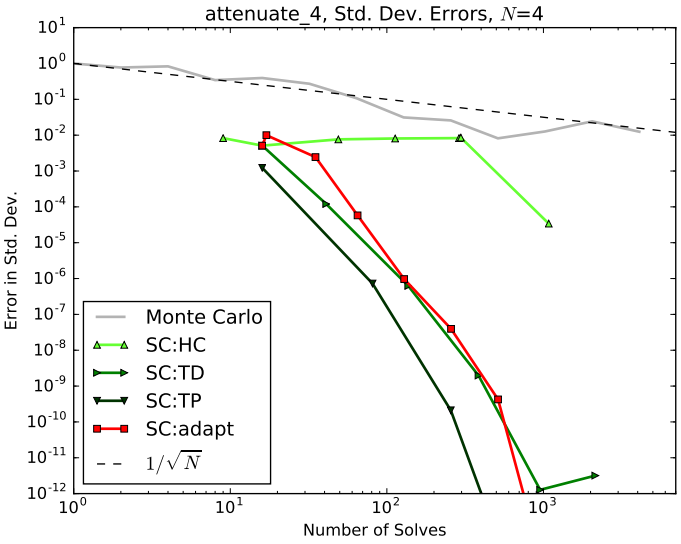

FIGURE 3.31: Attenuation, $N = 4$, Std. Dev. Convergence

3.4.5 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 consider 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.

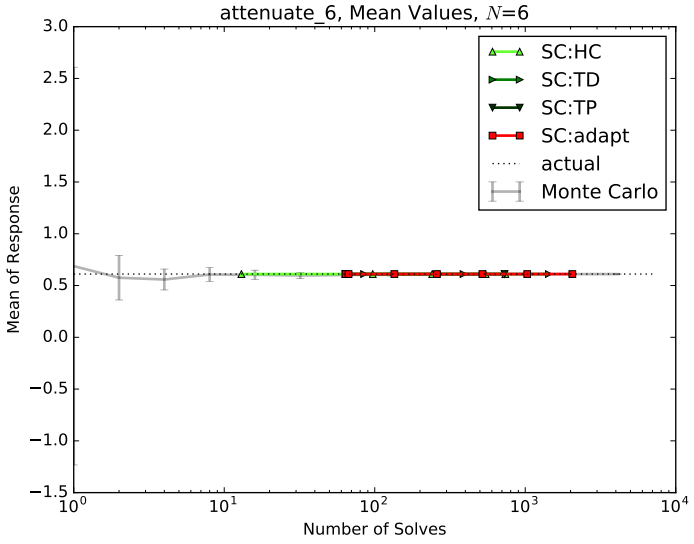

FIGURE 3.32: Attenuation, $N = 6$, Mean Values

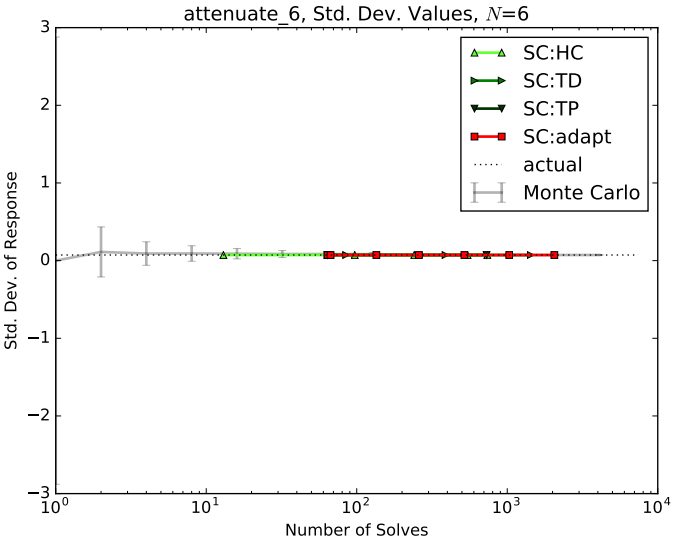

FIGURE 3.33: Attenuation, $N = 6$, Std. Dev. Values

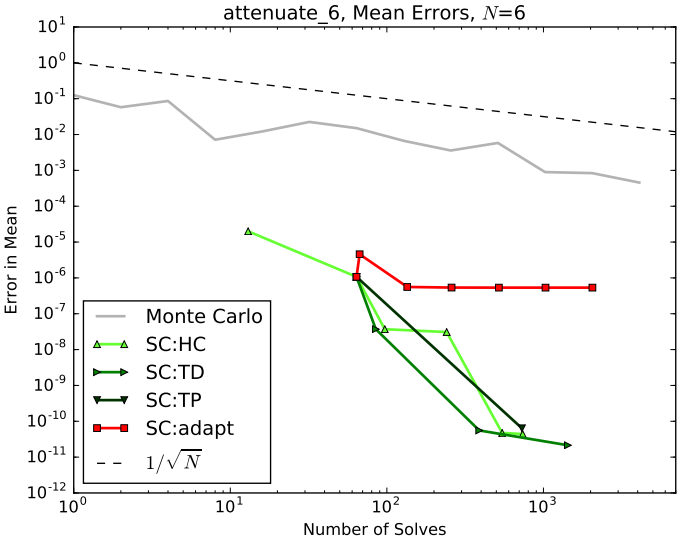

FIGURE 3.34: Attenuation, $N = 6$, Mean Convergence

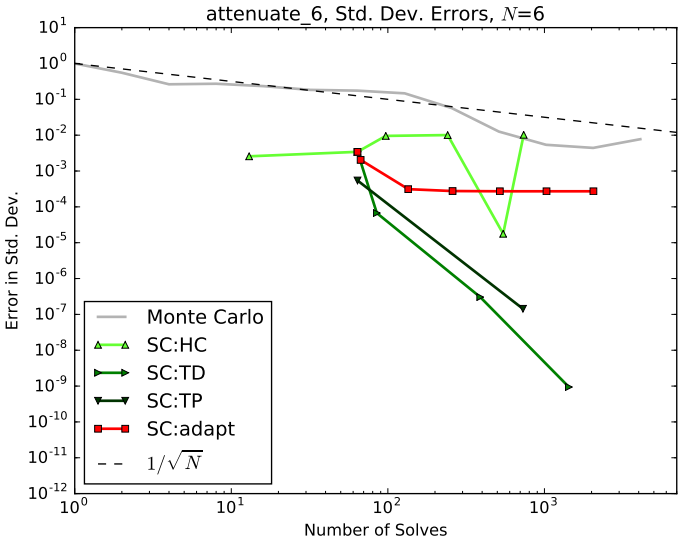

FIGURE 3.35: Attenuation, $N = 6$, Std. Dev. Convergence

3.5 Gauss Peak

3.5.1 Description

Similar to the attenuation model, the Gaussian peak [42] 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

$$u(Y) = \exp\left(-\sum_{n=1}^{N} a^2(y_n - \mu)^2\right). \tag{3.8}$$

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 function is given in Figure 3.36. A summary of analytic statistics is given in Table 3.4.

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 3.4: Analytic Expressions for Gaussian Peak Case

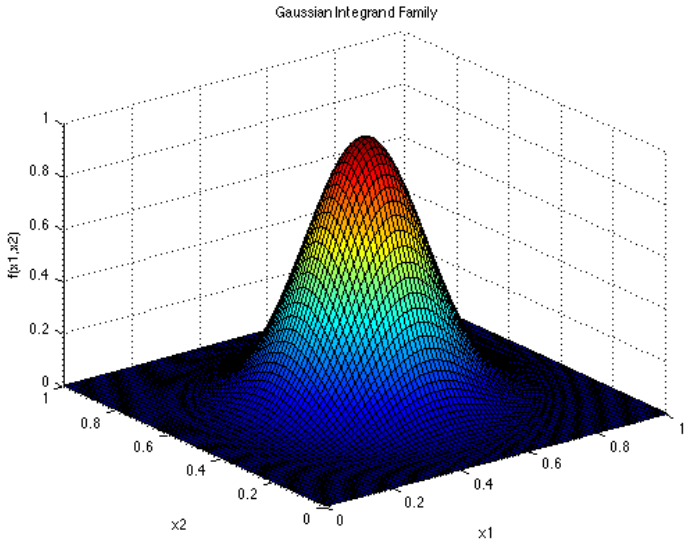

FIGURE 3.36: Gaussian Peak [1]

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

3.5.2 Discussion

As for the attenuation model, this model is best understood in light of its polynomial representation, given by the Taylor expansion (for example for $\mu = 0$ about $y = 0$):

$$e^{-a^2y^2} = 1 - a^2y^2 + \frac{(a^2y^2)^2}{2} - \frac{(a^2y^2)^3}{6} + \frac{(a^2y^2)^4}{24} - \frac{(a^2y^2)^5}{120} + \mathcal{O}(y^6). \tag{3.9}$$

Because this model's coefficient $a = 3$, it serves to balance against the cross-term preference in the product of two terms, instead of exacerbating it as in the attenuation model. Because of this, terms with the same total effective polynomial order are more nearly equal in importance. In addition, because of the peaking term, more polynomials are required to accurately represent this model. As a result, we observe errors to be larger for all methods, and the tensor product to have reduced error in comparison to the other methods, all when compared to the attenuation model.

3.5.3 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 for the standard deviation even with only three inputs. This demonstrates the challenge of this model to be represented well with polynomials.

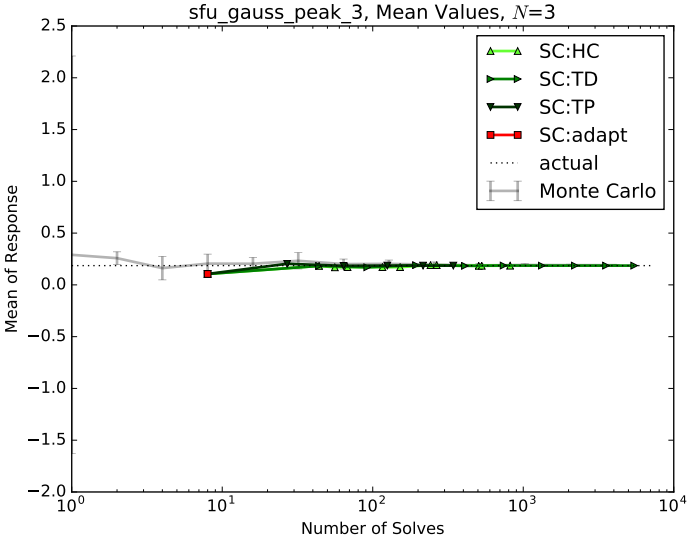

FIGURE 3.37: Gauss Peak, $N = 3$, Mean Values

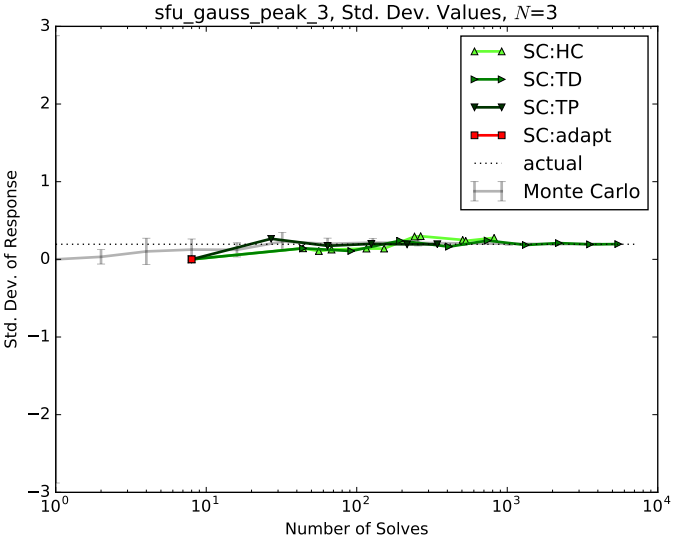

FIGURE 3.38: Gauss Peak, $N = 3$, Std. Dev. Values

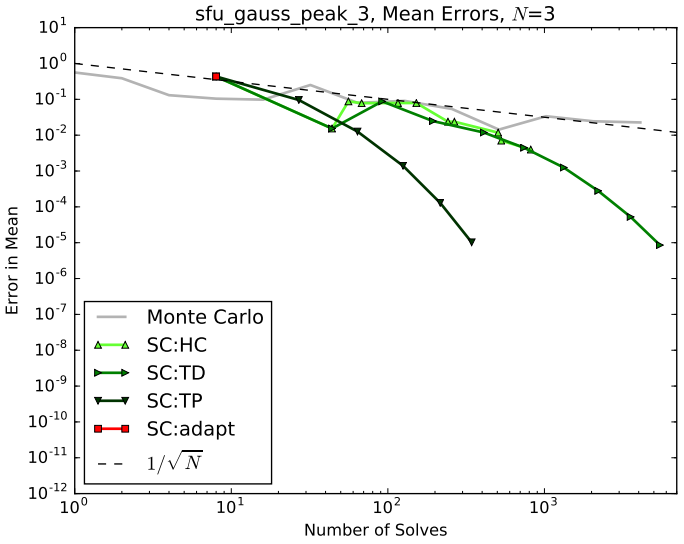

FIGURE 3.39: Gauss Peak, $N = 3$, Mean Convergence

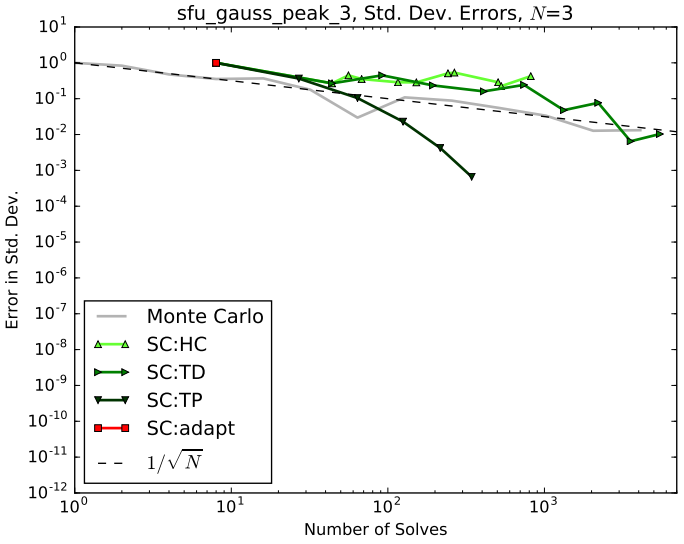

FIGURE 3.40: Gauss Peak, $N = 3$, Std. Dev. Convergence

3.5.4 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 convergence benefits in the collocation methods, for up to 1000 computation solves there is little advantage over Monte Carlo sampling.

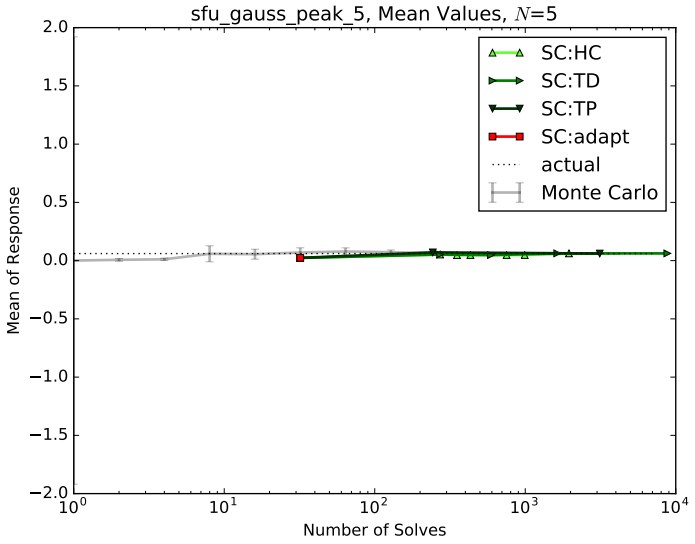

FIGURE 3.41: Gauss Peak, $N = 5$, Mean Values

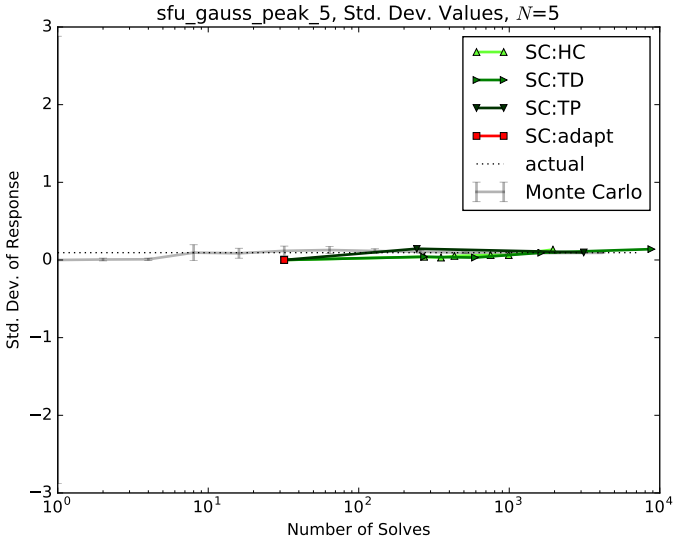

FIGURE 3.42: Gauss Peak, $N = 5$, Std. Dev. Values

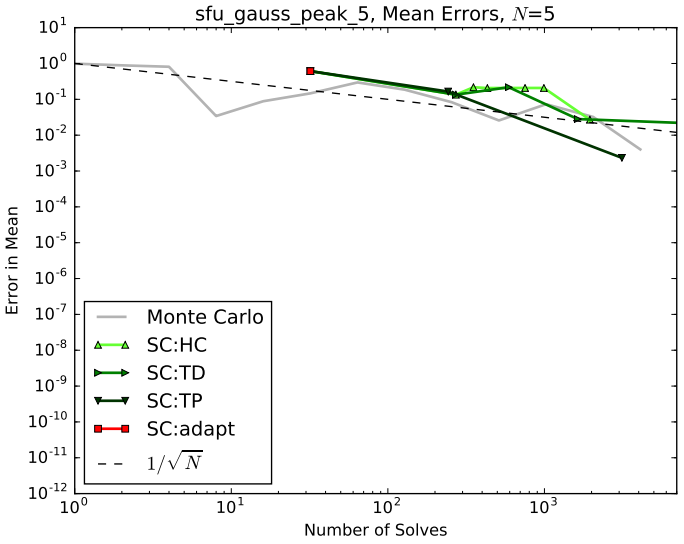

FIGURE 3.43: Gauss Peak, $N = 5$, Mean Convergence

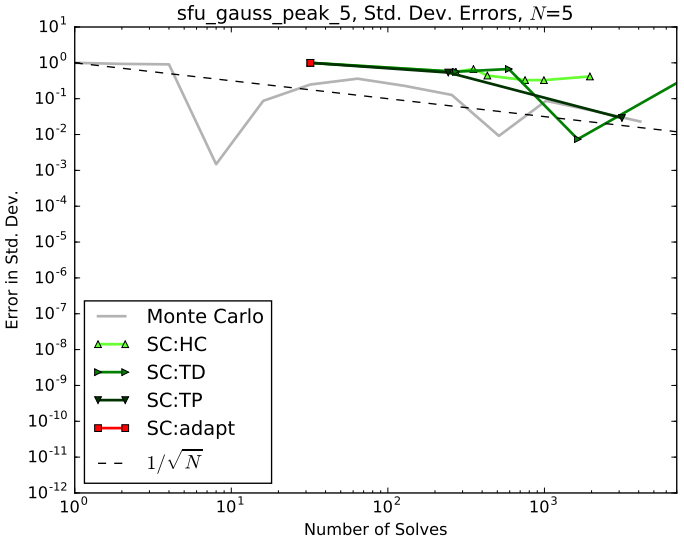

FIGURE 3.44: Gauss Peak, $N = 5$, Std. Dev. Convergence

3.6 Ishigami

3.6.1 Description

The Ishigami function [43] is a commonly-used function in performing sensitivity analysis. It is given by

$$u(Y) = \sin y_1 + a \sin^2 y_2 + b y_3^4 \sin(y_1). \tag{3.10}$$

In our case, we will use $a = 7$ and $b = 0.1$ as in [44]. The graphical representation of this function is given in Figure 3.45, 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

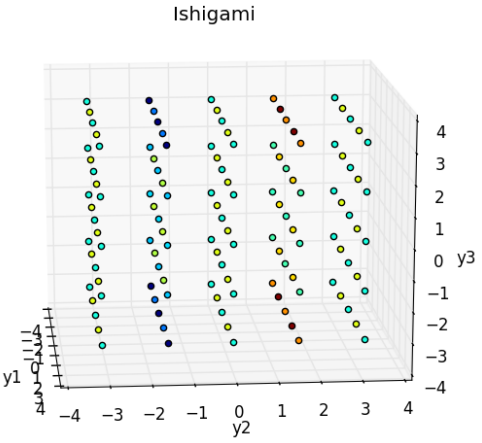

FIGURE 3.45: Ishigami Model

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 3.5, where D_n is the partial variance contributed by y_n and Sobol sensitivities $\mathcal{S}_n$ are obtained by dividing D_n by the total variance.

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

TABLE 3.5: Analytic Expressions for Ishigami Case

3.6.2 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.

3.6.3 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 Monte Carlo are not distinct until near 1000 computational solves, despite the small input space.

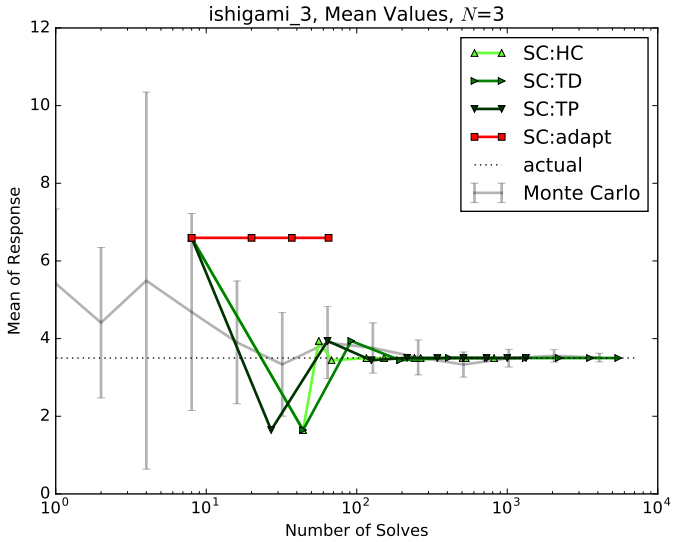

FIGURE 3.46: Ishigami, $N = 3$, Mean Values

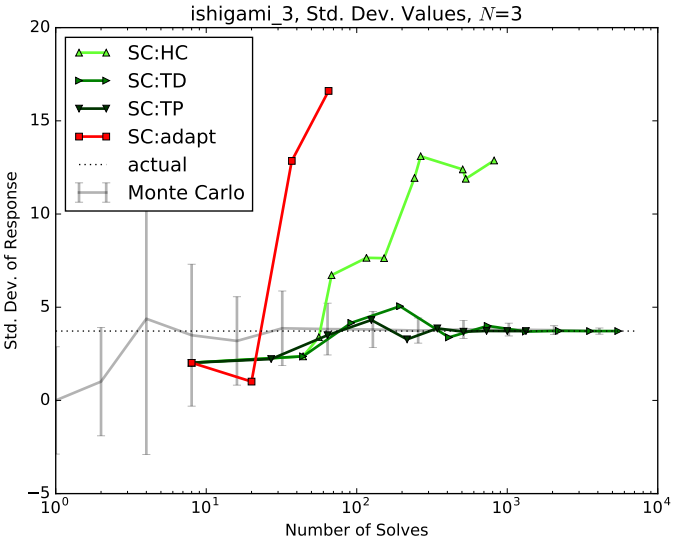

FIGURE 3.47: Ishigami, $N = 3$, Std. Dev. Values

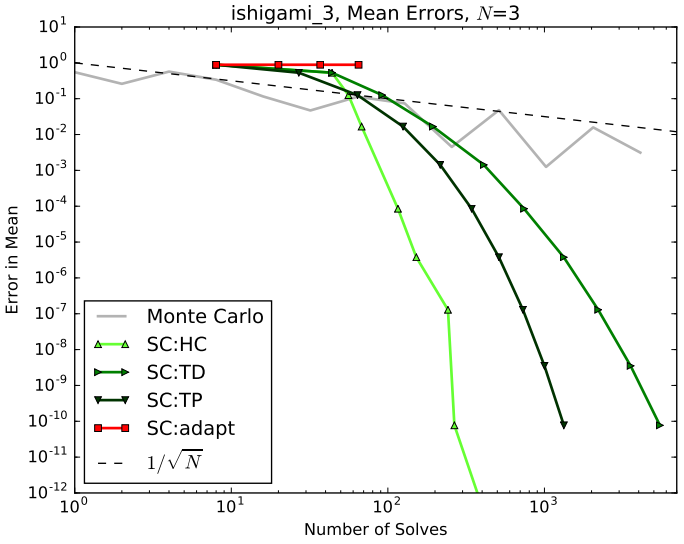

FIGURE 3.48: Ishigami, $N = 3$, Mean Convergence

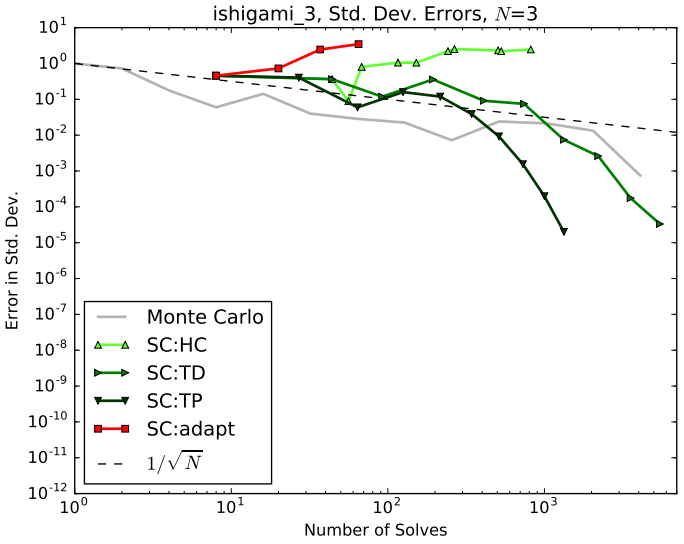

FIGURE 3.49: Ishigami, $N = 3$, Std. Dev. Convergence

3.7 Sobol G-Function

3.7.1 Description

The so-called "g-function" introduced by Saltelli and Sobol [46] 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 [47]. The function is given by

$$u(Y) = \prod_{n=1}^{N} \frac{|4y_n - 2| - a_n}{1 + a_n}, \tag{3.11}$$

where

$$a_n = \frac{n-2}{2}. \tag{3.12}$$

The two-dimensional representation of this function is given in Figure 3.50.

There are some implementations [47] 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.

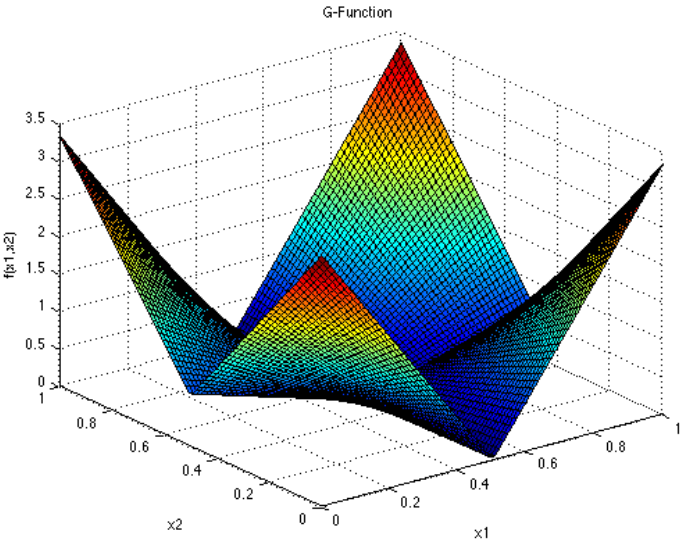

FIGURE 3.50: Sobol G-Function [1]

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.

3.7.2 Discussion

As expected, even for 3 input variables this discontinuous model provides a difficult challenges for polynomial representations. Even at thousands of computation solves, there is no discernible benefit in using collocation methods over traditional Monte Carlo.

3.7.3 3 Inputs

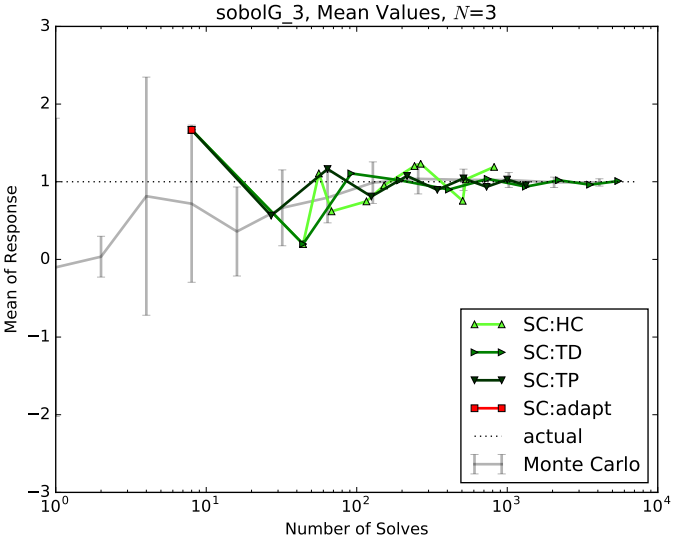

FIGURE 3.51: Sobol G-Function, $N = 3$, Mean Values

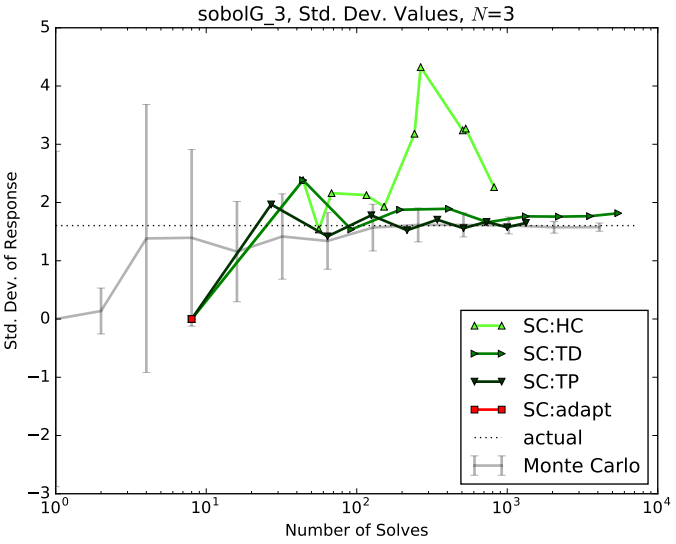

FIGURE 3.52: Sobol G-Function, $N = 3$, Std. Dev. Values

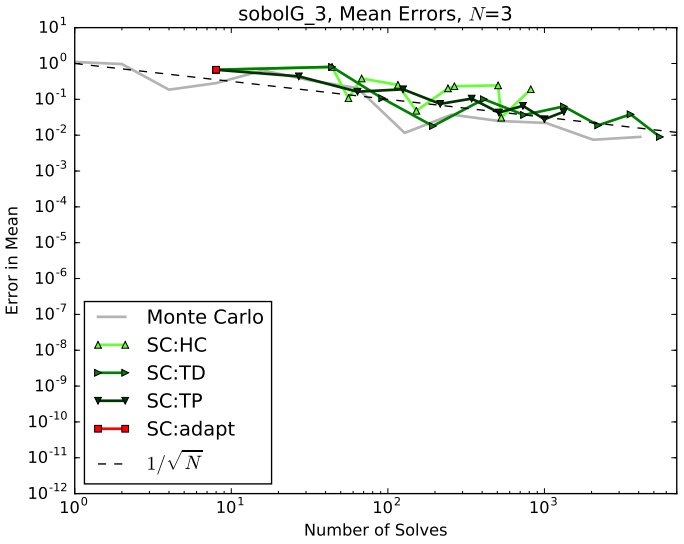

FIGURE 3.53: Sobol G-Function, $N = 3$, Mean Convergence

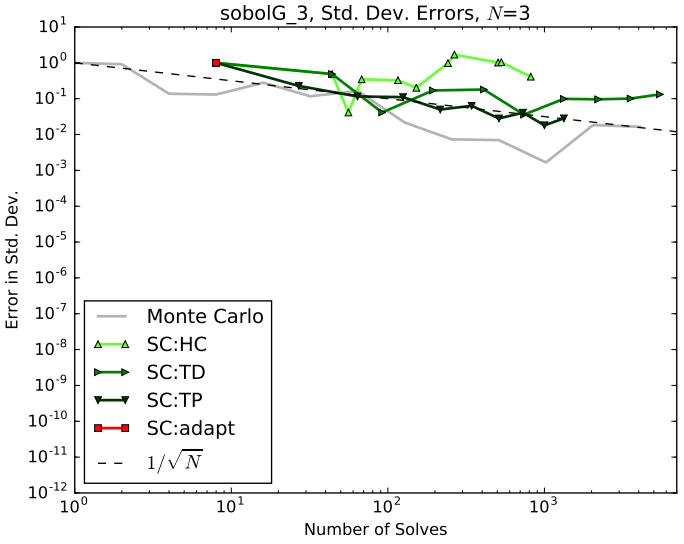

FIGURE 3.54: Sobol G-Function, $N = 3$, Std. Dev. Convergence

3.7.4 5 Inputs

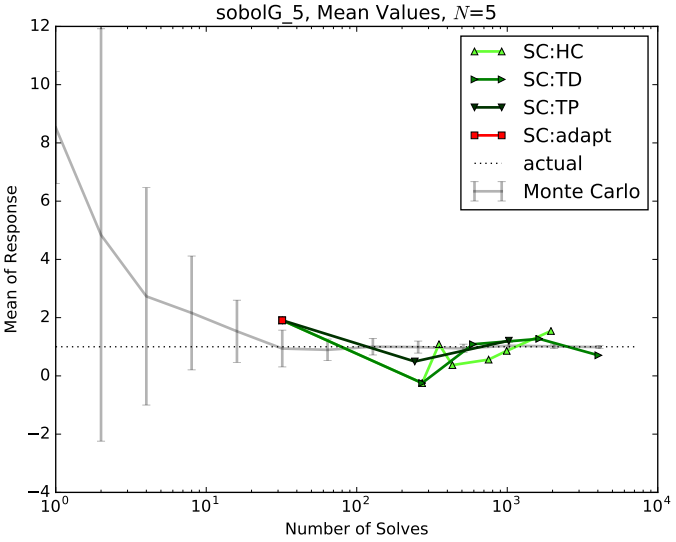

FIGURE 3.55: Sobol G-Function, $N = 5$, Mean Values

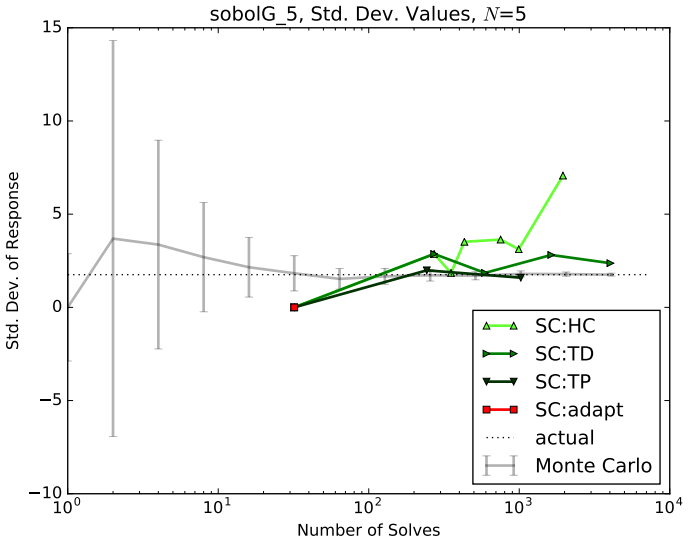

FIGURE 3.56: Sobol G-Function, $N = 5$, Std. Dev. Values

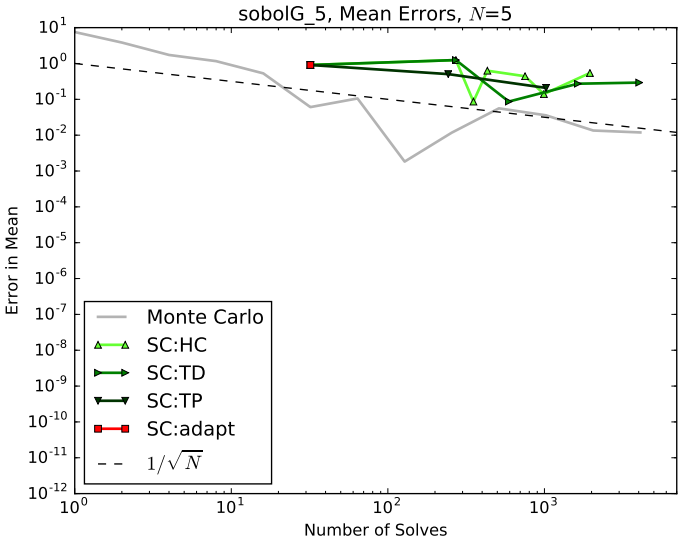

FIGURE 3.57: Sobol G-Function, $N = 5$, Mean Convergence

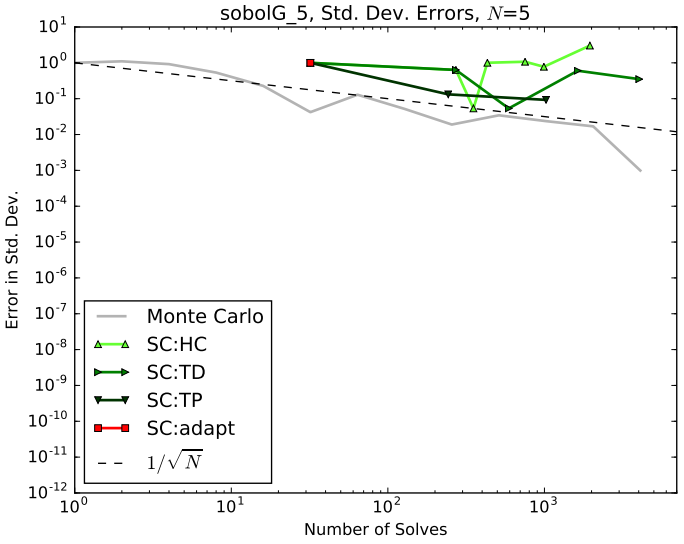

FIGURE 3.58: Sobol G-Function, $N = 5$, Std. Dev. Convergence

3.8 Conclusions

todo

Chapter 4

Methods: High-Density Model Reduction

4.1 Introduction

TODO

4.2 High-Dimension Model Representation (HDMR)

While using SCgPC is one method for creating a reduced-order model for a simulation code $u(Y)$, another useful model reduction is HDMR[32], sometimes known as Sobol decomposition because the expansion is conducive to easily determining Sobol sensitivity coefficients. HDMR is an ANalysis Of VAriance (ANOVA) method.

In general, the HDMR expansion involves the sum of several terms, each of which only depends on a subset of the full input space. The subsets are developed by integrating out the undesired dimensions. Letting $H(Y)$ represent the untruncated HDMR expansion of $u(Y)$,

$$u(Y) = H(Y) = h_0 + \sum_{n=1}^{N} h_n + \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} h_{n_1,n_2} + \cdots + h_{1,2,\ldots,N}, \tag{4.1}$$

where the expectation value h_0 is given by

$$h_0 \equiv \int_{\Omega_1} \rho_1(y_1) \ldots \int_{\Omega_N} \rho_N(y_N) u(y_1, \ldots, y_N)\ dy_1 \ldots\ dy_N, \tag{4.2}$$

$$= \int_{\Omega} \rho(Y) u(Y) dY, \tag{4.3}$$

where Ω denotes the uncertainty space spanned by Y and $\rho(Y)$ is the multidimensional probability distribution function of Y. The first-order expansion terms h_n are integrated as

$$h_n(y_n) \equiv \int_{\hat{\Omega}_n} \hat{\rho}_n(\hat{Y}_n) u(Y)\ d\hat{Y}_n - h_0, \tag{4.4}$$

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

$$\hat{Y}_n \equiv (y_1, \cdots, y_{n-1}, y_{n+1}, \cdots, y_N), \tag{4.5}$$

$$\hat{Y}_{m,n} \equiv (y_1, \cdots, y_{m-1}, y_{m+1}, \cdots, y_{n-1}, y_{n+1}, \cdots, y_N). \tag{4.6}$$

Second and higher-order HDMR expansion terms are defined as

$$h_{n_1,n_2}(y_{n_1}, y_{n_2})) \equiv \int_{\hat{\Omega}_{n_1,n_2}} \hat{\rho}_{n_1,n_2}(\hat{Y}_{n_1,n_2}) u(Y)\ d\hat{Y}_{n_1,n_2} - h_{n_1} - h_{n_2} - h_0, \tag{4.7}$$

and so on.

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 $h_1, (h_1, h_2), \cdots, (h_1, h_2, h_3)$, etc.

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

$$\int_{\Omega} h_a h_b dY = 0 \quad \forall\ a \neq b. \tag{4.8}$$

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,

$$\int_{\Omega_n} H(Y)^2 dy_n = \int_{\Omega_n} h_n^2 dy_n. \tag{4.9}$$

This in turn 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,

$$\mathcal{S}_n \equiv \frac{\text{var}[h_n]}{\text{var}[H(Y)]}, \tag{4.10}$$

$$\mathcal{S}_{m,n} \equiv \frac{\text{var}[h_{m,n}]}{\text{var}[H(Y)]}, \tag{4.11}$$

and so on.

4.2.1 Cut-HDMR

The primary challenge in implementing HDMR for arbitrary responses is the integrals in Eq. 4.2, 4.4, and 4.7. These integrals are of a higher dimensionality than those required for generalized polynomial chaos expansions. As a result, at first glance HDMR seems to offer no benefits over gPC. However, we make use of an approximation for HDMR called *cut-HDMR* [14] 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 $\bar{Y} = (\bar{y}_1, \bar{y}_2, \cdots, \bar{y}_N)$. 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 $T(Y)$ is expressed as

$$u(Y) = T(Y) = t_r + \sum_{n=1}^{N} t_n + \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} t_{n_1,n_2} + \cdots + t_{1,2,\ldots,N}. \tag{4.12}$$

Eq. 4.17 is identical in form to the traditional ANOVA HDMR expansion, but the subset components are defined differently,

$$t_r \equiv u(\bar{Y}), \tag{4.13}$$

$$t_n(y_n) \equiv u(y_n, \hat{\bar{Y}}_n) - t_r, \tag{4.14}$$

$$t_{m,n}(y_m, y_n) \equiv u(y_m, y_n, \hat{Y}_{m,n}) - t_m - t_n - t_r, \tag{4.15}$$

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:

$$\hat{\bar{Y}}_n = (\bar{y}_1, \cdots, \bar{y}_{n-1}, \bar{y}_{n+1}, \cdots, \bar{y}_N). \tag{4.16}$$

In the limit where each subset of cut-HDMR is at most linearly dependent on an input parameter, cut-HDMR and ANOVA are exact. Additionally, if all the cut-HDMR terms are kept, it converges exactly on ANOVA.

The immediate benefit from cut-HDMR is the ability to computationally calculate the terms in the expansion; we only need the reference input realization $\bar{Y}$ to construct the expansion. However, one drawback to cut-HDMR is that is component terms are not orthogonal, unlike ANOVA. This results in difficulty when attempting to algorithmically

determine statistical moments. Fortunately, this will be resolved in section **??**. First, however, we consider how to represent the subset terms in the HDMR expansion.

4.2.2 gPC and cut-HDMR

Consider the cut-HDMR expansion,

$$u(Y) = T(Y) = t_r + \sum_{n=1}^{N} t_n + \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} t_{n_1,n_2} + \cdots + t_{1,2,\ldots,N}, \tag{4.17}$$

with subsets t defined in section 4.2.1. 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, sensitivities, and other measures. In particular these subsets can be represented by generalized polynomial chaos expansions

$$t_n \approx \sum_{k' \in \Lambda'(L')} t_{n;k'} \Phi_{k'}(Y_n), \tag{4.18}$$

where we make use of prime notation k', Λ', L' to denote generalized polynomial chaos expansion for a subset term of a cut-HDMR expansion and $t_{n,k'}$ are the scalar expansion coefficients. Eq. 4.17 can then be written

$$\begin{aligned} T(Y) \approx t_r &+ \sum_{n=1}^{N} \left(\sum_{k' \in \Lambda'_n(L')} t_{n;k'} \Phi_{k'}(Y_n) \right) \\ &+ \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} \left(\sum_{k' \in \Lambda'_{m,n}(L')} t_{m,n;k'} \Phi_{k'}(Y_m, Y_n) \right) \\ &+ \cdots \\ &+ \left(\sum_{k' \in \Lambda'_{1,\cdots,n}(L')} t_{1,\cdots,n;k'} \Phi_{k'}(Y_1, \cdots, Y_n) \right). \end{aligned} \tag{4.19}$$

There are several synergies and advantages to using generalized polynomial chaos to expand the subset terms in the cut-HDMR expansion as in Eq. 4.19. First, the scalar expansion coefficients can be calculated using the same collocation-based methods developed for the stochastic collocation for generalize polynomial chaos method. As we demonstrate in section **??**, 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. 4.17. 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 all dimensionality three or smaller, which is ideal size for exceedingly efficient convergence of stochastic collocation for generalized polynomial chaos methods.

In addition, polynomial expansion methods are most efficient when the response is continuous. Whatever the continuity of the model, the continuity of the subsets in the HDMR expansion of that model are always at least as 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 stochastic collocation for generalized polynomial chaos can without using cut-HDMR for discontinuous responses.

Second, generalized polynomial expansion 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 will explore this further in section 4.2.4.

4.2.3 On convergence of gPC and cut-HDMR with gPC

We pause momentarily to make a note about converging stochastic collocation for generalized polynomial chaos expansion methods alone versus using SCgPC as part of a cut-HDMR expansion. There are two adjustments that can be made to static generalized polynomial chaos 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 adjustment is the polynomial order limit L. For cut-HDMR, however, we add another adjustment tool to the previous two: Sobol truncation level, or the maximum dimensionality of any subset in the cut-HDMR expansion.

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 generalized polynomial chaos 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 pure generalized polynomial chaos expansion. Any truncation of the cut-HDMR yields an approximation to the pure generalized polynomial expansion. As a result, for a given polynomial order limit, cut-HDMR can at most match the convergence of the

corresponding generalized polynomial chaos expansion. Additionally, the cut-HDMR will use a very similar number of numerical evaluations to obtain that same level of convergence.

However, the real benefit of cut-HDMR is seen in models with large input dimensionality. In this case, even a first-order generalized polynomial chaos expansion method 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 accrued by truncating cut-HDMR may be worth the reduction in necessary evaluations.

4.2.4 Reconstructing ANOVA from cut-HDMR

When using gPC to represent individual cut-HDMR subsets, it is simple to 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 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. 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.

To reconstruct the ANOVA decomposition of a cut-HDMR expansion, we simply apply ANOVA to the cut-HDMR expansion, which will results in significant reduction. We begin with the cut-HDMR expansion with subsets determined by generalized polynomial chaos expansions by repeating Eq. 4.19,

$$T(Y) \approx t_r + \sum_{n=1}^{N} \left(\sum_{k' \in \Lambda'_n(L')} t_{n;k'} \Phi_{k'}(Y_n) \right) + \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} \left(\sum_{k' \in \Lambda'_{m,n}(L')} t_{m,n;k'} \Phi_{k'}(Y_m, Y_n) \right), \tag{4.20}$$

and recall the definition of ANOVA in Eq. 4.1, 4.2, 4.4, and 4.7. For demonstration, note we truncate the cut-HDMR to second-order effects in Eq. 4.20, but the concepts extend to higher-order truncations trivially. To further simplify, we consider a three-dimension input space for $T(Y) = T(x, y, z)$, which again can be extended trivially 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 $G(y_n)$,

$$T(y_n, \hat{\hat{Y}}_n) \approx G(y_n) = \sum_{k' \in \Lambda'_n(L')} t_{n;k'} \Phi_{k'}(Y_n), \tag{4.21}$$

so that

$$t_n(y_n) = T(y_n, \hat{\hat{Y}}_n) - t_r \approx G(y_n) - t_r. \tag{4.22}$$

Eq. 4.20 then becomes

$$T(x, y, z) = t_r + t_x + t_y + t_z + t_{xy} + t_{xz} + t_{yz}, \tag{4.23}$$

with the following definitions:

$$t_r = T(\bar{x}, \bar{y}, \bar{z}), \tag{4.24}$$

$$t_x = T(x, \bar{y}, \bar{z}) - t_r \approx G(x) - t_r, \tag{4.25}$$

$$t_y = T(\bar{x}, y, \bar{z}) - t_r \approx G(y) - t_r, \tag{4.26}$$

$$t_z = T(\bar{x}, \bar{y}, z) - t_r \approx G(z) - t_r, \tag{4.27}$$

$$t_{xy} = T(x, y, \bar{z}) - t_x - t_y - t_r \approx G(x, y) - t_x - t_y - t_r, \tag{4.28}$$

$$t_{xz} = T(x, \bar{y}, z) - t_x - t_z - t_r \approx G(x, z) - t_x - t_z - t_r, \tag{4.29}$$

$$t_{yz} = T(\bar{x}, y, z) - t_y - t_z - t_r \approx G(y, z) - t_y - t_z - t_r. \tag{4.30}$$

Substituting and collecting terms,

$$T(x, y, z) \approx t_r - G(x) - G(y) - G(z) + G(x, y) + G(x, z) + G(y, z), \tag{4.31}$$

where the approximation depends entirely on the ability of generalized polynomial chaos expansions to represent each subset space. In the limit that infinite polynomials are available, the equation becomes exact.

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

$$\int_{\Omega_n} f(y_n) dy_n = \int_{a_n}^{b_n} \rho_n(y_n) f(y_n) dy_n, \tag{4.32}$$

$$\int_{\Omega} f(Y) dY = \int_{a_1}^{b_1} \cdots \int_{a_N}^{b_N} \rho(y_1, \cdots, y_N) f(y_1, \cdots, y_N) dy_1 \cdots, dy_N, \tag{4.33}$$

which simplifies the notation considerably.

The first term in ANOVA, the expectation value h_0, is given as

$$h_0 = \int_{\Omega} \rho(Y)T(Y)dY, \tag{4.34}$$

which expands into the sum of individual integrals

$$\begin{aligned} h_0 =& t_r \\ &- \int_{\Omega_x} G(x)dx - \int_{\Omega_y} G(y)dy - \int_{\Omega_z} G(z)dz \\ &+ \int_{\Omega_{x,y}} G(x,y)dxdy + \int_{\Omega_{x,z}} G(x,z)dxdz + \int_{\Omega_{y,z}} G(y,z)dydz, \end{aligned} \tag{4.35}$$

recalling that by definition

$$\int_{\Omega_n} \rho_n(y_n)dy_n = 1. \tag{4.36}$$

Also recalling the nature of the orthonormal polynomials families in the generalized polynomial chaos expansions,

$$\int_{\Omega_n} \phi_{k_n}(y_n)dy_n = \delta_{k_n,0}, \tag{4.37}$$

all nonzero polynomial terms integrate to zero,

$$\int_{\Omega_x} G(x)dx = \int_{\Omega_x} \sum_{k' \in \Lambda'} c_{k'}\Phi_{k'}(x)dx = c_{\varnothing}^{(x)}, \tag{4.38}$$

$$\int_{\Omega_{x,y}} G(x,y)dxdy = \int_{\Omega_{x,y}} \sum_{k' \in \Lambda'} c_{k'}\Phi_{k'}(x,y)dxdy = c_{\varnothing}^{(x,y)}, \tag{4.39}$$

where we use the parenthetical superscript to denote the subset origin of the scalar coefficients and the subscript $\varnothing$ to indicate $k = 0, 0, 0$. Because of symmetry, the same results are obtained for subsets (y) and (z) as for subset (x), and the same results are obtained for subsets (x, z) and (y, z) as for subset (x, y). As a result, the zeroth-order ANOVA term is

$$h_0 = t_r - c_{\varnothing}^{(x)} - c_{\varnothing}^{(y)} - c_{\varnothing}^{(z)} + c_{\varnothing}^{(x,y)} + c_{\varnothing}^{(x,z)} + c_{\varnothing}^{(y,z)}, \tag{4.40}$$

or simply the zeroth polynomial order contribution terms from each subset.

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

$$\begin{aligned} h_x &= \int_{\Omega_{y,z}} T(x,y,z)\ dy\ dz - h_0, \\ &= t_r - G(x) - \int_{\Omega_y} G(y)\ dy - \int_{\Omega_z} G(z)\ dz + \int_{\Omega_y} G(x,y)\ dy + \int_{\Omega_z} G(x,z)\ dz \\ &\quad + \int_{\Omega_{y,z}} G(y,z)\ dy\ dz - h_0. \end{aligned} \tag{4.41}$$

For the integrals, for example,

$$\begin{aligned} \int_{\Omega_y} G(x,y)\ dy &= \int_{\Omega_y} \sum_{k\in\Lambda} c_k \Phi_k(x,y)\ dx, \\ &= \left\{ \begin{array}{ll} 0, & k_y \geq 1, \\ c_{(k_x,0)}\phi_{k_x}(x), & k_y = 0, \end{array} \right\} \\ &= \sum_{\substack{k\ \in\ \Lambda \\ k_y\ =\ 0}} c_k \phi_{k_x}(x), \end{aligned} \tag{4.42}$$

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

$$\begin{aligned} h_x &= t_r - G(x) - c_\varnothing^{(y)} - c_\varnothing^{(z)} + \sum_{\substack{k\ \in\ \Lambda \\ k_y\ =\ 0}} c_k^{(xy)} \phi_{k_x}(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_z\ =\ 0}} c_k^{(xz)} \phi_{k_x}(x) + c_\varnothing^{(yz)} - h_0, \\ &= c_\varnothing^{(x)} - G(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_y\ =\ 0}} c_k^{(xy)} \phi_{k_x}(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_z\ =\ 0}} c_k^{(xz)} \phi_{k_x}(x) - c_\varnothing^{(xy)} - c_\varnothing^{(xz)}, \\ &= - \sum_{\substack{k\ \in\ \Lambda \\ k_x\ >\ 0}} c_k^{(x)} \Phi_k(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_x\ >\ 0 \\ k_y\ =\ 0}} c_k^{(xy)} \phi_{k_x}(x) + \sum_{\substack{k\ \in\ \Lambda \\ k_x\ >\ 0 \\ k_z\ =\ 0}} c_k^{(xz)} \phi_{k_x}(x). \end{aligned} \tag{4.43}$$

Note that all the terms in Eq. 4.43 are elements from each polynomial set where the only nonzero polynomial orders are those with respect to x. Because of the symmetry in the cut-HDMR expansion, the procedure and results for h_y and h_z will be identical in form to h_x.

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

$$\begin{aligned} h_{x,y} &= \int_{\Omega_z} T(x,y,z)\ dz - h_x - h_y - h_0, \\ &= t_r - G(x) - G(y) - c_{\varnothing}^{(z)} + G(x,y) + \sum_{\substack{k\in\Lambda \\ k_z=0}} c_k^{(xz)}\phi_{k_x}(x) + \sum_{\substack{k\in\Lambda \\ k_z=0}} c_k^{(yz)}\phi_{k_y}(y) \\ &\quad - h_x - h_y - h_0, \\ &= \sum_{\substack{k\in\Lambda \\ k_x>0 \\ k_y>0}} c_k\Phi_k(x,y). \end{aligned} \tag{4.44}$$

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. Also, symmetry allows the form for both $h_{x,z}$ and $h_{y,z}$ to be the same as $h_{x,y}$.

With all of the ANOVA terms calculated, it is possible to obtain moments of the cut-HDMR expansion using them. The expected value is trivial, as it is just the zeroth-order ANOVA term

$$\mathbb{E}[H[T](x,y,z)] = h_0. \tag{4.45}$$

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 terms.

$$\mathbb{E}[H[T](x,y,z)^2] = h_0^2 + \int_{\Omega} h_x^2 + h_y^2 + h_z^2 + h_{x,y}^2 + h_{x,z}^2 + h_{y,z}^2 dxdydz. \tag{4.46}$$

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

$$\int_{\Omega} \sum_{\ell\in\Lambda_1}\sum_{k\in\Lambda_2} \Phi_\ell(x,y,z)\Phi_k(x,y,z)dxdydz = \delta_{\ell,k}, \tag{4.47}$$

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

$$\Phi_{k_x=1}(x) = \Phi_{k_x=1,k_y=0,k_z=0}(x,y,z) = \Phi_{1,0,0}(x,y,z), \tag{4.48}$$

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

$$\int_{\Omega_x} h_x^2 dx = \sum_{\substack{k\in\Lambda \\ k_x>0}} \left(c_k^{(x)}\right)^2 + \sum_{\substack{k\in\Lambda \\ k_x>0 \\ k_y=0}} \left(c_k^{(xy)}\right)^2 + \sum_{\substack{k\in\Lambda \\ k_x>0 \\ k_z=0}} \left(c_k^{(xz)}\right)^2, \tag{4.49}$$

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

$$\int_{\Omega} h_{xy}^2 \, dx \, dy = \sum_{\substack{k \in \Lambda \\ k_x > 0 \\ ky > 0}} \left(c_k^{(xy)} \right)^2, \tag{4.50}$$

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

Note than 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 polynomial chaos expansion terms for whom the only nonzero polynomials are those that the Sobol coefficient terms 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.

4.3 Adaptive HDMR

As discussed in the adaptive stochastic collocation for generalized polynomial chaos method in section 2.5, it is not only possible but likely that different input variables have different levels of impact on a response. When constructing an HDMR expansion, it is computationally wasteful to construct subsets that contain inputs that have little impact on the response. Often, however, an analyst cannot know a priori to which inputs a response is most sensitive. This is especially true when working with abstract inputs such as those provided through a Karhunen-Leove expansion [39]. As a result, as with the adaptive sparse grid for generalized polynomial chaos expansions, it would be convenient to have an algorithmic adaptive HDMR construction strategy. Such an algorithm has been proposed by Gerstner and Griebel [28] and demonstrated by Ayres and Eaton [8]. We extend their methodology here 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. 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 subset.

(b) Determine the Sobol sensitivity coefficient for each subset using cut-HDMR to ANOVA algorithms.

(c) Predict the expected impact of adding each new eligible subset to the HDMR expansion.

(d) Compare the product of a polynomial impact times its Sobol sensitivity versus the expected impact of the eligible subsets.

(e) Perform the most likely impactful method (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 remaining variance

(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 4.1. Note that this diagram assumes there is a sample submission process in a larger uncertainty quantification framework such as `RAVEN`, which handles computation resources in an efficient manner. In the flow chart, green indicates initialization, purple is the high-density model reduction portion, purple is the stochastic collocation for generalized polynomial chaos 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 finite amount of computational resources.

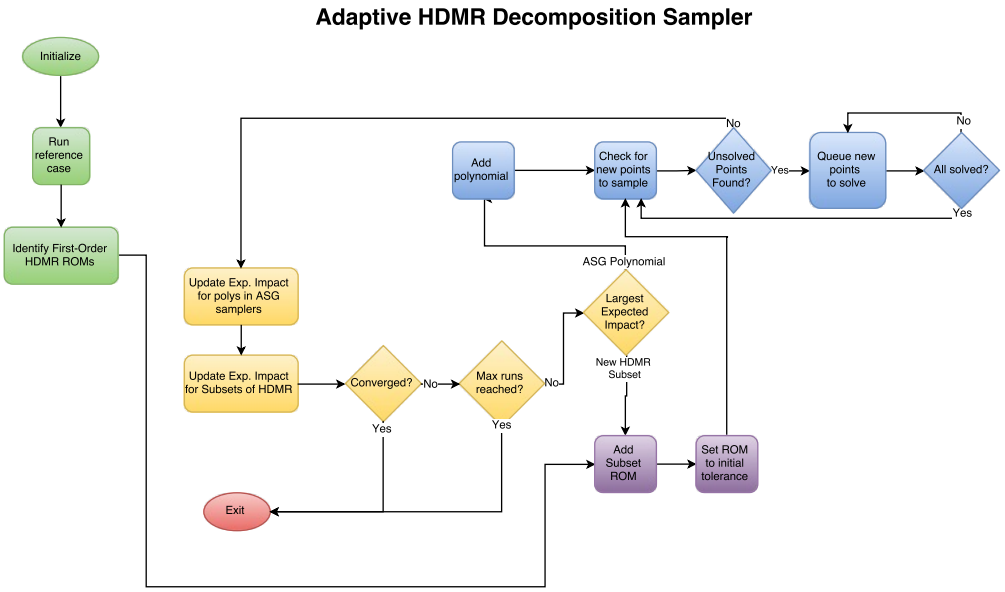

FIGURE 4.1: Adaptive HDMR with Adaptive Sparse Grid Flow Chart

Recall from section 2.5 that the impact of a polynomial within a subset generalized polynomial chaos expansion is given by Eq. 2.53,

$$\tilde{\eta}_k = \frac{1}{N-j} \sum_{n=1}^{N} \eta_{k-e_n}, \tag{4.51}$$

where we omit a superscript (y_n) to indicate the subset for which this polynomial is part of the generalized polynomial chaos expansion. As we discuss below on page 79, 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,

$$S_\beta = \frac{\text{var}[h_\beta]}{\text{var}[T(Y)]}, \tag{4.52}$$

computing h as described in section 4.2.4, 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,

$$\tilde{\xi}_k = \tilde{\eta}_k \cdot S_\beta. \tag{4.53}$$

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,

$$\xi_k = \eta_k \cdot S_\beta, \tag{4.54}$$

with η_k given in Eq. 2.52. The estimated impact $\tilde{S}_\beta$ of adding a new subset to the HDMR expansion is the average of its dependent subsets' Sobol sensitivities,

$$\tilde{S}_\beta = \frac{1}{m} \sum_{\substack{\gamma \subset \beta \\ 1 + \dim(\gamma) = m}} S_\gamma, \tag{4.55}$$

where m is the dimensionality of β,

$$m = \dim(\beta), \tag{4.56}$$

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

$$\tilde{S}_{y_1,y_2} = \frac{1}{2}(S_{y_1} + S_{y_2}). \tag{4.57}$$

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,

$$\frac{\text{local contribution}}{\text{local variance}} \cdot \frac{\text{local variance}}{\text{global variance}} = \frac{\text{local contribution}}{\text{global variance}}. \tag{4.58}$$

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

$$\tilde{\eta}_{k_1} \overset{?}{>} \tilde{\eta}_{k_2}. \tag{4.59}$$

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

$$\tilde{\eta}_{k_1} S_{\beta_1} \overset{?}{>} \tilde{\eta}_{k_2} S_{\beta_2}. \tag{4.60}$$

Comparing polynomials to subsets is somewhat more ambiguous,

$$\tilde{\eta}_{k_1} S_{\beta_1} \overset{?}{>} \tilde{S}_{\beta_3}. \tag{4.61}$$

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 before trusting the estimate of the new subset's impact.

Because the predictive algorithm is somewhat arbitrary, we additionally offer a method to provide analysts a tool to guide the adaptive algorithm. 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.

$$(\tilde{\eta}_{k_1})^{\alpha} S_{\beta_1}^{2-\alpha} \overset{?}{>} \left(\tilde{S}_{\beta_3}\right)^{2-\alpha}. \tag{4.62}$$

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 resolves, 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 polynomial chaos expansions with high-density model reduction. Previously, such as in [28], 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 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-calculate all the sensitivities and impacts, both estimated and actual, for each subset and polynomial. Because of the efficiencies of both the generalized polynomial chaos expansion and high-density model reduction expansion, this is quite computationally efficient. Whenever a new element is added to the global expansion, it changes the total variance, and so adjusts all the impact parameters.

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 the polynomial index set. However, this is a poor choice for this algorithm. Because the subset 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 such, 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.

We note that the same polynomial may appear in several different subset terms and have a different impact in each. 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 $k = (1, 0, 0)$ in a three-dimensional problem is potentially eligible for subset t_1 but not for $t_{1,2}$.

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 $u(x, y)$ consists of three subsets t_x, t_y, and $t_{x,y}$. Let the adaptive polynomial set for t_x be $\Lambda_x = ((0, 0), (1, 0))$, and for t_y be $\Lambda_y = ((0, 0), (0, 1))$, and for $t_{x,y}$ 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 $t_{x,y}$. Traditionally, this would require $k = (0, 2)$ to be present first. However, because $(0, 2)$ belongs to subset t_y, the adaptive algorithm step adds $(1, 2)$ to $t_{x,y}$ and $(0, 2)$ to t_y simultaneously. This is most likely to occur when there are strong interaction effects that overshadow single-input effects.

There are several further improvements that can be made to this combined adaptive algorithm, which we discuss in section **??**.

Chapter 5

Results: High-Density Model Reduction

TODO change the analysis to only consider HDMR, and add more static Sobol analysis!

5.1 Introduction

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

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

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

$$\epsilon_{95} = \frac{1.96\bar{\sigma}_N}{\sqrt{N}}, \tag{5.1}$$

$$\bar{\sigma}_N^2 = \frac{N}{N-1}\sigma_N^2 = \frac{N}{N-1}\left(\frac{1}{N}\sum_{i=1}^{M} u(Y_i)^2 - \bar{u}(Y)_N^2\right), \tag{5.2}$$

where Y_i are a set of M independent identically-distributed realizations taken from the input space. These errorbars estimate where the value of the statistic is with a

probability near 0.95. The estimate of this error improves as additional samples are taken.

Convergence figures are log-log error graphs with the number of computational solves on the x-axis and error with respect to the analytical solution on the y-axis. The distinct lines demonstrate series of results obtained for each UQ method. The series we show here are analog traditional Monte Carlo (mc); static stochastic collocation for generalized polynomial chaos expansion using the hyperbolic cross index set (hc), total degree index set (td), and (where possible) tensor product index set (tp); adaptive stochastic collocation for polynomial chaos method (adaptSC); and adaptive Sobol decomposition with polynomial chaos subsets (adaptSobol). Each series obtains additional values by increasing the refinement of the method. For Monte Carlo, 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 and/or dimension subsets.

The measure of success for a method is less dependent on the absolute value of the error shown. While this is useful, 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 on two factors: the dimensionality of the uncertain space for the model, and the continuity 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.

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

5.2 Tensor Monomials

This model is described in section **??**. As this polynomial contains only combinations of first-order polynomials, we expect the Tensor Product index set construction method (TP) to be very efficient in absolute error magnitude. As such, it will be difficult to see the convergence rate for the tensor product method. Because the model has infinite continuity, we expect all collocation-based methods to be quite efficient. The values and errors of the mean and standard deviation are given in Figures 5.5 through 5.8

for 5 uncertain inputs, and the same for 10 dimensions is given in Figures 5.9 through 5.12. Note that TP exactly reproduces the original model with expansion order 1, so no convergence is observed past the initial sampling point.

5.2.1 3 Inputs

The strength of collocation methods is clear for this small-dimensionality problem of three uncertain inputs. The convergence on the mean and standard deviation is swift for all the methods. 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. Both adaptive methods converge at nearly identical rates; this is not surprising, as for small dimension problems the adaptive search follows a very similar path. The adaptive SCgPC method reaches convergence somewhat slower because it naturally tends to explore higher-order polynomials before extending to low-order polynomials with more cross terms.

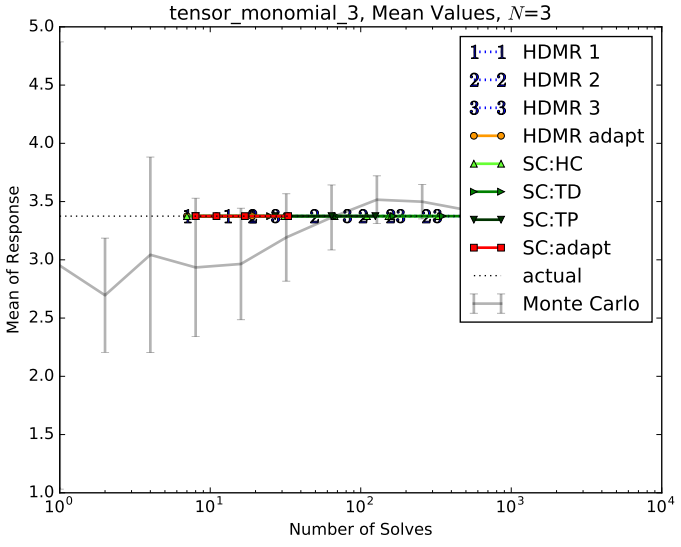

FIGURE 5.1: Tensor Monomial, $N = 3$, Mean Values

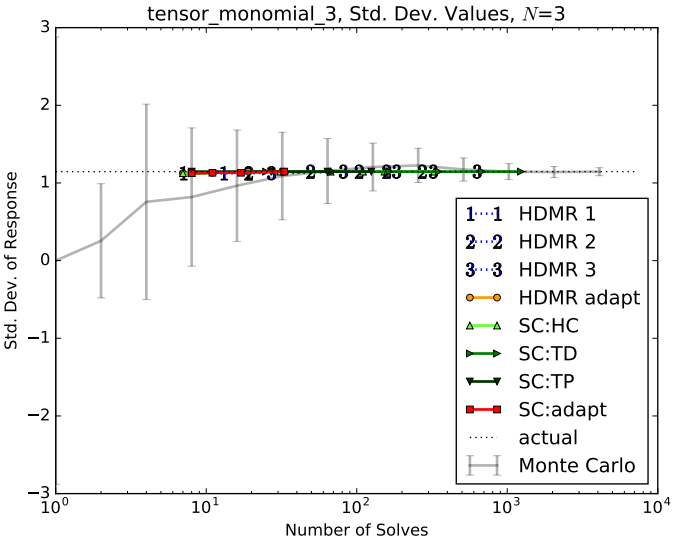

FIGURE 5.2: Tensor Monomial, $N = 3$, Std. Dev. Values

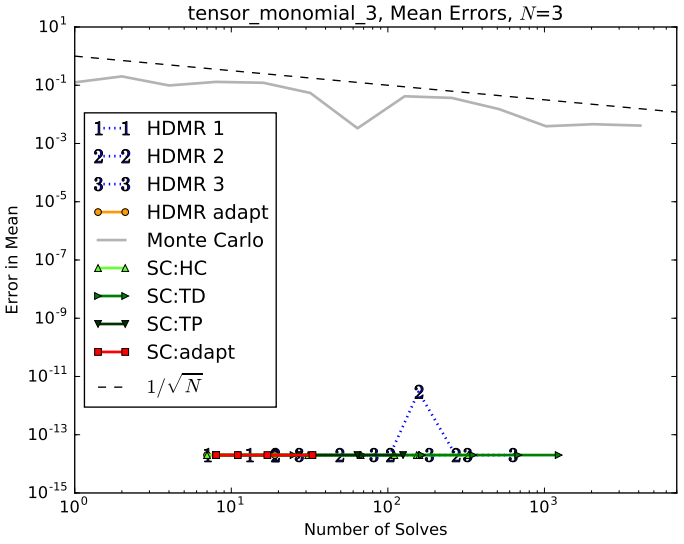

FIGURE 5.3: Tensor Monomial, $N = 3$, Mean Convergence

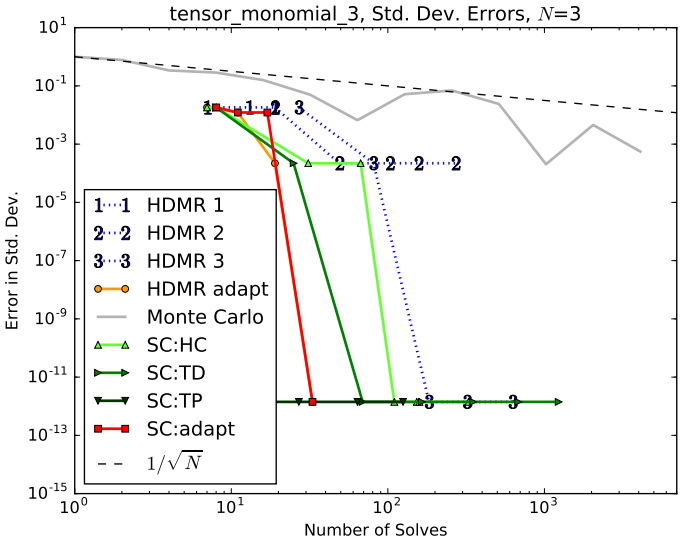

FIGURE 5.4: Tensor Monomial, $N = 3$, Std. Dev. Convergence

5.2.2 5 Inputs

While the convergence on the mean is still direct for the five-dimensional input problem, we begin to see degradation in the convergence of collocation-based methods. As with the three variable case, the mean is trivial and obtained with the zeroth-order polynomial. Exponential convergence can be seen for the adaptive Sobol, total degree, and hyperbolic cross methods, while the adaptive SCgPC 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. Total Degree outperforms adaptive methods, as the search algorithms struggle to find the optimal tensors of low-order polynomials required. Hyperbolic Cross is outperformed by Total Degree, as expected for a problem with this level of regularity.

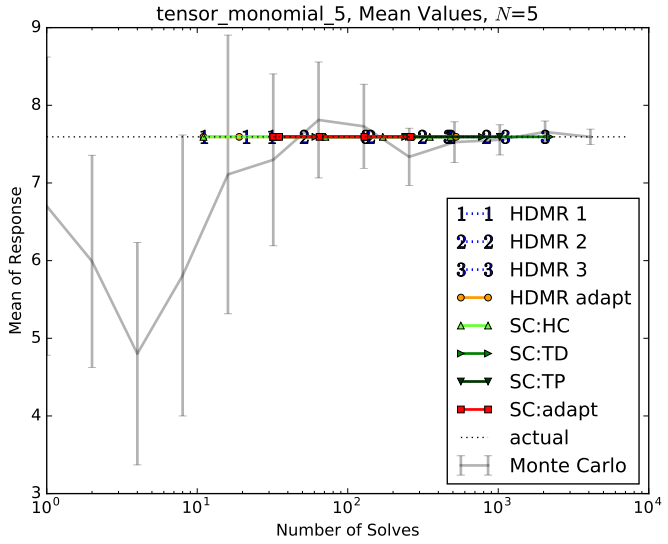

FIGURE 5.5: Tensor Monomial, $N = 5$, Mean Values

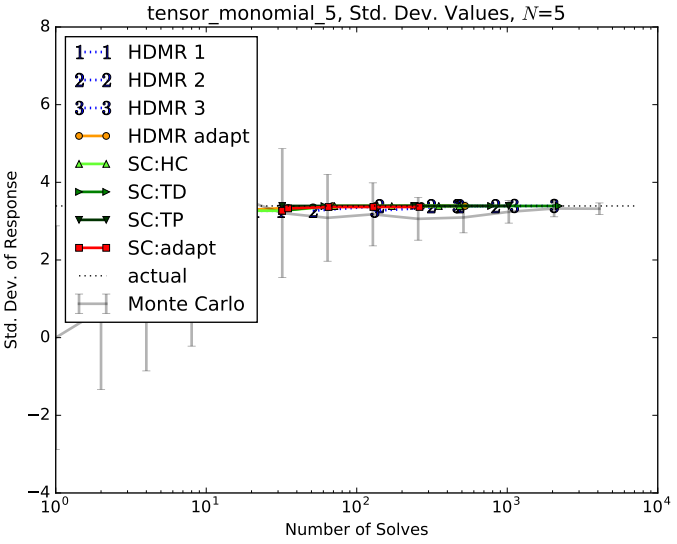

FIGURE 5.6: Tensor Monomial, $N = 5$, Std. Dev. Values

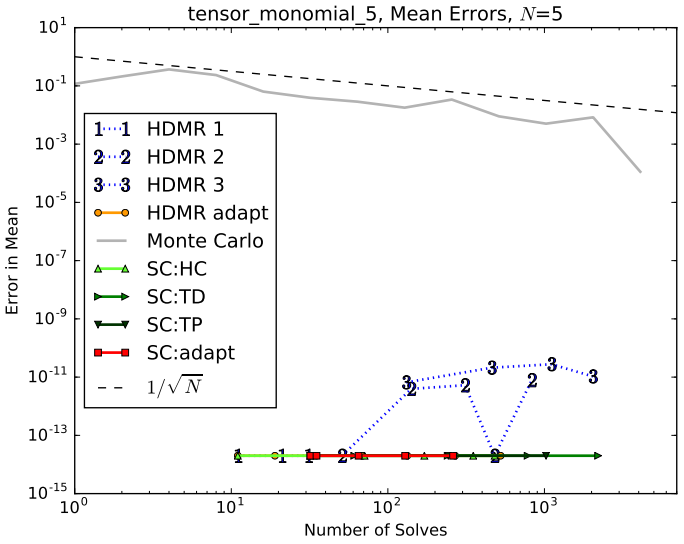

FIGURE 5.7: Tensor Monomial, $N = 5$, Mean Convergence

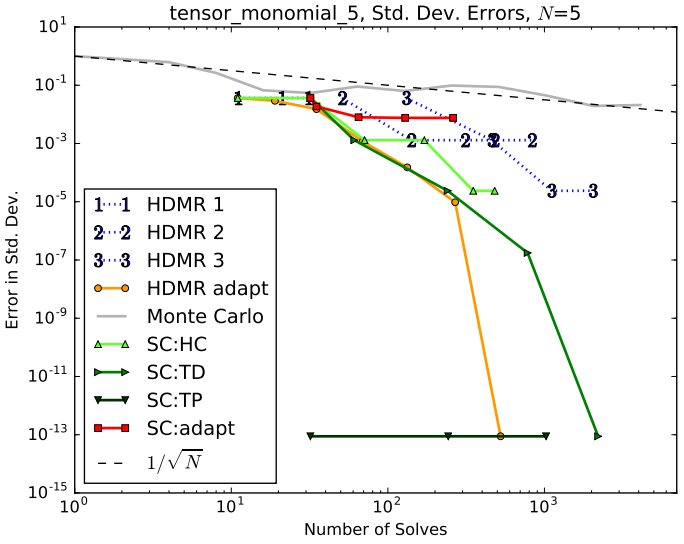

FIGURE 5.8: Tensor Monomial, $N = 5$, Std. Dev. Convergence

5.2.3 10 Inputs

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

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

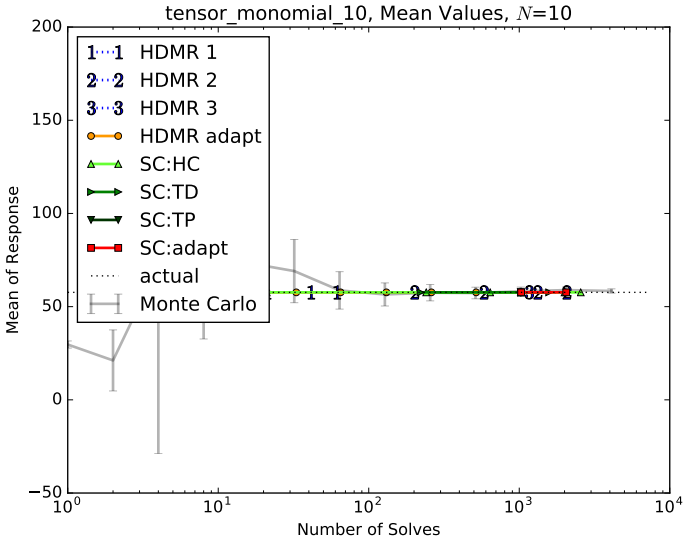

FIGURE 5.9: Tensor Monomial, $N = 10$, Mean Values

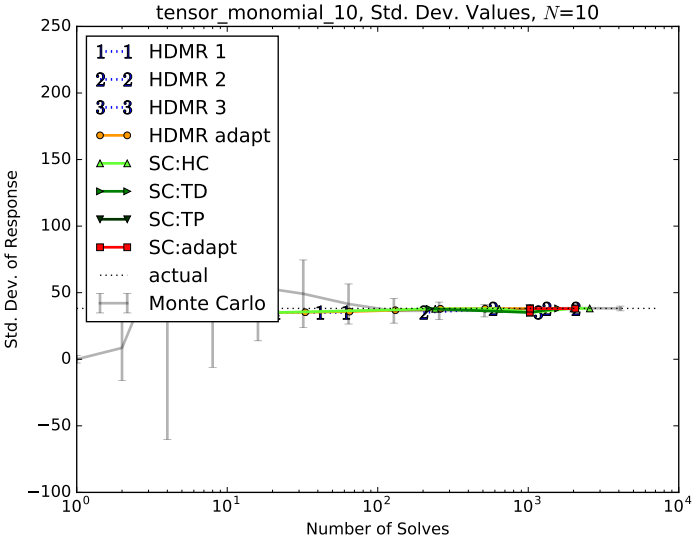

FIGURE 5.10: Tensor Monomial, $N = 10$, Std. Dev. Values

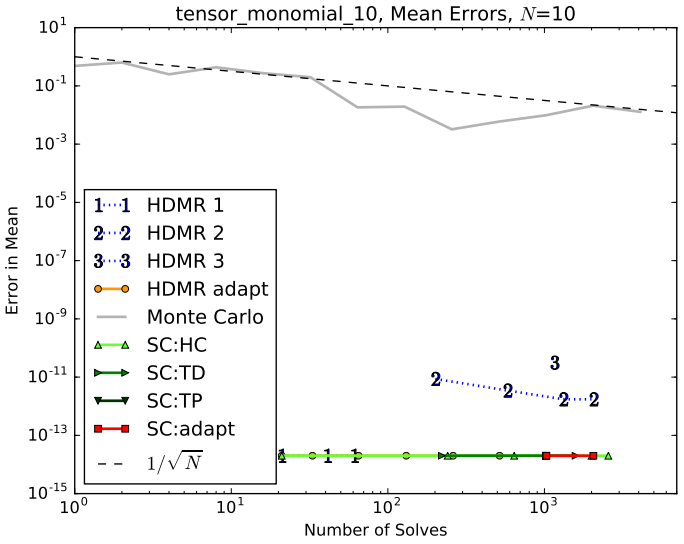

FIGURE 5.11: Tensor Monomial, $N = 10$, Mean Convergence

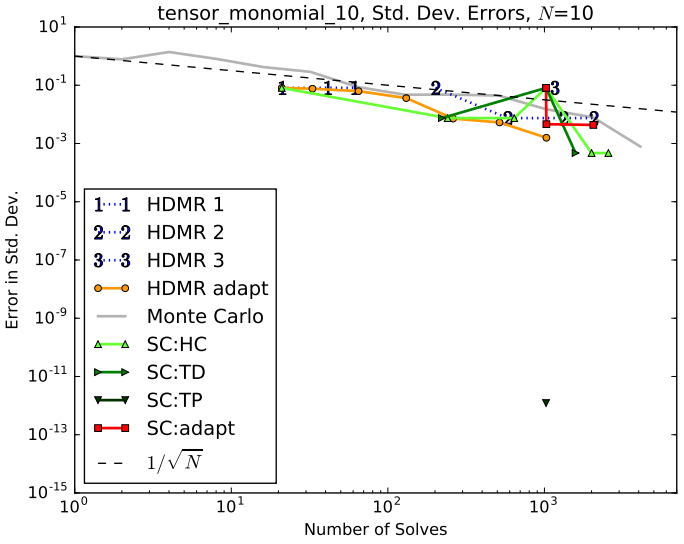

FIGURE 5.12: Tensor Monomial, $N = 10$, Std. Dev. Convergence

5.3 Sudret Polynomial

This model is described in section **??**. The Sudret polynomial model is a near neighbor to the tensor monomials model; however, it includes only even-ordered polynomials, which provides more of a challenge to the adaptive methods. Because the model is still a tensor product, the tensor product collocation method converges most directly in all dimensionality cases.

5.3.1 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 sobol 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, with the exception of the adaptive Sobol algorithm, which is not as effective as the adaptive SCgPC at finding the second-order polynomials for inclusion in the expansion. The tensor product still converges very rapidly, and total degree shows a good rate of convergence.

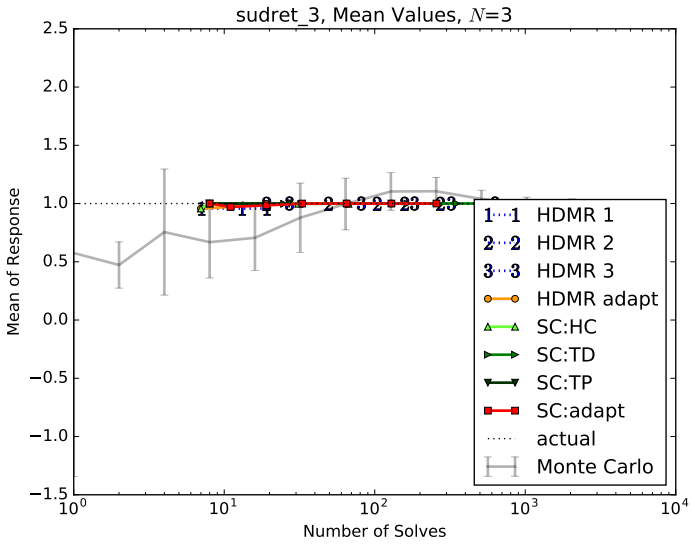

FIGURE 5.13: Sudret Polynomial, $N = 3$, Mean Values

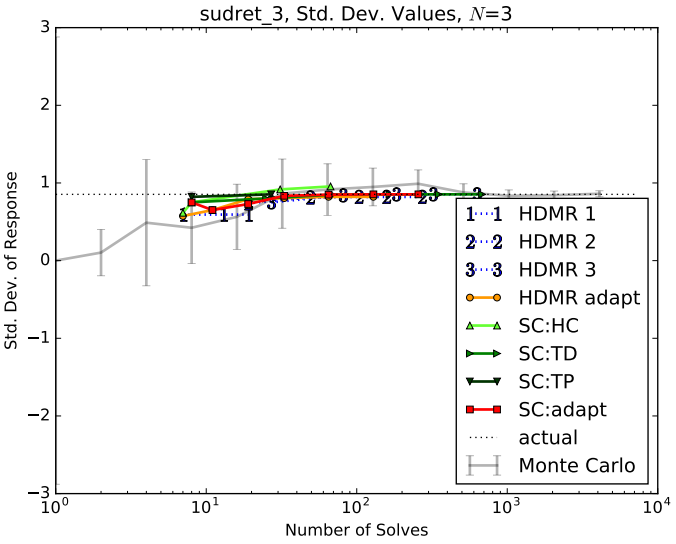

FIGURE 5.14: Sudret Polynomial, $N = 3$, Std. Dev. Values

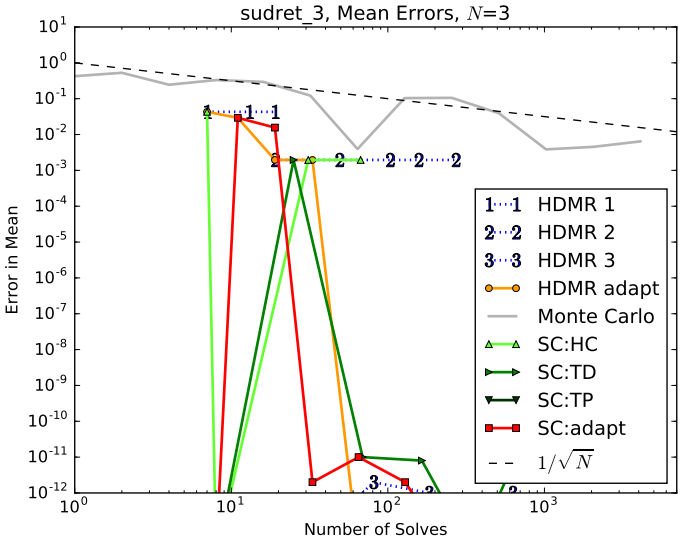

FIGURE 5.15: Sudret Polynomial, $N = 3$, Mean Convergence

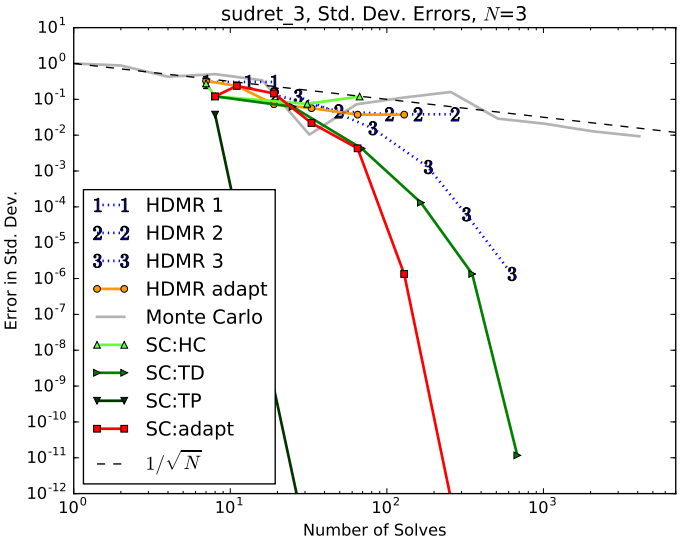

FIGURE 5.16: Sudret Polynomial, $N = 3$, Std. Dev. Convergence

5.3.2 5 Inputs

In the five-input case for the Sudret polynomials, we see slower convergence for all methods, as expected for collocation-based methods. For the mean, the adaptive sobol shows the most rapid rate of convergence, but generally each method is showing some level of exponential convergence. For the standard deviation, however, the radius of curvature for the convergence is quite large, and the adaptive Sobol method seems to be searching fairly ineffectively for the most crucial polynomials in the expansion. This is likely because all the first-order effects are missing in the expansion, leading the algorithm to be blinded as to the most effective route forward. We discuss this limitation and approches to correcting it in Chapter **??**.

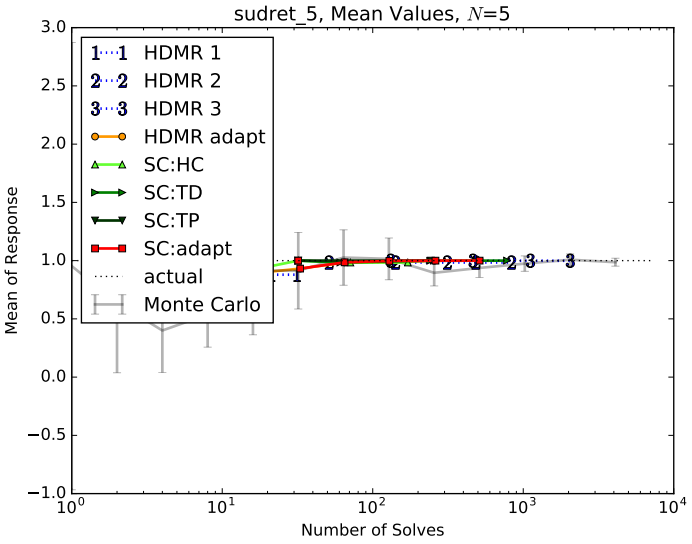

FIGURE 5.17: Sudret Polynomial, $N = 5$, Mean Values

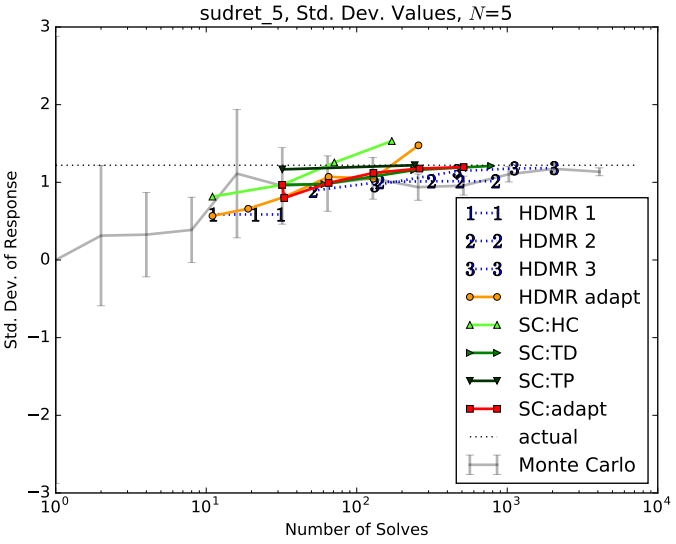

FIGURE 5.18: Sudret Polynomial, $N = 5$, Std. Dev. Values

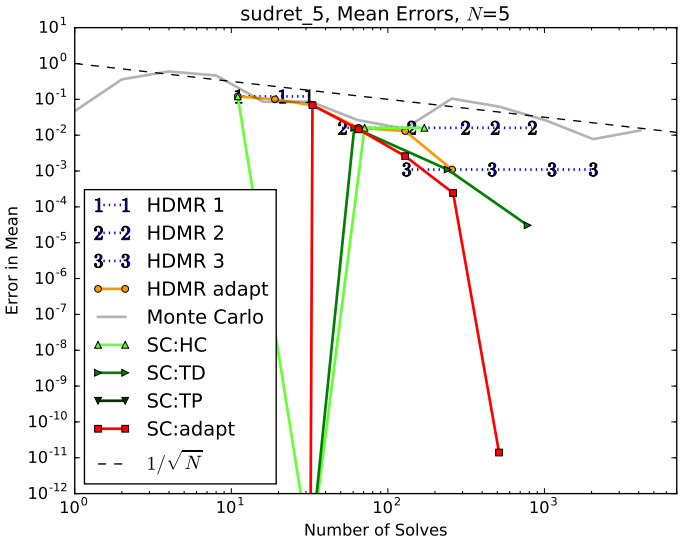

FIGURE 5.19: Sudret Polynomial, $N = 5$, Mean Convergence

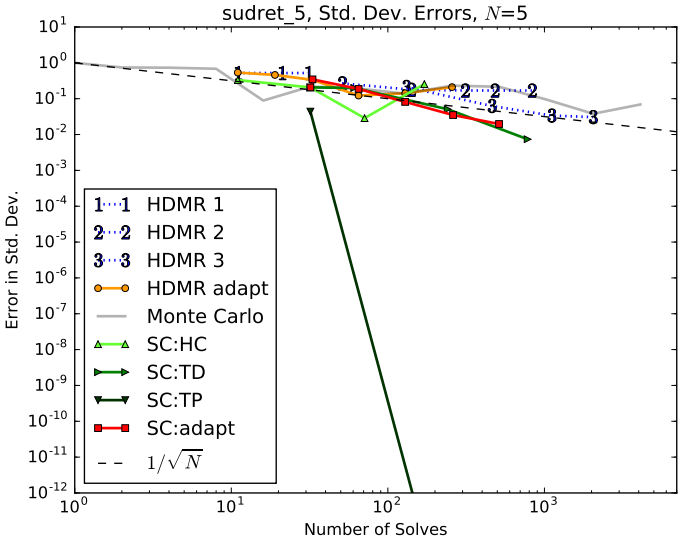

FIGURE 5.20: Sudret Polynomial, $N = 5$, Std. Dev. Convergence

5.4 Attenuation

This model is described in section **??**. Similar to the tensor monomial model and Sudret polynomial model, the attenuation model can be expanded as the infinite sum of ever-increasing polynomials, making it tensor product in shape. However, unlike the previous two models, the magnitude of the contribution from the higher-order polynomials decreases swiftly, making lower-order polynomials more characteristic of the model

in general. Because this model is still a tensor model and the response is infinitely continuous, we expect to see a similar trend in the most effective methods for polynomial expansion.

5.4.1 2 Inputs

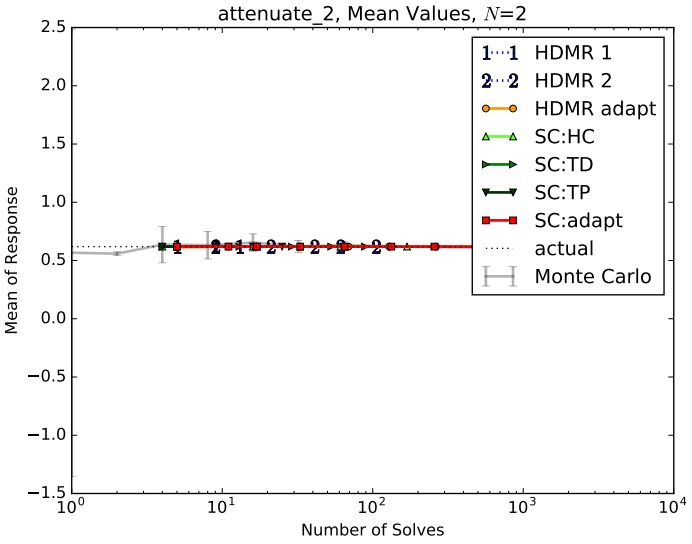

FIGURE 5.21: Attenuation, $N = 2$, Mean Values

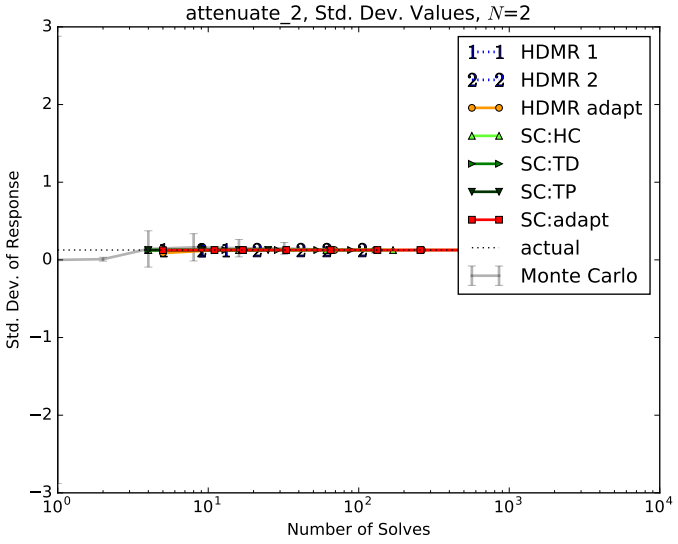

FIGURE 5.22: Attenuation, $N = 2$, Std. Dev. Values

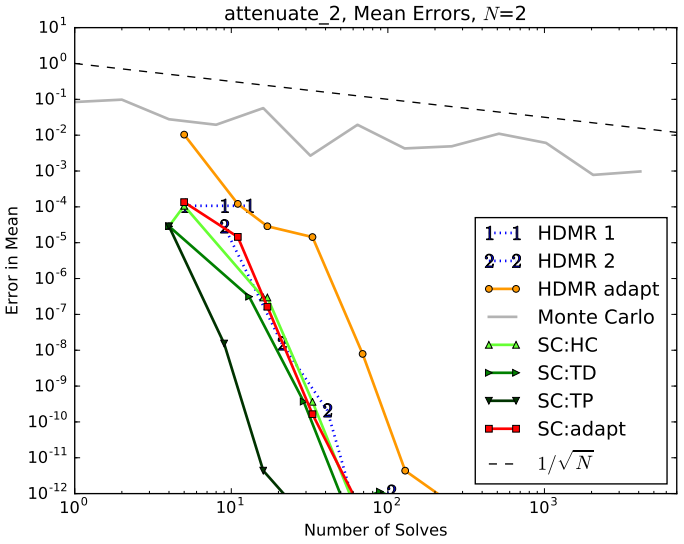

FIGURE 5.23: Attenuation, $N = 2$, Mean Convergence

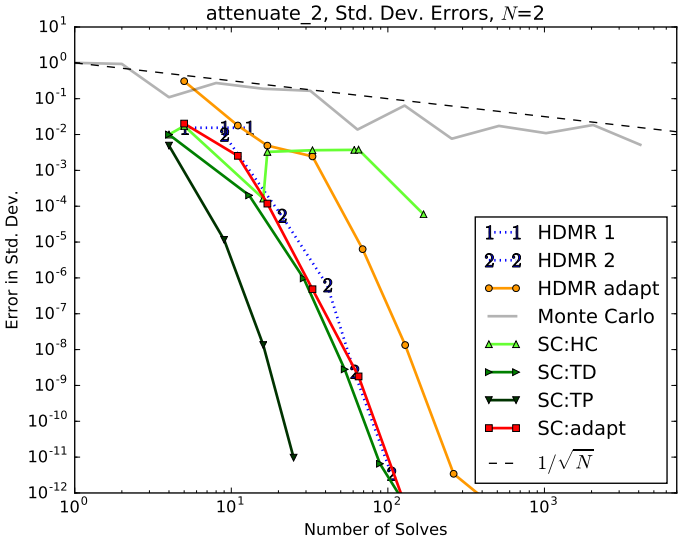

FIGURE 5.24: Attenuation, $N = 2$, Std. Dev. Convergence

5.4.2 4 Inputs

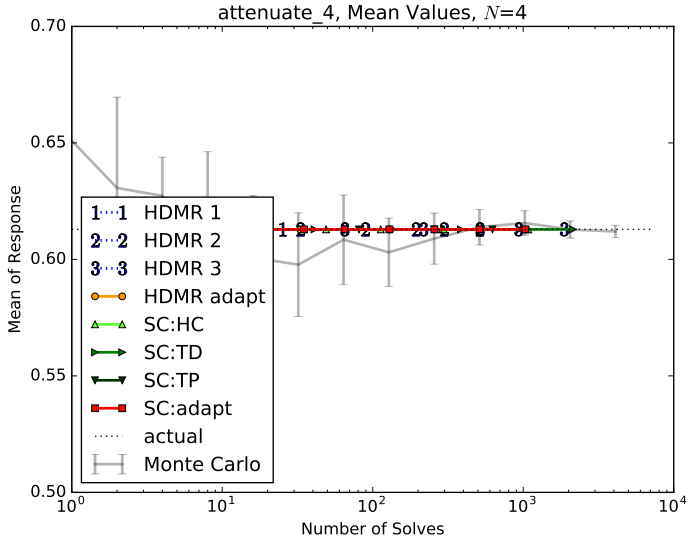

FIGURE 5.25: Attenuation, $N = 4$, Mean Values

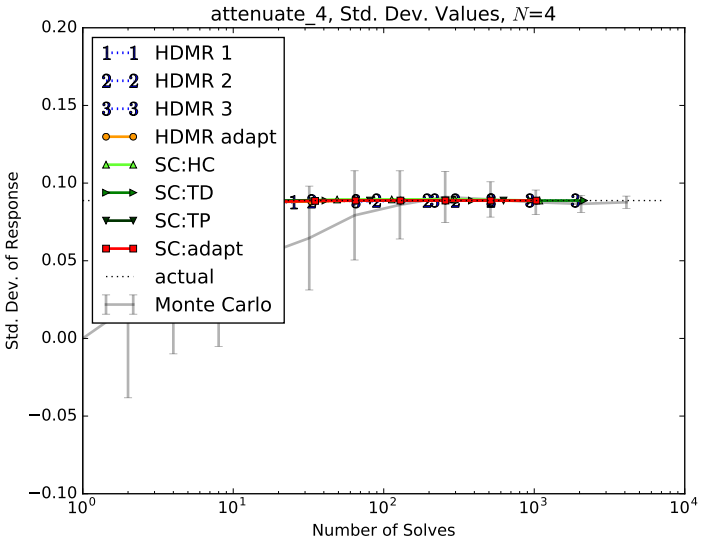

FIGURE 5.26: Attenuation, $N = 4$, Std. Dev. Values

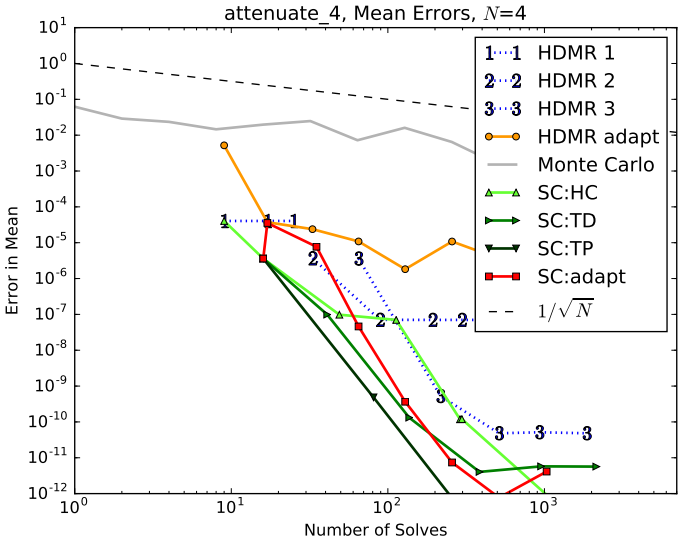

FIGURE 5.27: Attenuation, $N = 4$, Mean Convergence

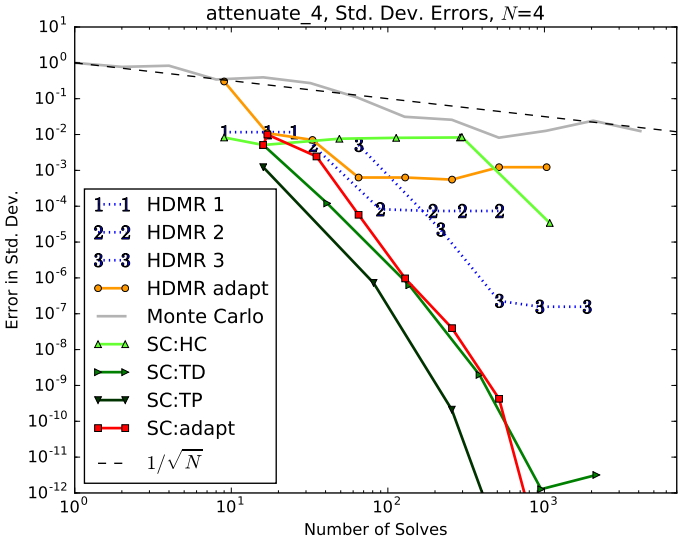

FIGURE 5.28: Attenuation, $N = 4$, Std. Dev. Convergence

5.4.3 6 Inputs

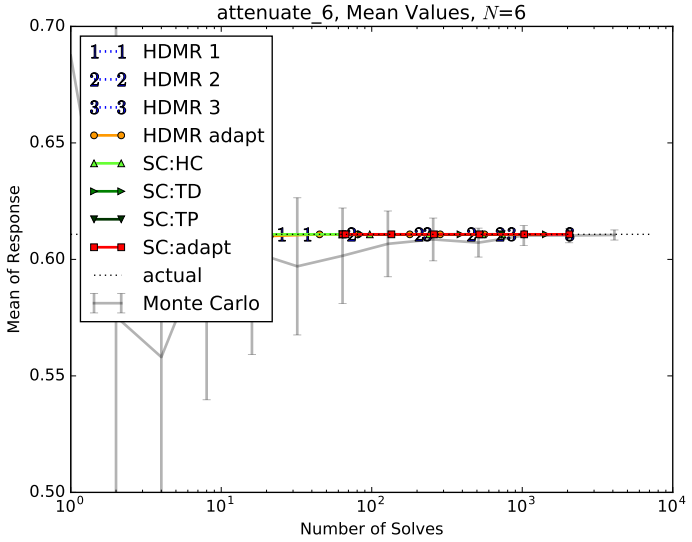

FIGURE 5.29: Attenuation, $N = 6$, Mean Values

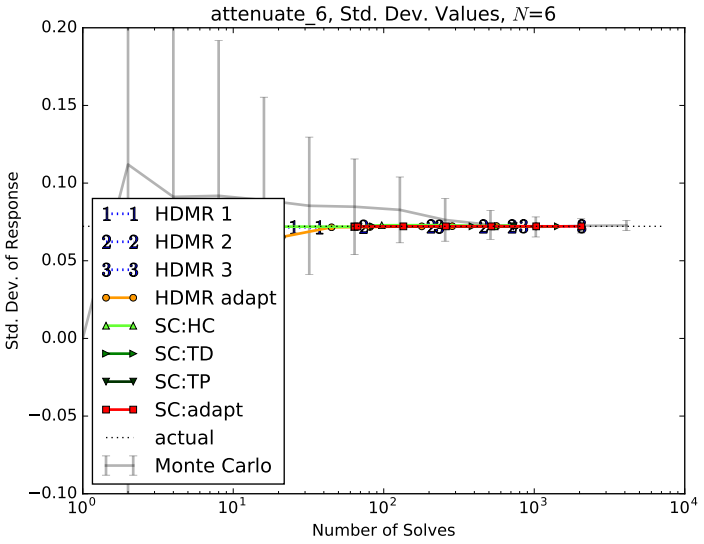

FIGURE 5.30: Attenuation, $N = 6$, Std. Dev. Values

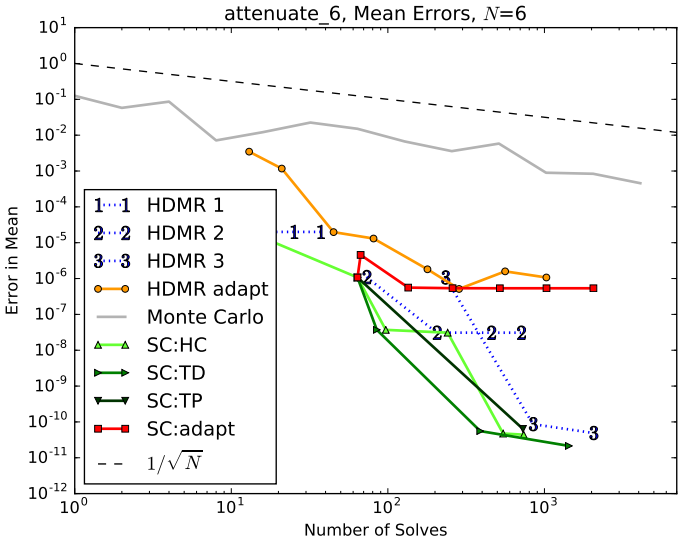

FIGURE 5.31: Attenuation, $N = 6$, Mean Convergence

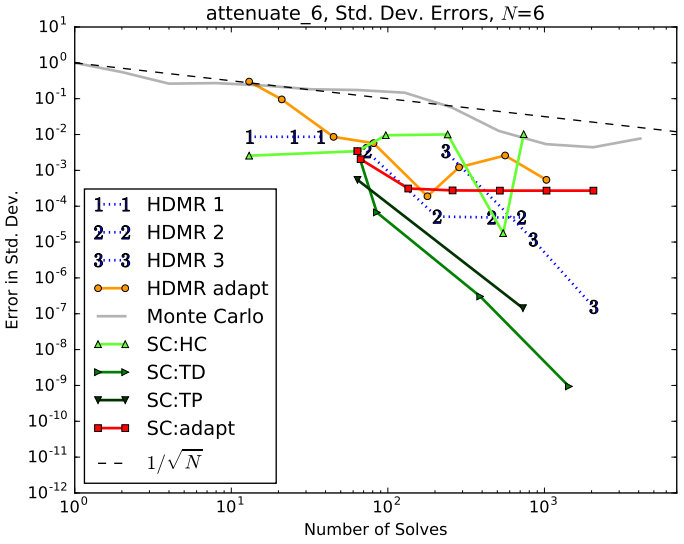

FIGURE 5.32: Attenuation, $N = 6$, Std. Dev. Convergence

5.5 Gauss Peak

This model is described in section **??**.

5.5.1 3 Inputs

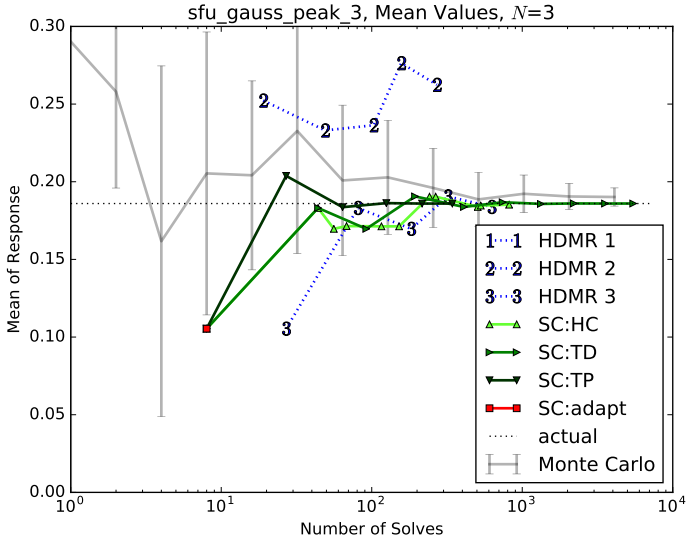

FIGURE 5.33: Gauss Peak, $N = 3$, Mean Values

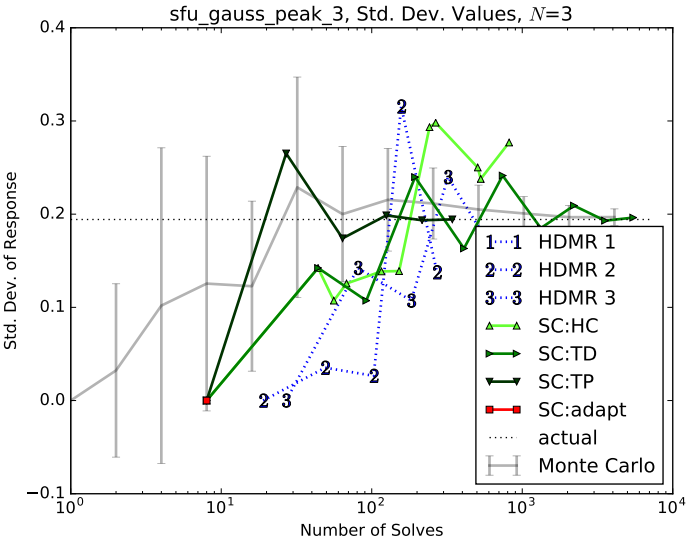

FIGURE 5.34: Gauss Peak, $N = 3$, Std. Dev. Values

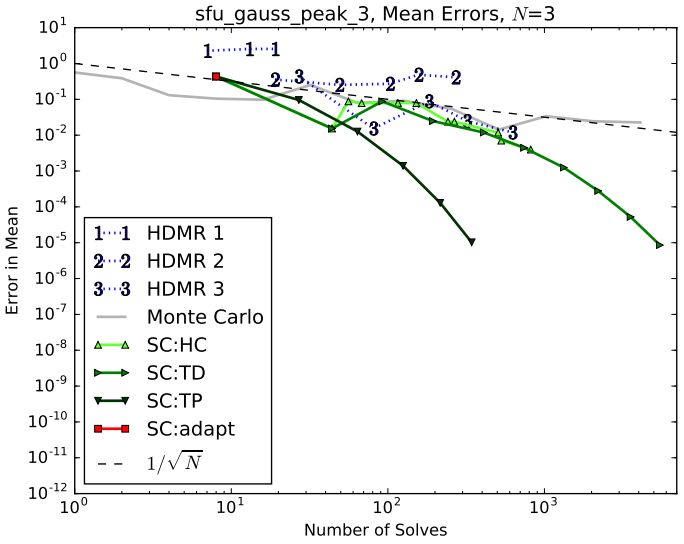

FIGURE 5.35: Gauss Peak, $N = 3$, Mean Convergence

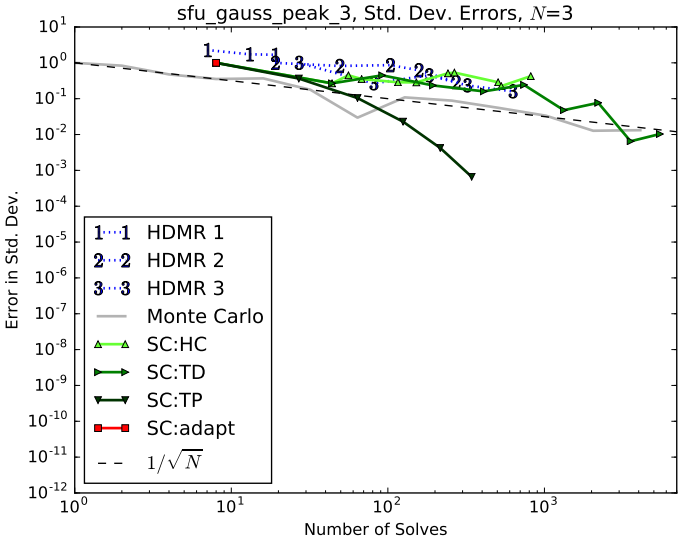

FIGURE 5.36: Gauss Peak, $N = 3$, Std. Dev. Convergence

5.5.2 5 Inputs

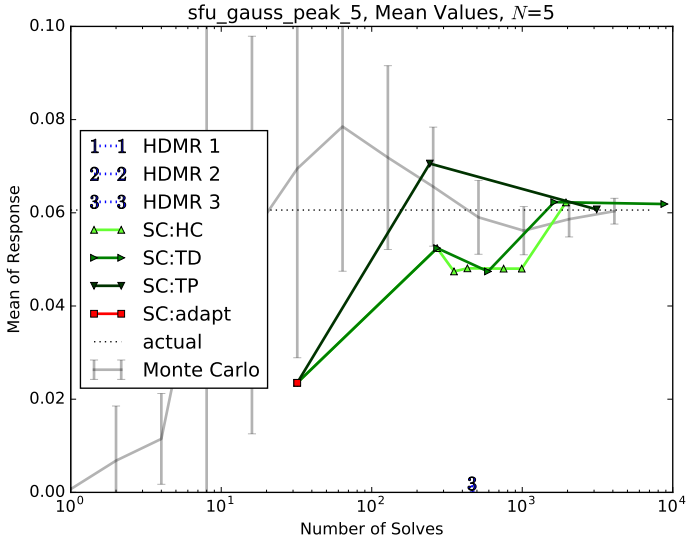

FIGURE 5.37: Gauss Peak, $N = 5$, Mean Values

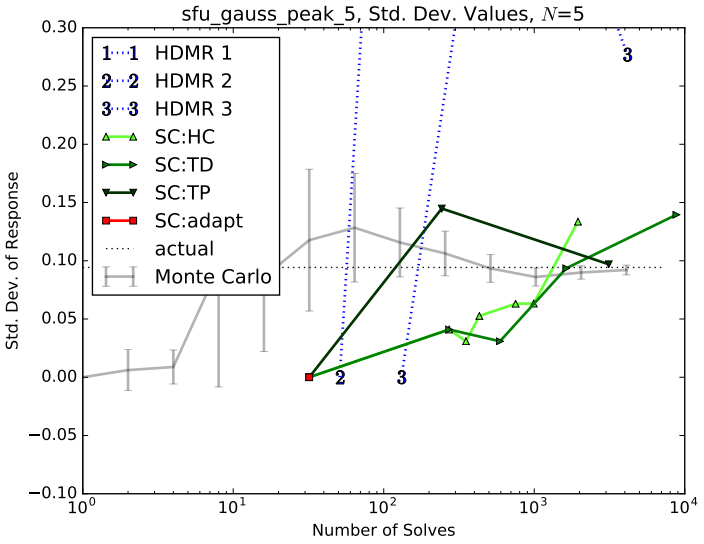

FIGURE 5.38: Gauss Peak, $N = 5$, Std. Dev. Values

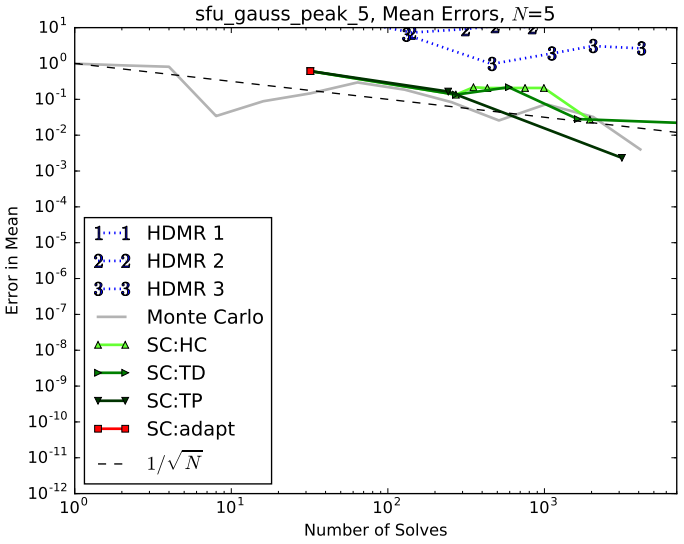

FIGURE 5.39: Gauss Peak, $N = 5$, Mean Convergence

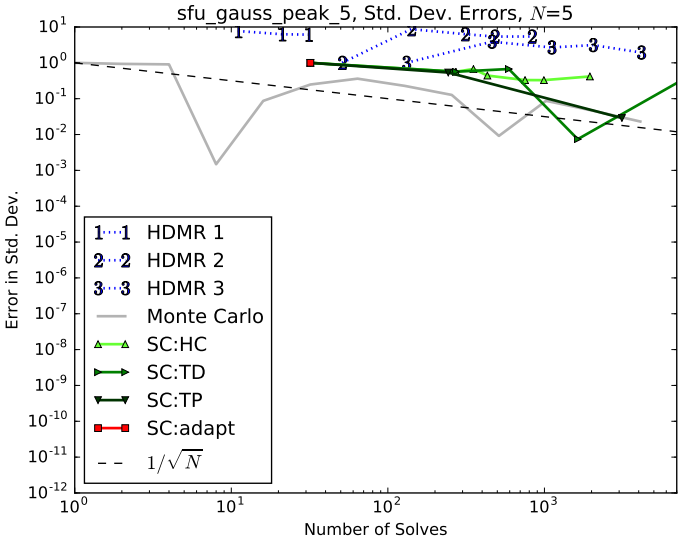

FIGURE 5.40: Gauss Peak, $N = 5$, Std. Dev. Convergence

5.6 Ishigami

This model is described in section **??**.

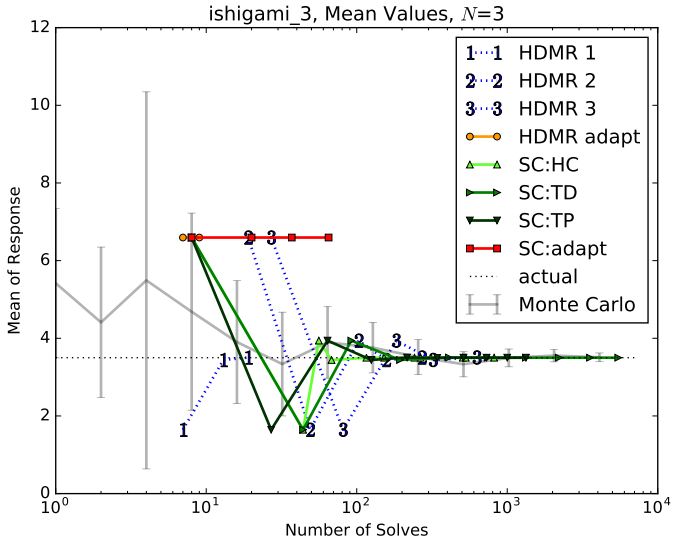

FIGURE 5.41: Ishigami, $N = 3$, Mean Values

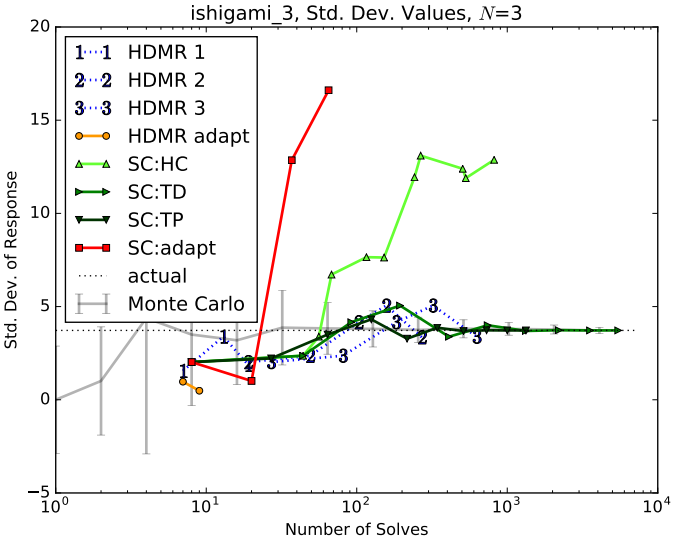

FIGURE 5.42: Ishigami, $N = 3$, Std. Dev. Values

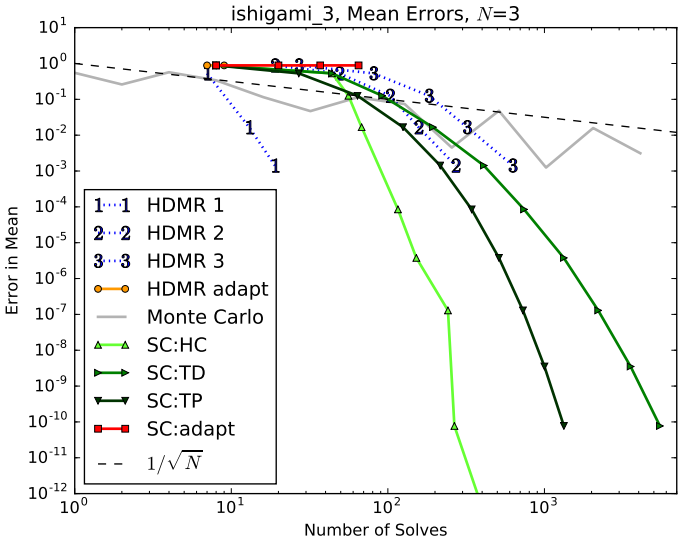

FIGURE 5.43: Ishigami, $N = 3$, Mean Convergence

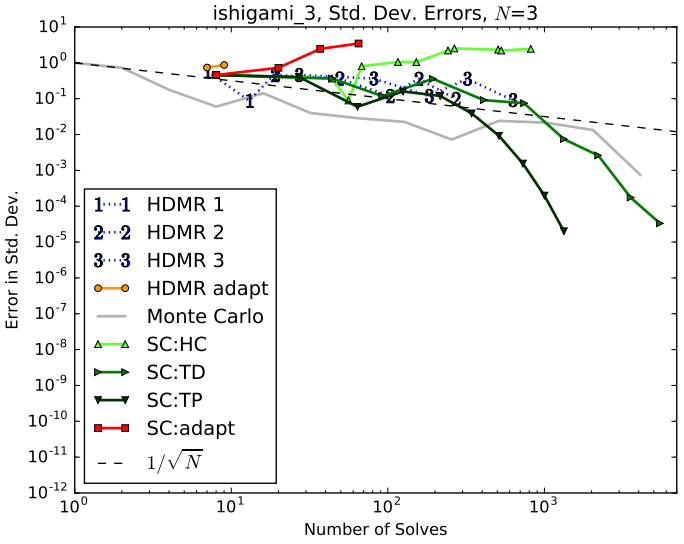

FIGURE 5.44: Ishigami, $N = 3$, Std. Dev. Convergence

5.7 Sobol G-Function

This model is described in section **??**.

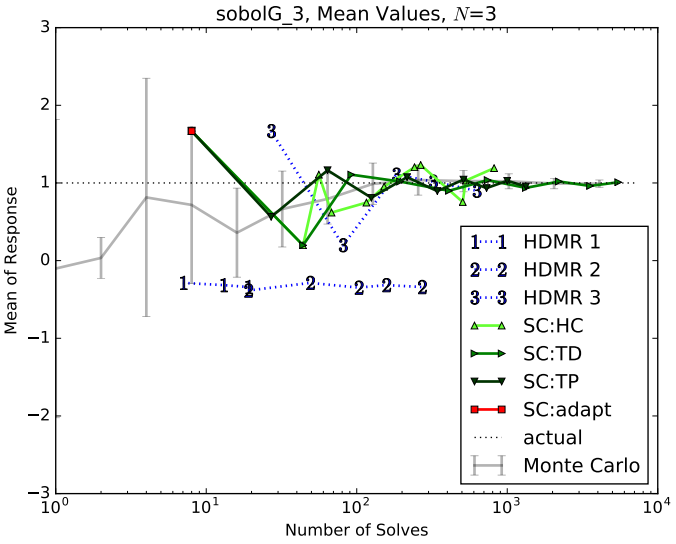

FIGURE 5.45: Sobol G-Function, $N = 3$, Mean Values

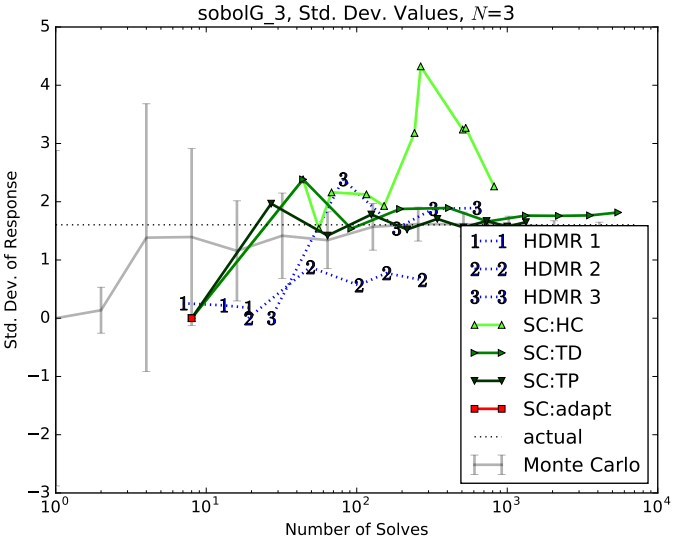

FIGURE 5.46: Sobol G-Function, $N = 3$, Std. Dev. Values

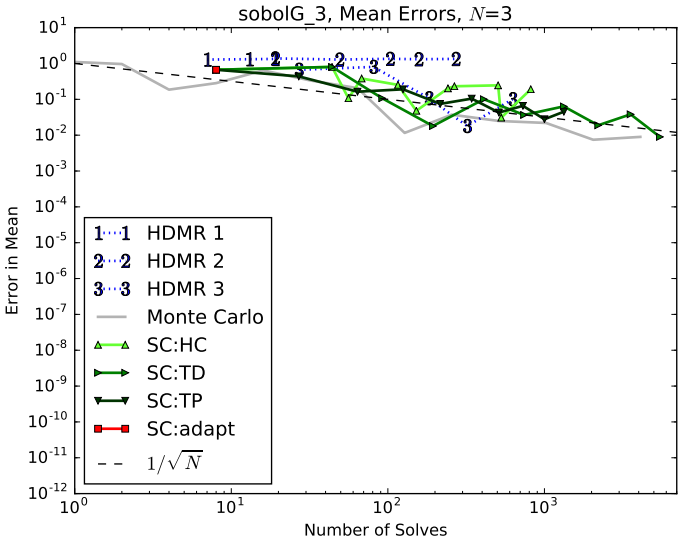

FIGURE 5.47: Sobol G-Function, $N = 3$, Mean Convergence

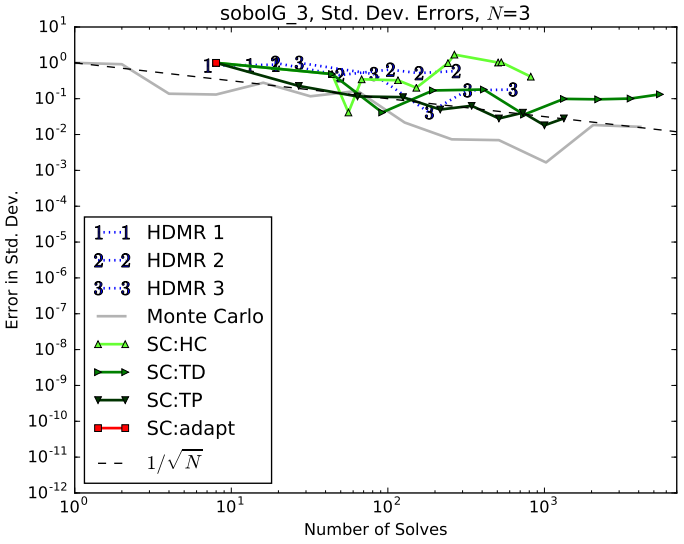

FIGURE 5.48: Sobol G-Function, $N = 3$, Std. Dev. Convergence

5.8 Conclusions

todo

Chapter 6

Demonstration: Neutron Transport

6.1 Introduction

Todo

Chapter 7

Engineering Demonstration

7.1 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.

7.2 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 [53].

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, and clad conductivity. The macroscopic cross sections are calculated using `Scale` [54] along with a covariance matrix which provides the correlation between cross sections. Karhunen-Loevre [39] is used to decouple the resulting 671 inputs and reduce them to 20

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

TABLE 7.1: Application Versions Used

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 [52] commit hashes. This computation was performed with the application versions listed in Table 7.1. There is nothing particularly special about these versions, except that they were concurrent and compatible at the time calculations were begun.

7.3 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.

7.4 Results

Figures 7.1 and 7.3 show the values obtained for the mean and variance of this model for a selection of uncertainty quantification methods, including traditional Monte Carlo (mc),

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

TABLE 7.2: Evaluations Required for 23 Input Pin Cell Model

static stochastic collocation for generalized polynomial chaos expansion using the total degree polynomial construction indices (`td`), first- and second-order static Sobol decomposition (or HDMR) (`sobol`), and adaptive Sobol decomposition using both adaptive cut-HDMR and adaptive generalized polynomial chaos (`adaptSobol`). For additional clarity, we provide graphs centered more especially on the non-Monte Carlo data in Figures 7.2 and 7.4. 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 7.2 for first-order static Sobol 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 Sobol2 matches TD, and how that suggests linearity in the second-order interactions.

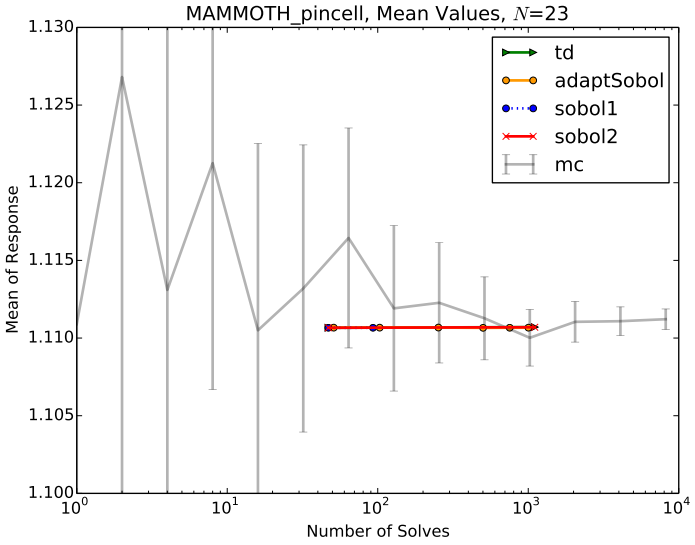

FIGURE 7.1: MAMMOTH Pin Cell, Mean Values

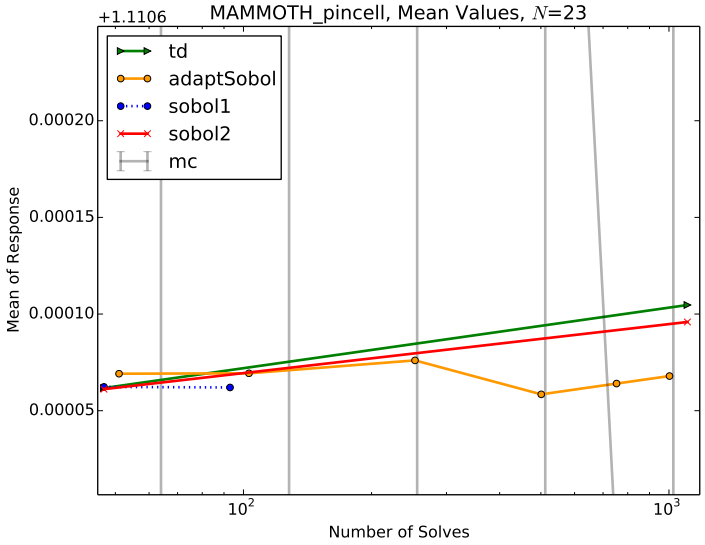

FIGURE 7.2: MAMMOTH Pin Cell, Mean Values (Zoomed)

7.5 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

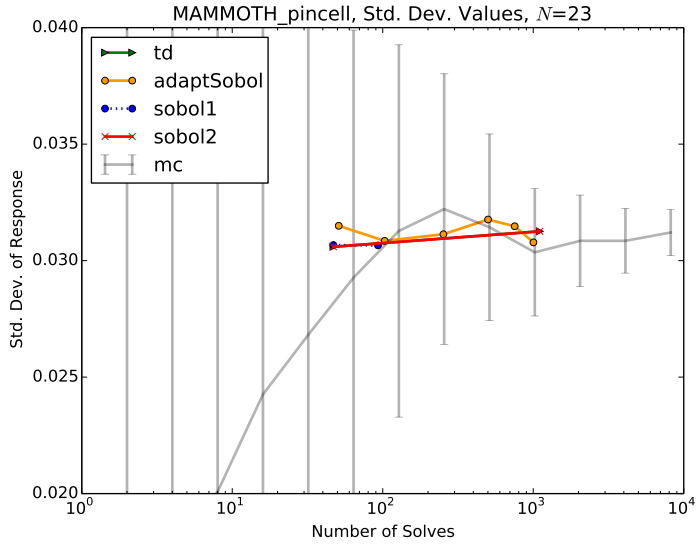

FIGURE 7.3: MAMMOTH Pin Cell, Variance Values

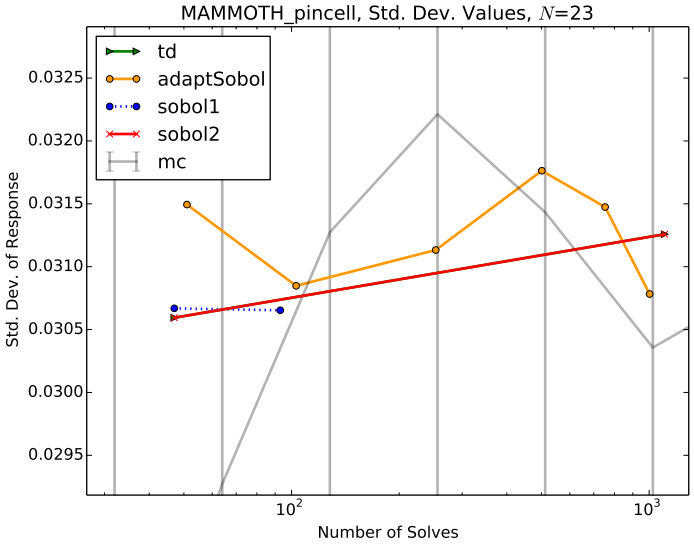

FIGURE 7.4: MAMMOTH Pin Cell, Variance Values (Zoomed)

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 8

Time-Dependent Analysis

8.1 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` [51] 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` [50] simulation of an OECD benchmak [48] 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 [49], and we use the same input variables and uncertainty distributions here.

8.2 Problem Description

The problem considered here is Case 2a defined in Chapter 2 of the benchmark report [48]. 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 [49]. 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 [49]. 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 [49]. 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 8.2. 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.

8.3 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 8.1: 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 8.2: 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 8.3 through 8.6 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 8.1 and 8.2 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.

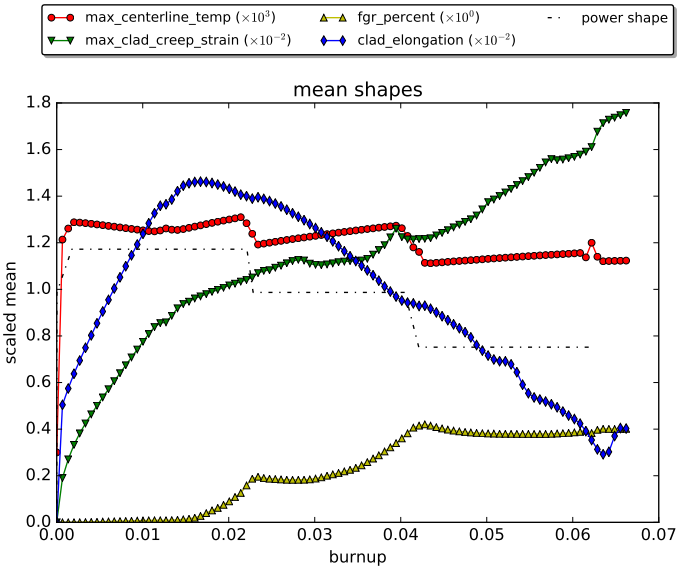

FIGURE 8.1: OECD Response Mean Values over Burnup

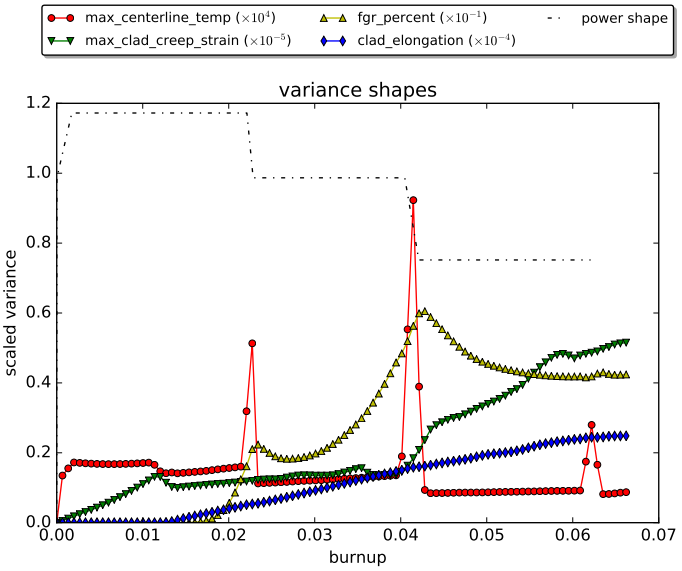

FIGURE 8.2: OECD Response Variance Values over Burnup

8.3.1 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.

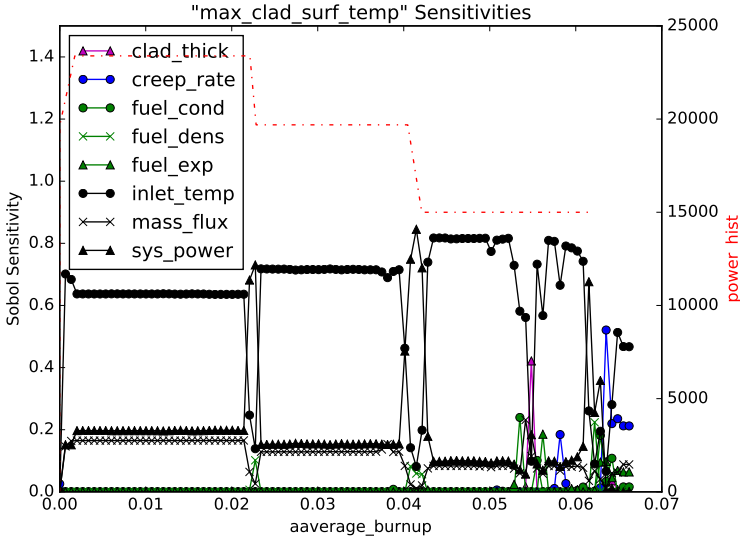

FIGURE 8.3: Maximum Clad Surface Temperature Sensitivities

As can be seen in Figure 8.3, 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.

8.3.2 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.

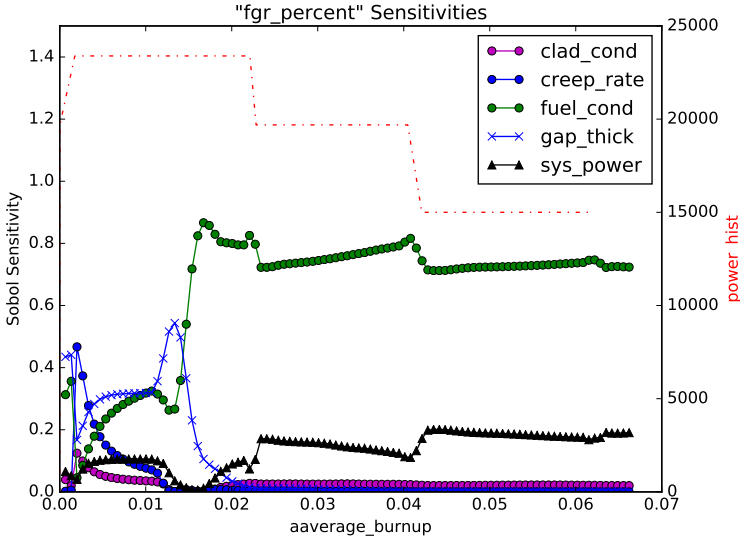

FIGURE 8.4: Percent Fission Gas Released Sensitivities

As can be seen in Figure 8.4, 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.

8.3.3 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.

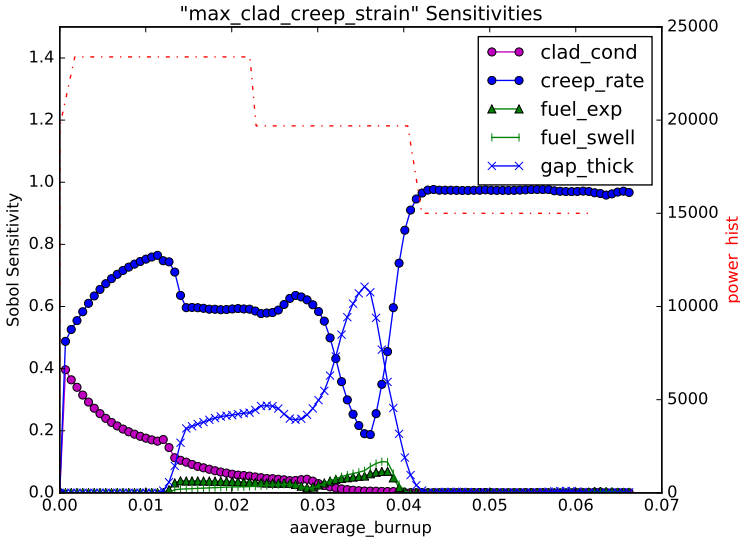

FIGURE 8.5: Maximum Cladding Creep Strain Sensitivities

As can be seen in Figure 8.5, 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.

8.3.4 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 8.1, the clad tends to elongate

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

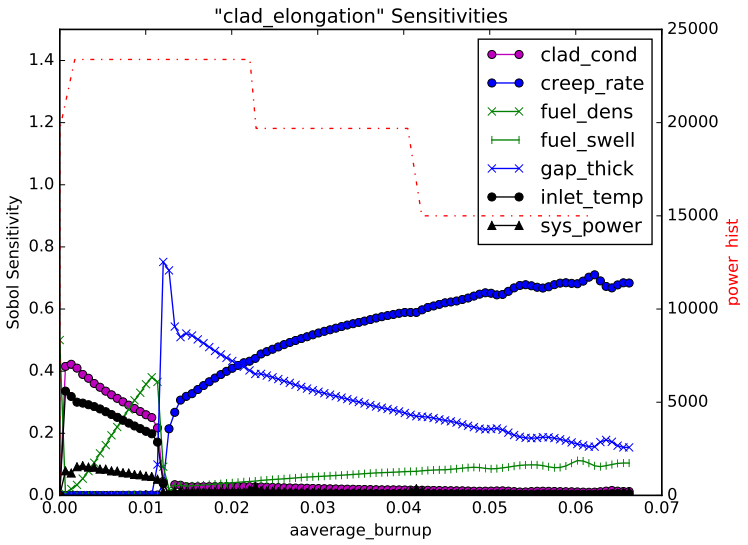

FIGURE 8.6: Clad Elongation Sensitivities

As can be seen in Figure 8.6, 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.

8.4 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 9

Conclusions

9.1 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.

9.2 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 Sobol 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.

9.3 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.

9.4 Final Comments

Needed?

Bibliography

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

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

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

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

[5] 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.

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

[7] 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.

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

[9] 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.

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

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

[12] 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.

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

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

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

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

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

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

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

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

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

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

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

[34] 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.

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

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

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

[38] 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.

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

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

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

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

[43] 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.

[44] 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.

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

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

[53] 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.

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