University of New Mexico

Doctoral Thesis

Advanced Methods in Stochastic Collocation for Polynomial Chaos in RAVEN

Author:
Paul W. Talbot

Supervisor:
Dr. Anil K. Prinja

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

in the

Department of Nuclear Engineering

December 2015

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 the impact of input uncertainties in such systems has grown with the complexity of the systems. Traditionally, uncertainty quantification has been approached using random sampling methods like Monte Carlo. For some problems, the input parametric space and corresponding quantity of interest output space is sufficiently explored with a few low-cost computational calculations. For others, however, the computational model is costly and it takes a great many random samples 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 isotropic tensor product (TP) SCgPC, then expand to consider truncated polynomial spaces using total degree (TD) and hyperbolic cross (HC) construction strategies. Next, we consider applying anisotropy to the polynomial space construction, and analyze methods whereby the level of anisotropy can be approximated. We analyze these methods on a series of models, including two analytic models, some simple projectile physics, and a nontrivial neutron diffusion transport model. For this analysis, we demonstrate the application of the algorithms discussed in `Raven`, a production-level uncertainty quantification framework.

Finally, we propose several improvements to the algorithms discussed. First, we propose developing Sobol decomposition, or the high-dimensional model representation (HDMR) method, wherein the uncertainty space is divided into constituent subspaces. Second, we propose developing adaptive algorithms for the intelligent construction of both the SCgPC and HDMR methods. Lastly, we propose all algorithms discussed in this proposal be implemented in `Raven` and tested on an engineering-scale multiphysics model coupling neutronics and nuclear fuel performance.

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). Typically, a solver accepts a set of inputs and produces a set of outputs, often single-valued. 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 have been uncorrelated from one another.

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 QoI

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

While Monte Carlo sampling is effective at developing an understanding of the response, the sheer volume of samples needed to improve that understanding is sometimes prohibitive. One way to reduce the number of samples is by a Latin Hypercube Sampling (LHS) approach. In this method, the input domain is divided into orthogonal hypervolumes. For instance, for a problem with two uncertain input parameters, the corresponding input space is two-dimensional, and the LHS grid would appear as a Cartesian grid of the input space. Once the grid is constructed, a random sample is taken. Once a sample is taken in one of the grid locations, no additional samples can be taken that are within the same uncertainty range in any dimension. In the two dimensional example, once a point is sampled, no additional points can be sampled in the same row or column as the sampled points. In this way, the samples are distributed more thoroughly throughout the input space, and may provide a more clear response with less realizations than analog Monte Carlo.

There are some beneficial properties to both Monte Carlo and LHS methods. Significantly, they are unintrusive; that is, 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.

However, both Monte Carlo and LHS sampling are relatively slow to converge on the response surface. For example, with Monte Carlo sampling, in order to reduce the error in the mean value of the response by a factor of two, it is necessary to take 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 run sufficient realizations to obtain a clear response.

In this work, 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. We then apply the methods to an intermediate-level solver for range of a projectile including drag. This model is nonlinear and has finite regularity, and has no analytic solution. These properties challenge the methods while still maintaining a fast and simple solver. Finally, we apply the methods to an engineering-scale solver that models the neutronics of a nuclear reactor core. While not as complex as a multiphysics model, this will demonstrate the “real life” 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 isotropic, tensor-product stochastic collocation for generalized polynomial chaos (SCgPC)[1][2][3][4], 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 multidimensional 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 and LHS, 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 [5], 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 total degree (TD) and hyperbolic cross (HC) polynomial set construction methods[6]. These bases will then be used to construct Smolyak-like sparse grids [7] 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.

Based on the preliminary work demonstrated here, we propose several improvements. First, we propose implementing high-dimension model representation (HDMR) as a method to decompose the input uncertainty space into low-dimensional subspaces [8]. This method, sometimes referred to as Sobol decomposition, provides two significant functions. First, this method provides the *Sobol sensitivities*, which are parameters

based on analysis of variance (ANOVA). Sobol sensitivities indicate the sensitivity of the response QoI to changes in each subset space. The first-order sensitivities can be used to determine suitable anisotropic weighting for SCgPC. Additionally, HDMR acts as a reduced-order model for the solver, potentially providing more efficient evaluations.

Additionally, we propose continued work on developing adaptive algorithms for both SCgPC and HDMR[9]. In adaptive SCgPC, the polynomial basis is constructed level-by-level based on the highest-impact subset polynomials. In adaptive HDMR, the constituent subset input spaces are developed similarly, based on the highest-impact input subset. The crowning achievement we propose is combining HDMR and SCgPC to develop both the subset input space as well as the overall reduced-order model adaptively in an attempt to construct a competitively-efficient method for uncertainty quantification.

Finally, we propose all these methods be developed within Idaho National Laboratory's `Raven`[10] 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 propose 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 propose to use the coupled `RattleSNake`[11] and `Bison`[12][13] production codes. Both of these codes are developed and maintained within the `Moose`[14] environment. The multiphysics nonlinear system will provide 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 mathematically describe the problems solved by the simulations we will be running, including polynomial evaluations, attenuation, projectile, neutronics, and rector performance. We discuss their potential approach and applications.
- Chapter 3: We describe several methods for uncertainty quantification, including Monte Carlo, Latin Hypercube sampling, and generalized Polynomial Chaos. We also discuss methods to accelerate the convergence of SCgPC.
- Chapter 4: We analyze results obtained thus far for SCgPC methods, and contrast them with traditional Monte Carlo convergence on statistical moments.
- Chapter 5: We discuss proposed work both with HDMR and extending SCgPC to be constructed adaptively. We also discuss the predicted shortfalls in the adaptive algorithms and some potential methods to address them.

Chapter 2

Simulation Physics

2.1 Simulations Used

To demonstrate the efficacy of the various uncertainty quantification methods we use in this work, we employ several independent simulations (hereafter referred to as "codes") of increasing complexity. These codes range from simple polynomial evaluations to analytic solutions of simple physics, to nonlinear multistage iterative solvers representing complex codes. We describe each briefly here.

2.2 Polynomial Evaluations

In order to benchmark the simplest cases, we make use of simple polynomial expressions of the form

$$u(Y) = \prod_{n=1}^{N} (y_n + 1), \tag{2.1}$$

where $u(Y)$ is the quantity of interest and $Y = [y_1, y_2, \ldots, y_N]$ is the vector of uncertain inputs. The input variables Y can be distributed arbitrarily; we distribute them uniformly from -1 to 1. This results in an isotropic hypercube domain extending from -1 to 1 in all dimensions. Because this model is a sum of polynomials, SCgPC can represent it exactly. Because it is highly regular, we also expect SCgPC to be efficient.

2.3 Attenuation

To demonstrate the convergence rates of various methods, we make use of an adjusted attenuation problem that is equivalent to the penetration of neutral particles in a one-dimensional purely-absorbing medium. For this problem we set the physical domain to $x \in [0, 1]$, with the material occupying all of the domain space. If m particles are incident on a material with length dx, the resulting change in the number of particles dm due to absorption is

$$dm = -ymdx, \tag{2.2}$$

where y is the absorption cross section (probability of absorption per unit length) of the neutral particles in the material. The solution for the fraction of particles still present after passing through the material for any position x is

$$m(x) = e^{-yx}. \tag{2.3}$$

To expand the number of input parameters to the problem, we subdivide the material into N independent materials indexed by n, each with absorption cross section y_n and length $h = \frac{1}{N}$. We introduce uncertainty by allowing each absorption cross section to vary uniformly between 0 and 1, resulting in a hypercube uncertainty domain. We establish our quantity of interest as the percent of incident particles at the left domain ($m(0)$) that remain after passing completely through the material ($m(L)$). Introducing the uncertain material properties as parameters for the quantity of interest, we define

$$u(Y) = m(L, Y), \qquad Y \equiv [y_1, \ldots, y_N]. \tag{2.4}$$

This subdivided domain model can be considered as distinct attenuation problems, where the fraction of particles passing through one material are the full number incident on the next material. Thus, the solution is a product of the solution to the one-dimension ODE,

$$u(Y) = m_1(h, y_2) \cdot m_2(h, y_2) \cdot \ldots \cdot m_N(h, y_N), \tag{2.5}$$

$$= \prod_{n=1}^{N} e^{-hy_n}, \tag{2.6}$$

$$= \prod_{n=1}^{N} e^{-y_n/N}. \tag{2.7}$$

The benefit of this model is a simple analytic solution, which makes analytic benchmarks straightforward to compute. In addition, because of the exponential form, SCgPC will converge on a solution, but not replicate it exactly as in the polynomial case. This allows

accurate comparison between MC, SCgPC methods, and HDMR methods in a highly regular analytic case.

2.4 Projectile

For a nonlinear case without an analytic solution, we consider the path traveled by a projectile near the surface of the Earth, considering drag on the projectile from stagnant air. The physical domain for this problem extends from $x = 0, y = 0$ as far as necessary for the projectile to reach its apex and return to $y = 0$. The equations governing travel in both vertical position x and horizontal position y are given by

$$y(t) = \frac{v_T}{g}(v \sin\theta + v_T)\left(1 - \exp\left[\frac{-gt}{v_T}\right]\right) - v_T t, \tag{2.8}$$

$$x(t) = \frac{v v_T \cos\theta}{g}\left(1 - \exp\left[\frac{-gt}{v_T}\right]\right), \tag{2.9}$$

where t is time, g is acceleration due to gravity, v is scalar velocity, θ the angle between the velocity vector and the horizontal ground (x-axis), $v_T = \frac{mg}{D}$ is terminal velocity, $D = \frac{\rho C A}{2}$ is the acceleration due to drag, C is the drag coefficient, and $A = \pi r^2$ is the surface area of the projectile in the direction of travel. The projectile is assumed to present an identical surface area in both x and y directions. The quantity of interest is the range, or the total distance in x traveled by the ball before reaching a height of $y = 0$. The uncertain input variables are distributed uniformly as described in Table 2.1.

Variable	Name	Mean	Range (±)	Units
y_i	Initial Height	1	1	m
v	Initial Velocity	35.5	2.5	m/s
θ	Initial Angle	45	10	degrees
g	Accel. Gravity	9.79888	0.0349	m/s/s
m	Projectile Mass	0.145	0.0725	kg
r	Projectile Radius	0.0336	0.00336	m
C	Drag Coefficient	0.5	0.5	
ρ	Air Density	1.2	0.1	kg/m^3

TABLE 2.1: Projectile Problem Distributions

This simulation has the benefit of an analytic solution when $C = 0$ and eight distinct input parameters of varying importance. This is especially useful in considering anisotropic treatment of the input space. The regularity of the response is not well known, as there is no analytic solution to this case; however, we expect the range to be less regular than the quantity of interest in the previous two cases.

2.5 Neutron Diffusion

For a nonlinear system with complicated physics, we consider a two-group, two-dimensional neutron diffusion criticality calculation. The physical domain for this problem extends along $x \in [0, 165 \text{ cm}]$ and $y \in [0, 165 \text{ cm}]$. We make use of the diffusion approximation for neutron transport, which when restricted to two energy groups provides us with a coupled set of elliptic PDEs to solve:

$$-\boldsymbol{\nabla} \cdot (D_1(\bar{x})\boldsymbol{\nabla}\phi_1(\bar{x})) + \left(\Sigma_a^{(1)}(\bar{x}) + \Sigma_s^{(1\to 2)}(\bar{x})\right)\phi_1(\bar{x}) = \frac{1}{k(\phi)} \sum_{g'=1}^{2} \nu_{g'} \Sigma_f^{(g')}(\bar{x})\phi_{g'}(\bar{x}), \tag{2.10}$$

$$-\boldsymbol{\nabla} \cdot (D_2(\bar{x})\boldsymbol{\nabla}\phi_2(\bar{x})) + \Sigma_a^{(2)}(\bar{x})\phi_2(\bar{x}) = \Sigma_s^{(1\to 2)}(\bar{x})\phi_1(\bar{x}), \tag{2.11}$$

where we use the following parametric coefficients: the absorption cross section $\Sigma_{g,a} = \Sigma_{g,c} + \Sigma_{g,f}$; the capture and fission cross sections $\Sigma_{g,c}$ and $\Sigma_{g,f}$; the energy-group scattering cross section $\Sigma_s^{(g'\to g)}$; the diffusion coefficient D_g which depends on the scattering cross section of the medium; and the fission multiplication factor ν_g, the ratio of new neutrons per fission-producing neutron. The solution to this PDE is the neutron scalar flux $\phi_g(\bar{x})$. We apply no-traction conditions on the vacuum boundaries and zero-derivative current on the reflecting boundaries for both energy groups:

$$\frac{\phi_g}{4} - \frac{D_g}{2} \frac{\partial \phi_g}{\partial x_1}\bigg|_{\partial\Omega_{\text{top}}} = 0, \;\; g = 1, 2, \tag{2.12}$$

$$\frac{\phi_g}{4} - \frac{D_g}{2} \frac{\partial \phi_g}{\partial x_2}\bigg|_{\partial\Omega_{\text{right}}} = 0, \;\; g = 1, 2, \tag{2.13}$$

$$-D_g \frac{\partial \phi_g}{\partial x_1}\bigg|_{\partial\Omega_{\text{bottom}}} = 0, \;\; g = 1, 2, \tag{2.14}$$

$$-D_g \frac{\partial \phi_g}{\partial x_2}\bigg|_{\partial\Omega_{\text{left}}} = 0, \;\; g = 1, 2. \tag{2.15}$$

The criticality eigenvalue and quantity of interest $k(\phi)$ is given by

$$k(\phi) = \sum_{g=1}^{2} \iint_D \frac{\nu\Sigma_f^{(g)}\phi_g(\bar{x})}{\left(-\nabla \cdot D_g \nabla + \Sigma_r^{(g)}\right)\phi_g(\bar{x})} \, d\bar{x}. \tag{2.16}$$

We address solving ϕ_1, ϕ_2, and k nonlinearly and simultaneously. The material properties are shown in Table 2.2, and the domain $\Omega = [0, 200 \text{ cm}]^2$. The reference flux solutions are plotted in Fig. 2.2, and for the reference problem k=1.00007605445.

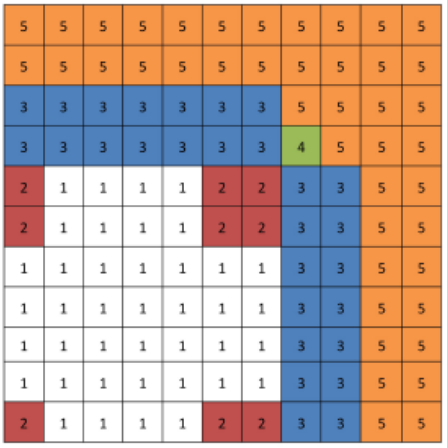
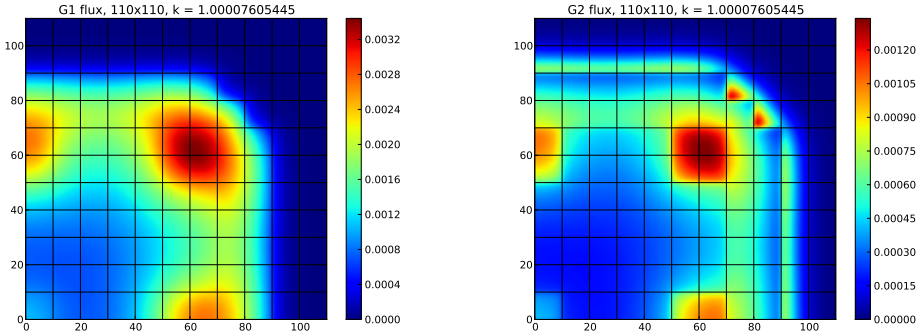


FIGURE 2.1: Core Geometry

Region	Group	D_g	$\Sigma_{a,g}$	$\nu\Sigma_{f,g}$	$\Sigma_s^{1,2}$
1	1	1.255	8.252e-3	4.602e-3	2.533e-2
	2	2.11e-1	1.003e-1	1.091e-1	
2	1	1.268	7.181e-3	4.609e-3	2.767e-2
	2	1.902e-1	7.047e-2	8.675e-2	
3	1	1.259	8.002e-3	4.663e-3	2.617e-2
	2	2.091e-1	8.344e-2	1.021e-1	
4	1	1.259	8.002e-3	4.663e-3	2.617e-2
	2	2.091e-1	7.3324e-2	1.021e-1	
5	1	1.257	6.034e-4	0	4.754e-2
	2	1.592e-1	1.911e-2	0	

TABLE 2.2: Reference Material Properties for Benchmark Core



(A) ϕ, Group 1 (B) ϕ, Group 2

FIGURE 2.2: Reference Flux Profiles

Chapter 3

Uncertainty Quantification Techniques

3.1 Uncertainty Quantification Methods

We consider a few methods for uncertainty quantification (UQ). One method to classify UQ methods is by their interaction level with a simulation code. Non-intrusive methods treat the code as a black box, perturbing the inputs and collecting the outputs without modifying the solver code. These methods are ideal for generic application frameworks, where the simulation code may be unknown or precompiled. Examples of non-intrusive methods include analog Monte Carlo (MC), Latin Hypercube sampling (LHS), and some deterministic collocation methods. Alternatively, intrusive methods require access to the solution algorithm itself. Sometimes this can provide more efficient solutions. In particular, adjoint methods require the solution operator to be reversible, and provide very efficient methods to determine sensitivities and analyze output-input dependence. While many intrusive methods have benefits, they lack the flexibility and universal applicability of non-intrusive methods, and so we focus strictly on non-intrusive methods in this work.

3.2 Correlation

An assumption we make in this work is that the uncertain input parameters are independent or at least uncorrelated. If this is not the case, methods such as Karhunen-Loeve expansion can be used to develop an orthogonal input space to replace the original. Similarly, principle component analysis can be used to attempt to find the fundamental uncorrelated space that maps into the correlated variable space.

3.3 Monte Carlo

In analog Monte Carlo uncertainty quantification, a single point in the input space is selected randomly, and an output collected, until an appropriately accurate view of the output space is acquired. While few samples result in a poor understanding of the quantity of interest, sufficiently large samples converge on a correct solution. The convergence rate of Monte Carlo is consistent, as

$$\epsilon = \frac{c}{\sqrt{\eta}}, \tag{3.1}$$

where c is a constant and η is the number of samples taken. While this convergence rate is slow, it is possibly one of the most reliable methods available. This makes MC a good choice for benchmarking.

One of the downsides of MC (and LHS) when compared with other methods is that they do not generate a reduced-order model as part of the evaluation; however, interpolation methods can be used to generate additional samples.

3.4 Latin Hypercube Sampling

Latin hypercube sampling[15] is another stochastic method that specializes in multi-dimensional distributions. In this method, the input domain is divided into M^N sub-domains that contain an equal probability volume, with M divisions along each axis. Sampling is then performed in these volumes, assuring no two samples share the same "row" for a given input dimension; that is, once a sample is taken within the i-th subdivision of an input range (axis), another sample may not be taken that is within that subdivision for that variable. Thus, subdividing each axis into M sections results in M total possible samples.

LHS is useful as a sampling strategy because it assures samples are distributed throughout the entirety of the input space, which is less guaranteed using MC sampling. There is some cost, however, associated with subdividing and tracking the grid on the input space.

3.5 Generalized Polynomial Chaos

In general, polynomial chaos expansion (PCE) methods seek to represent the simulation code as a combination of polynomials of varying degree and combination in each

dimension of the input space. Originally Wiener proposed expanding in Hermite polynomials for Gaussian-normal distributed variables [16]. Askey and Wilson generalized this method 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[17].

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)[4]. Two significant methodologies have grown from gPC application. The first makes use of Lagrange polynomials to expand the original function or simulation code, as they can be made orthogonal over the same domain as the distributions[18]; the other uses the Wiener-Askey polynomials[4]. We consider the latter in this work.

We consider a simulation code that produces a quantity of interesy 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 and without analytic solution. There may be other input parameters that contribute to the solution of $u(Y)$; we neglect these, as our interest is chiefly in the uncertainty space. 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; in any case, we restrict $u(Y(\omega))$ to a single scalar output.

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

$$\Phi_k(Y) = \prod_{n=1}^{N} \phi_{k_n}(Y_n), \tag{3.2}$$

where $\phi_{k_n}(Y_n)$ is a single-dimension Wiener-Askey orthonormal polynomial of order k_n and $k = [k_1, \ldots, k_n, \ldots, k_N]$. The gPC for $u(Y)$ using this notation is

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

where u_k is a weighting polynomial coefficient. The polynomials used in the expansion are determined by the set of multi-indices $\Lambda(L)$, where L is a truncation order for

isotropic methods. In the limit that Λ contains all possible combinations of polynomials of any order, Eq. 3.2 is exact. Practically, however, Λ is truncated to some finite set of combinations, discussed in section 3.5.1.

Using the orthonormal properties of the Wiener-Askey polynomials,

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

where $\rho(Y)$ is the combined PDF of Y, Ω is the multidimensional domain of Y, and δ_{nm} is the Dirac delta, we can isolate an expression of the polynomial expansion coefficients. We multiply both sides of Eq. 3.2 by $\Phi_{\hat{k}}(Y)$, integrate both sides over the probability-weighted input domain, and sum over all $\hat{k}$ to obtain the coefficients, sometimes referred to as polynomial expansion moments,

$$u_k = \frac{\langle u(Y)\Phi_k(Y)\rangle}{\langle \Phi_k(Y)^2\rangle}, \tag{3.5}$$

$$= \langle u(Y)\Phi_k(Y)\rangle, \tag{3.6}$$

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

$$\langle f(Y)\rangle \equiv \int_{\Omega} f(Y)\rho(Y)dY. \tag{3.7}$$

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

3.5.1 Polynomial Index Set Construction

There are many ways by which a polynomial index set can be constructed. Here we present three: 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

$$\Lambda_{\mathrm{TP}}(L) = \left\{\bar{p} = [p_1, \cdots, p_N] : \max_{1\leq n\leq N} p_n \leq L\right\}. \tag{3.8}$$

The cardinality of this index set is $|\Lambda_{\mathrm{TP}}(L)| = (L+1)^N$. For example, for a two-dimensional input space (N=2) and truncation limit $L = 3$, the index set $\Lambda_{\mathrm{TP}}(3)$ is

given in Table 3.1, where the notation $(1, 2)$ signifies the product of a polynomial that is first order in Y_1 and second order in Y_2.

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

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

It is evident there is some inefficiencies in this index set. First, it suffers dramatically from the *curse of dimensionality*; that is, the number of polynomials required grows exponentially with increasing dimension. Second, the total order of polynomials is not considered. Assuming the contribution of each higher-order polynomial is smaller than lower-order polynomials as per gPC, 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) index set construction strategies[6].

In TD, only multidimensional polynomials whose *total* order is less than L are permitted, as

$$\Lambda_{\mathrm{TD}}(L) = \left\{\bar{p} = [p_1, \cdots, p_N] : \sum_{n=1}^{N} p_n \leq L\right\}. \tag{3.9}$$

The cardinality of this index set is $|\Lambda_{\mathrm{TD}}(L)| = \binom{L+N}{N}$, which grows with increasing dimension much more slowly than TP. For the same $N = 2, L = 3$ case above, the TD index set is given in Table 3.3.

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

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

In HC, the *product* of polynomial orders is used to restrict allowed polynomials in the index set. This tends to polarize the expansion, emphasizing higher-order polynomials in each dimension but lower-order polynomials in combinations of dimensions, as

$$\Lambda_{\mathrm{HC}}(L) = \left\{\bar{p} = [p_1, \ldots, p_N] : \prod_{n=1}^{N} p_n + 1 \leq L + 1\right\}. \tag{3.10}$$

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 **??** for $N = 2, L = 3$.

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

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

It has been shown that the effectiveness of TD and HC as index set choices depends strongly on the regularity of the output space[6]. TD tends to be most effective for infinitely-continuous response surfaces, while HC is more effective for surfaces with limited smoothness or discontinuities.

3.5.2 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 a problem for problems of large dimension. Another method that can be applied to mitigate dimensionality problem is anisotropy, or the unequal treatment of multiple dimensions. In this strategy, weighting factors $\alpha = [\alpha_1, \ldots, \alpha_n, \ldots, \alpha_N]$ are applied in each dimension to allow additional polynomials in some dimensions and less in others. This change adjusts the TD and HC construction rules as follows, where $|\alpha|_1$ is the one-norm of α.

$$\tilde{\Lambda}_{\mathrm{TD}}(L) = \left\{\bar{p} = [p_1, \cdots, p_N] : \sum_{n=1}^{N} \alpha_n p_n \leq |\alpha|_1 L\right\}, \tag{3.11}$$

$$\tilde{\Lambda}_{\mathrm{HC}}(L) = \left\{\bar{p} = [p_1, \cdots, p_N] : \prod_{n=1}^{N} (p_n + 1)^{\alpha_n} \leq (L+1)^{|\alpha|_1}\right\}. \tag{3.12}$$

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

$$|\alpha|_1 = \frac{\sum_{n=1}^{N} \alpha_n}{N}. \tag{3.13}$$

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

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

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

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

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

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

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, discussed later, could also serve as a basis for these weights. As will be shown, a good choice of anisotropy weight can greatly speed up convergence; however, a poor choice can slow convergence considerably, as computational resources are used to resolve low-importance dimensions.

3.6 Stochastic Collocation

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

$$\int_a^b f(x)\rho(x) = \sum_{\ell=1}^{\infty} w_\ell f(x_\ell), \tag{3.14}$$

$$\approx \sum_{\ell=1}^{L} w_\ell f(x_\ell), \tag{3.15}$$

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

$$q^{(\hat{L})}[f(x)] \equiv \sum_{\ell=1}^{\hat{L}} w_\ell f(x_\ell). \tag{3.16}$$

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

$$Q^{(\mathbf{L})} = q_1^{(\hat{L}_1)} \otimes q_2^{(\hat{L}_2)} \otimes \cdots, \tag{3.17}$$

$$= \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}. \tag{3.18}$$

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. ¡TODO¿ A summary of the Gaussian quadratures and corresponding probability distribution weight functions are described in an appendix ¡/TODO¿. For convenience we repeat here the coefficient integral we desire to evaluate, Eq. 3.5.

$$u_k = \langle u(Y)\Phi_k(Y)\rangle. \tag{3.19}$$

We can approximate this integral with the appropriate Gaussian quadrature as

$$u_k \approx Q^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)], \tag{3.20}$$

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

$$u_k \approx q^{(\hat{L})}[u(Y)\Phi_k(Y)], \tag{3.21}$$

$$= \sum_{\ell=1}^{\hat{L}} w_\ell u(Y_\ell)\Phi_k(Y_\ell). \tag{3.22}$$

In order to determine the quadrature order $\hat{L}$ needed to accurately integrate this expression, we consider the gPC formulation for $u(Y)$ in Eq. 3.2 and replace it in the sum,

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell \Phi_k(Y_\ell) \sum_{k\in\Lambda(L)} u_{\hat{k}}\Phi_{\hat{k}}(Y_\ell). \tag{3.23}$$

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

$$u_k \approx \sum_{\ell=1}^{\hat{L}} w_\ell u_k \Phi_k(Y_\ell)^2. \tag{3.24}$$

Thus, the integral, to the same approximation that the gPC expansion itself is approximated, the quadrature is approximating an integral of order $2k$, and the quadrature order itself should be order

$$p = \frac{2k+1}{2} = k + \frac{1}{2} < k+1, \tag{3.25}$$

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

be a poor approximation.

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

3.6.1 Smolyak Sparse Grids

Smolyak sparse grids[7] are an attempt to discover the smallest necessary quadrature set to integrate a multidimensional integral with varying orders of predetermined quadrature sets. In our case, the polynomial index sets determine the quadrature orders each one needs in each dimension to be integrated accurately. For example, the polynomial index set point (2,1,3) requires three points in Y_1, two in Y_2, and four in Y_3,or

$$Q^{(2,1,3)} = q_1^{(3)} \otimes q_2^{(2)} \otimes q_3^{(4)}. \tag{3.26}$$

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

$$Q^{(\Lambda(L))} = \bigotimes_{k \in \Lambda} Q^{(k)}. \tag{3.27}$$

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

$$\Delta_k^{(\hat{L})}[f(x)] \equiv \left(q_k^{(\hat{L})} - q_{k-1}^{(\hat{L})}\right)[f(x)], \tag{3.28}$$

$$q_0^{(\hat{L})}[f(x)] = 0. \tag{3.29}$$

A Smolyak sparse grid then be defined and applied to the desired integral in Eq. 3.5,

$$S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)] = \sum_{k \in \Lambda(L)} (\Delta_{k_1}^{(\hat{L}_1)} \otimes \cdots \otimes \Delta_{k_N}^{(\hat{L}_N)}[u(Y)\Phi_k(Y)]. \tag{3.30}$$

Equivalently, and in a more algorithm-friendly approach,

$$S_{\Lambda,N}^{(\hat{\mathbf{L}})}[u(Y)\Phi_k(Y)] = \sum_{k \in \Lambda(L)} c(k) \bigotimes_{n=1}^{N} q_n^{(\hat{L}_n)}[u(Y)\Phi_k(Y)] \tag{3.31}$$

where

$$c(k) = \sum_{\substack{j=\{0,1\}^N, \\ k+j \in \Lambda}} (-1)^{|j|_1}, \tag{3.32}$$

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

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

$$\approx S^{(\hat{\mathbf{L}})}_{\Lambda,N}[u(Y)\Phi_k(Y)], \tag{3.34}$$

and gPC for $u(Y)$ is complete.

3.7 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[8], or Sobol decomposition. In particular, we consider Cut-HDMR in this work. In this methodology, the representation of $u(Y)$ is built up from the contributions of subsets of the input space. We consider a reference point realization of Y (nominally the mean of each distribution) $Y^{(0)} = [Y_1^{(0)}, \ldots, Y_N^{(0)}]$ and introduce the notation

$$\hat{Y}_n \equiv [Y_1, \ldots, Y_{n-1}, Y_{n+1}, \ldots, Y_N], \tag{3.35}$$

and

$$u(Y_n)\Big|_{\hat{Y}_n} \equiv u(\hat{Y}_n^{(0)}, Y_n). \tag{3.36}$$

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

$$u(Y_m, Y_n)\Big|_{\hat{Y}_{m,n}} \equiv u(\hat{Y}^{(0)}_{m,n}, Y_m, Y_n), \tag{3.37}$$

$$= u(Y_1^{(0)}, \ldots, Y_{m-1}^{(0)}, Y_m, Y_{m+1}^{(0)}, \ldots, Y_{n-1}^{(0)}, Y_n, Y_{n+1}^{(0)}, \ldots, Y_N^{(0)}). \tag{3.38}$$

The HDMR reduced-order model is then constructed as

$$\begin{aligned} u(Y) = u_0 &+ \sum_{n=1}^{N} u_{Y_n} \\ &+ \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} u_{Y_{n_1}, Y_{n_2}} \\ &+ \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} \sum_{n_3=1}^{n_2-1} u_{Y_{n_1}, Y_{n_2}, Y_{n_3}} \\ &\cdots \qquad (3.39) \\ &+ u_{Y_{n_1}, \cdots, Y_{n_N}}, \qquad (3.40) \end{aligned}$$

where

$$u_0 = u(Y^{(0)}), \tag{3.41}$$

$$u_{Y_n} = u(Y_n)\Big|_{Y_n} - u_0, \tag{3.42}$$

$$u_{Y_m,Y_n} = u(Y_m, Y_n)\Big|_{Y_m,Y_n} - u_{Y_m} - u_{Y_n} - u_0, \tag{3.43}$$

$$\dots \tag{3.44}$$

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

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

$$s_n = \frac{\mathrm{var}(u_{Y_n})}{\mathrm{var}(u(Y))}, \tag{3.45}$$

and for second-order,

$$s_{m,n} = \frac{\mathrm{var}(u_{Y_m,Y_n})}{\mathrm{var}(u(Y))}, \tag{3.46}$$

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

Chapter 4

Preliminary Results

We consider some results from applying the methods described in Chapter 3 to the physical models outlined in Chapter 2, organized by model.

4.1 Polynomial Evaluations

Using an analytic polynomial evaluation as our model allows observation of best-case performance of SCgPC for several dimension cardinalities. Uniformly distributing from 0 to 1, the mean and variance are

$$\text{mean} = \left(\frac{3}{2}\right)^N, \tag{4.1}$$

$$\text{var} = \left(\frac{7}{3}\right)^N - \left(\frac{3}{2}\right)^{2N}, \tag{4.2}$$

where N is the cardinality of the input space. As seen in Figures 4.1 and 4.1, the increase in dimensionality has great impact on the convergence rate of SCgPC methods, but their exponential convergence eventually does beat out Monte Carlo. Also of interest is the adaptive methods. In the figures, *adapt* refers to adaptive Legendre quadrature sparse grid, while *adapt_cc* refers to adaptive Clenshaw Curtis sparse grid. While Legendre quadrature is traditionally considered ideal for polynomial fitting, the Clenshaw Curtis quadrature seems to perform very similarly due to the benefit of nested quadrature points.

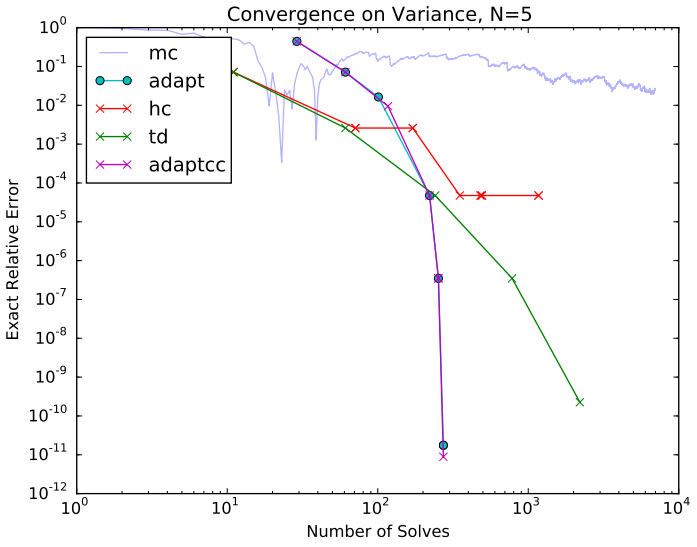


FIGURE 4.1: Analytic $N = 5$ Error Convergence, Variance

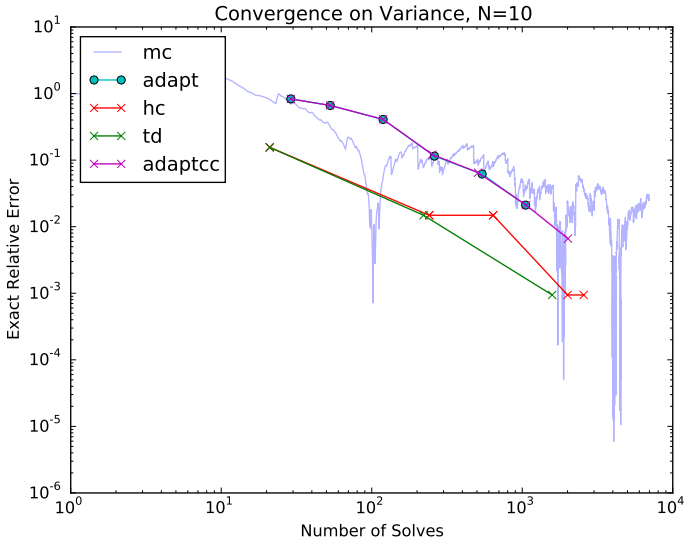


FIGURE 4.2: Analytic $N = 10$ Error Convergence, Variance

4.2 Attenuation

Similar to the polynomial model, the attenuation model does not converge exactly for any order of SCgPC expansion, as exponential terms cannot be perfectly represented by a finite sum of polynomials. Again distributing the input variables from 0 to 1, the

mean and variance are

$$\text{mean} = N^N \left(1 - \exp\left(-\frac{1}{N}\right)\right)^N, \tag{4.3}$$

$$\text{var} = \left(\frac{N}{2}\right)^N \left(1 - \exp\left(-\frac{2}{N}\right)\right)^N - \text{mean}^2, \tag{4.4}$$

where N is the cardinality of the input space. The convergence plots are shown in Figs. 4.3 and 4.4. Interestingly, while the convergence of the adaptive method seems to be converging at a better rate than the total degree isotropic set, there is no clear advantage for up to thousands of solves. This is somewhat expected, as there is little coupling between inputs in the attenuation model. As with the polynomial model, the curse of dimensionality is clear.

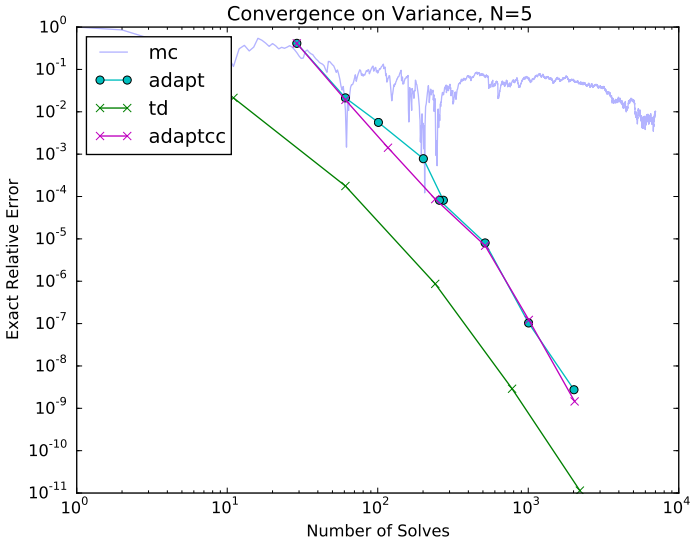


FIGURE 4.3: Attenuation $N = 5$ Error Convergence, Variance

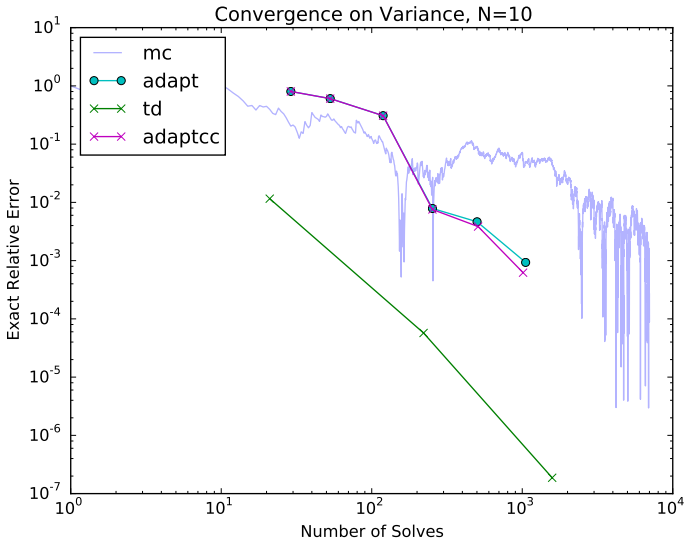


FIGURE 4.4: Attenuation $N = 10$ Error Convergence, Variance

4.3 Projectile

Unlike the attenuation and polynomial models, the projectile model is a nonlinear problem without an analytic solution. As a result, the convergence benchmark is one achieved by a large Monte Carlo run, instead of an analytic benchmark; thus, the convergence of the various methods is limited to the convergence of the Monte Carlo point. The most converged Monte Carlo point for this work is

$$\text{mean} = 1234, \tag{4.5}$$

$$\text{var} = 1234. \tag{4.6}$$

Additionally, this is the first model where significant anisotropy exists in the model. For instance, the drag coefficient parameter is many orders of magnitude more influential on the quantity of interest than the gravitational acceleration. The results in Fig. 4.5 are interesting, and require additional data points to understand clearly. However, we include Fig. 4.6 for clarity. While it initially looks like the adaptive methods converge quickly, it appears that this is misleading, as shown in the values graph. The adaptive method is converging, but passes through the benchmark value before turning back to converge on it. This indicates the adaptive construction introduces too much variance initially, and additional terms are required to approach the benchmark solution.

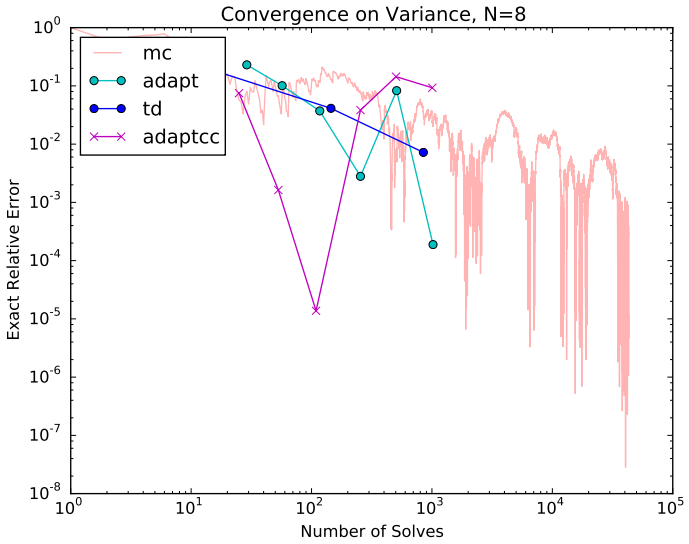


FIGURE 4.5: Projectile $N = 8$ Error Convergence, Variance

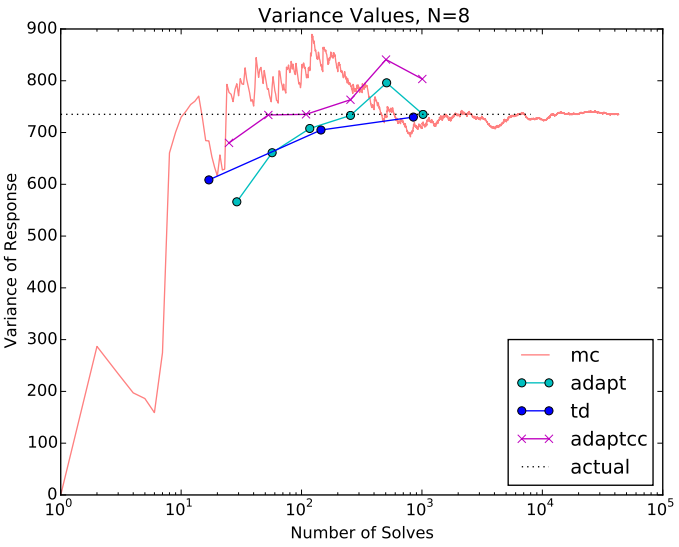


FIGURE 4.6: Projectile $N = 8$ Values, Variance

4.4 Neutron Diffusion

The neutron diffusion model is a complex, nonlinear model that begins to approach an engineering-scale model. Because of complications with conflicting libraries, the adaptive SCgPC method was not available for this model. Particularly of note is the clear and

obvious loss of convergence rate with increase in the cardinality of the input space. {Note to self: fix coloring in plots.}

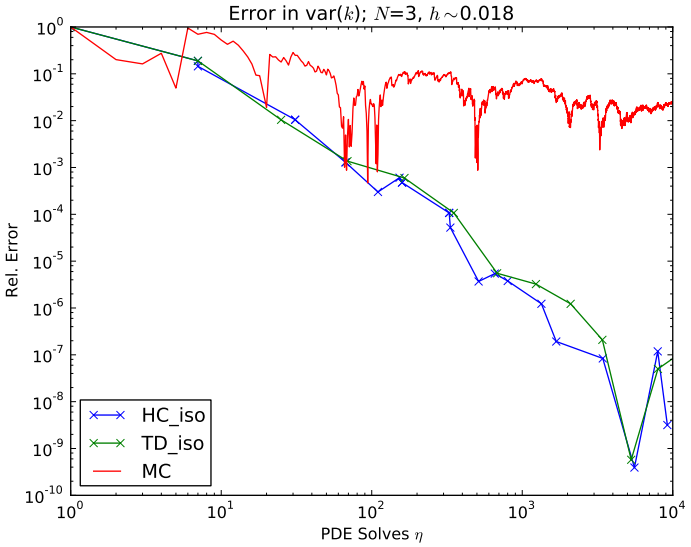


FIGURE 4.7: Diffusion $N = 3$ Error Convergence, Variance

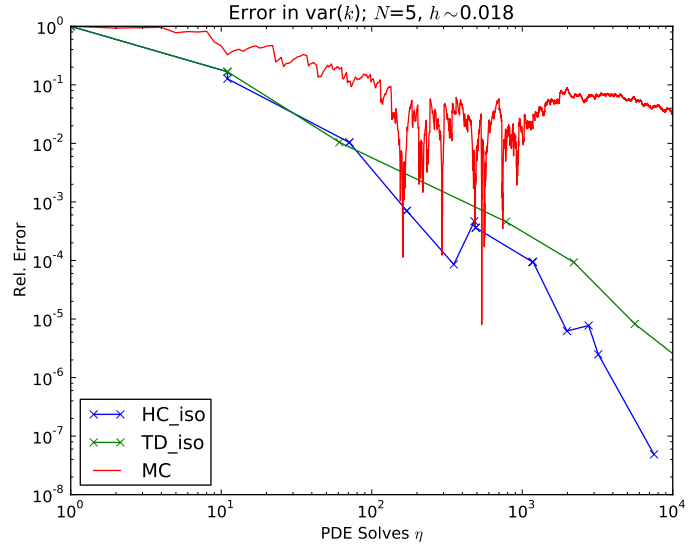


FIGURE 4.8: Diffusion $N = 5$ Error Convergence, Variance

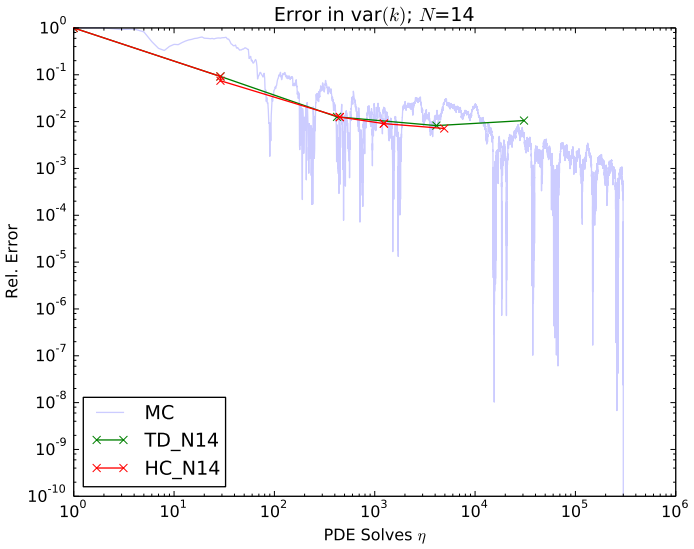


FIGURE 4.9: Diffusion $N = 14$ Error Convergence, Variance

4.5 Fuel Rod Performance

Because of non-deterministic failures in BISON, insufficient data has been collected to present convergence plots for this model.

4.6 Conclusions

From evaluating the results using the various methods on these models, there are a few useful conclusions.

First, especially for models with small input cardinality, SCgPC methods tend to be faster converging than traditional Monte Carlo methods. In particular, the adaptive SCgPC method eventually exhibits exponential convergence in the models considered here. This is encouraging for many uncertainty quantification problems where the variance is chiefly centered on a few input parameters.

Second, using Clenshaw Curtis quadrature appears to have no negative consequence on converging with the adaptive SCgPC method, but no striking positive consequence either. This suggests it is worth exploring other nested quadrature methods that might be beneficial in general, or at least in certain circumstances.

Third, it is clear that even with adaptive SCgPC, convergence is poor for input spaces with more than a dozen inputs for less than a thousand solves. For costly engineering codes, even a thousand runs may be impractical, so further improvement of the adaptive algorithm is necessary. This leads to the desirability of the adaptive HDMR method, which can further subdivide the input domain. These subdivided domains have cardinality much more suitable to SCgPC methods.

Lastly, because SCgPC methods improve drastically with decreases in input cardinality, we expect SCgPC to benefit from coupling with input reduction methods, such as sensitivity-weighted input reduction through principal component analysis using input-input covariance matrices.

Chapter 5

Proposed Work

Given the results in Chapter 4, we consider the proposed work to complete as part of completing degree work.

5.1 DRAFT NOTES

- I've included results from the Adaptive Sparse Grid sampler in the Results section as well as this proposal section, as I'm not confident which section we want it to belong to. I can move the "proposed" to "methods" if we want to showcase it for the proposal.
- The colors for the first two convergence plots are off for the Diffusion model, but that should be easy to fix for the final draft.
- There's no results for the adaptive sparse grid for the diffusion model, as I haven't been able to run the diffusion model in RAVEN and my pre-RAVEN UQ framework doesn't include the adaptive methodology. There's an outside chance I could get the two working together on a clean-built machine.

5.2 Adaptive Sparse Grid

There is a method proposed by [19] and used in [9] whereby the polynomial index set to be used in constructing the sparse grid quadrature is constructed adaptively. 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 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.

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 [9]. This is somewhat inefficient, because of SCgPC representations created that are not used in the final product. In addition to including this algorithm, we propose to develop a method for predicting high-impact points based on the impact of their predecessors in the set.

Second, there are certain types of models for which the adaptive algorithm will stall, converge too early, or otherwise generally 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

$$f(a,b) = a^3 b^3, \tag{5.1}$$

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. In addition to implementing this algorithm, we propose to find a way either to work around this limitation, or at least to warn the user when such a case is likely to exist.

5.3 Adaptive HDMR

Despite the benefits of ideal anisotropic polynomial set construction promised by the adaptive SCgPC algorithm, we expect the adaptive SCgPC to fall short for models where many input parameters provide relevant uncertainty. While the adaptive polynomial index set construction can preferentially emphasize particular dimensions, it does not reduce the input cardinality of the problem. An additional algorithm presented in [9] uses an adaptively-constructed HDMR representation to alleviate the curse of dimensionality for the adaptive SCgPC method.

In Chapter 3, we identify the static HDMR expansion, whereby subsets of the input domain can be joined together to approximate the full model. We replicate the essential expansion here:

$$\begin{aligned} u(Y) = u_0 &+ \sum_{n=1}^{N} u_{Y_n} \\ &+ \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} u_{Y_{n_1},Y_{n_2}} \\ &+ \sum_{n_1=1}^{N} \sum_{n_2=1}^{n_1-1} \sum_{n_3=1}^{n_2-1} u_{Y_{n_1},Y_{n_2},Y_{n_3}} \\ &\dots \\ &+ u_{Y_{n_1},\cdots,Y_{n_N}}, \end{aligned} \tag{5.2}$$

where $u(Y)$ is the model, Y is the combined input space, and u_{Y_n} is an SCgPC surrogate model where only Y_n is considered varying and other input parameters are held at their reference value. In static HDMR, the desired largest subset size is selected, and all subset combinations are considered. For instance, for a model $f(a, b, c)$, the full HDMR

expansion is

$$f(a,b,c,d) = f_0 + f_a + f_b + f_c \tag{5.3}$$
$$+ f_{ab} + f_{ac} + f + bc \tag{5.4}$$
$$+ f_{abc}. \tag{5.5}$$

Truncating, a first-order HDMR expansion includes just the terms in Eq. 5.3, and a second-order expansion additionally includes all the terms in Eq. 5.4. However, it is entirely possible and often likely that some subsets have larger impact than others, and some might be ignored altogether without significantly impacting the overall expansion. Adaptive HDMR (AHDMR) is an algorithm that considers the potential impacts of adding new polynomials to existing subsets (as in the Adpative Sparse Grid algorithm) as well as the potential impact of adding new subsets to the HDMR expansion. The AHDMR algorithm proceeds as follows:

- Evaluate the reference (all mean) case.
- Construct all first-order expansion surrogate models.
- While not converged:
  - Using existing subset sensitivities, predict the importance of future subsets
  - Consider the impact of adding polynomial indices to existing subset PCE representations.
  - Choose between expanding existing subsets or adding new subsets based on impact.
  - If the relative contribution of the new HDMR expectation is less than tolerance, convergence is reached.

All the surrogate models for each subtype in the expansion are constructed using adaptive SCgPC expansions. Because the input space for the subset surrogates is small, and grows slowly as the method progresses, they are ideally suited for SCgPC methods.

For the case demonstrated by [9], combining adaptive HDMR and adaptive SCgPC makes it possible to compete in convergence with Monte Carlo for input spaces with cardinality nearly up to a thousand. Adding more efficient polynomial index set choices for adaptive SCgPC, we expect to see additional increases in performance.

Bibliography

[1] 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.

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

[3] Hans-Joachim Bungartz and Michael Griebel. Sparse Grids. *Acta Numerica*, 13, 2004.

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

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

[6] Erich Novak and Klaus 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.

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

[8] G. Li, C. Rosenthal, and H. Rabitz. High Dimensional Model Representations. *J. Phys. Chem. A*, 105, 2001.

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

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

[11] Y. 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.

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

[13] Frederick Gleicher, Richard Williamson, Javier Ortensi, Yaqi Wang, Benjamin W Spencer, Stephen Novascone, Jason D Hales, and Richard C 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.

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

[15] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, 1979.

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

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

[18] Ivo Babuska, Fabio Nobile, and Raul Tempone. A Stochastic Collocation Method for Elliptic Partial Differential Equations with Random Input Data. *SIAM Journal on Numerical Analysis*, 45, 2007.

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