Discontinuity Detection in Multivariate Space for Stochastic Simulations

Rick Archibald* Anne Gelb† Rishu Saxena‡ Dongbin Xiu§

Abstract

Edge detection has traditionally been associated with detecting physical space jump discontinuities in one dimension, e.g. seismic signals, and two dimensions, e.g. digital images. Hence most of the research on edge detection algorithms is restricted to these contexts. High dimension edge detection can be of significant importance, however. For instance, stochastic variants of classical differential equations not only have variables in space/time dimensions, but additional dimensions are often introduced to the problem by the nature of the random inputs. The stochastic solutions to such problems sometimes contain discontinuities in the corresponding random space and a prior knowledge of jump locations can be very helpful in increasing the accuracy of the final solution. Traditional edge detection methods typically require uniform grid point distribution. They also often involve the computation of gradients and/or Laplacians, which can become very complicated to compute as the number of dimensions increases. The polynomial annihilation edge detection method, on the other hand, is more flexible in terms of its geometric specifications and is furthermore relatively easy to apply. This paper discusses the numerical implementation of the polynomial annihilation edge detection method to high dimensional functions that arise when solving stochastic partial differential equations.

Key Words: stochastic partial differential equations, multivariate edge detection, generalized polynomial chaos method

1 Introduction

There has been recent growing interest in developing numerical methods for stochastic computations. The purpose is to effectively conduct uncertainty quantification (UQ) for practical problems. Uncertainty is ubiquitous in engineering problems and can present itself in mathematical models via "data", e.g., parameters, initial and boundary conditions, material

*Computer Science and Mathematics, ORNL, Oak Ridge, TN 37831. E-mail: *ArchibaldRK@ORNL.gov*

†Department of Mathematics and Statistics, Arizona State University, Tempe, AZ 85287 E-mail: *ag@math.asu.edu*

‡Department of Mathematics and Statistics, Arizona State University, Tempe, AZ 85287 E-mail: *saxena@mathpost.asu.edu*

§Department of Mathematics, Purdue University, West Lafayette, IN 47907 E-mail: *dxiu@math.purdue.edu*

properties, and the like. Due to our inability to conduct accurate measurements, such data can never be known precisely. The associated uncertainty in the data will inevitably affect our prediction of the true physics. In order to conduct reliable computation, it is therefore important to incorporate the uncertainty in the problem from the beginning of the simulation, as well as to understand and quantify the impacts of the uncertainty on the solution.

The most widely adopted approach for UQ is stochastic modeling. To this end, the uncertainty in data is modeled as random variables and random processes. Subsequently, the original deterministic model, for example a system of partial differential equations (PDEs), become a stochastic model, such as a system of stochastic partial differential equations (SPDEs). One of the most important features of the stochastic model is that the corresponding SPDEs now reside in a higher dimensional space which includes the original space/time dimensions and additional random dimensions determined by the nature of the random inputs. It becomes imperative to develop efficient multi-dimensional methods so that solving the problem does not become computationally prohibitive. One of the most widely used methods is based on generalized polynomial chaos (gPC), [19], which is an extension of the classical polynomial chaos (PC) method pioneered by R. Ghanem in [9]. In gPC, stochastic quantities are expressed as convergent orthogonal polynomial series in terms of the input random variables. This closely resembles spectral methods, [10], where fast convergence can be achieved when the solutions are sufficiently smooth. Since its first systematical introduction, gPC has been successfully applied to a large variety of stochastic problems. In many problems the stochastic solutions are smooth in random space and consequently gPC offers highly accurate results. However, when a stochastic solution contains a discontinuity in its corresponding random space, Gibbs oscillations can occur, [11, 13], causing the accuracy of the gPC solution to suffer. To circumvent this difficulty, it is necessary to discretize the random space and employ piecewise continuous polynomials, similar to what is done in classical deterministic numerical analysis. The piecewise continuous gPC approach has been proposed and analyzed in [3, 13, 17]. Although the discretization is straightforward conceptually, it is numerically challenging because when the computational domain, i.e., the random space, is discretized into elements, there is little choice but to use the tensor product rule to construct the elements. Hence the total number of elements grows exponentially fast as the dimensionality of the random space increases. As a result, in practical stochastic computations it is imperative to split the random space into as few elements as possible unless the random dimensionality is low (e.g. less than five). It is therefore extremely useful to identify the location of discontinuities so that one knows exactly where to split the domain into elements.

Based on an extension of the polynomial annihilation edge detection method previously used for deterministic applications, [1], this paper proposes and analyzes a global polynomial basis numerical procedure for determining jump discontinuities in a gPC multivariate stochastic solution of a given stochastic simulation. The method further allows the complete domain classification into smooth sub-regions. Consequently, the random space can be split into elements that avoid the inclusion of a jump discontinuity, and in turn will generate a much more efficient piecewise gPC computation. Other traditional edge detection methods, e.g. [5, 15], most often require (i) uniform grid points and (ii) the construction of gradients and Laplacians. As such they are less flexible and more costly when applied to higher dimensional problems. Finally we note that the polynomial annihilation edge detection method is high order, making it easier to detect jump discontinuities in highly variable functions using

fewer grid point values.

This paper is organized as follows: In Section 2 we establish the general framework of the gPC procedure for solving the stochastic partial differential equations. Section 3 reviews the polynomial annihilation edge detection method, which is extended in Section 4 so that it can be applied to stochastic simulation results in the form of gPC expansions. In Section 5 we propose an adaptive algorithm to completely classify the domain based on the regions of smoothness of the function. Section 6 demonstrates our techniques on some numerical experiments, and we conclude with ideas for future investigations in Section 7. The terms 'edge' and 'jump discontinuity' are used interchangeably throughout.

2 Solution of Stochastic Differential Equations

This section briefly reviews the solution technique for stochastic differential equations with a focus on one of the most widely adopted non-sampling approaches — generalized polynomial chaos (gPC), [19].

2.1 Governing Equation

For notational convenience, the exposition here is restricted to boundary value problems. The framework is nevertheless applicable to general time dependent problems.

Let $D \subset \mathbb{R}^\ell$, $\ell = 1, 2, 3$, be a fixed physical domain with boundary ∂D, and $x = (x^{(1)}, \ldots, x^{(\ell)})$ be the coordinates. We consider a partial differential equation (PDE)

$$\begin{array}{rcl} L\,(x, u; y) & = & 0 \quad \text{in } D, \\ B(x, u; y) & = & 0 \quad \text{on } \partial D, \end{array} \tag{1}$$

where L is a differential operator and B is a boundary operator. The operator B can take various forms on different boundary segments, e.g., $B \triangleq I$ on Dirichlet segments where I is the identity operator, and $B \triangleq n \cdot \nabla$ on Neumann segments whose outward unit normal vector is n. Here $y = (y^{(1)}, \ldots, y^{(d)}) : \Omega \to \mathbb{R}^d, d \geq 1$, are *random variables* defined on a proper probability space $(\Omega, \mathcal{A}, \mathcal{P})$ with event space Ω, σ-algebra $\mathcal{A}$, and probability measure $\mathcal{P}$, that characterize the random inputs from various sources, e.g., boundary condition, system parameters, etc. We assume that these random variables $(y^{(1)}, \ldots, y^{(d)})$ are mutually independent of each other. In other words, any other possible additional random parameters, if they exist, are functions of y.

Let us assume for all $i = 1, \ldots, d$, the random variables $y^{(i)}$ are continuous with probability density functions (PDF) $\rho^{(i)} : \Gamma^{(i)} \to \mathbb{R}^+$, where $\Gamma^{(i)} \triangleq y^{(i)}(\Omega)$ is the image of $y^{(i)}$. Then the random vector $y = (y^{(1)}, \cdots, y^{(d)})$ has support and joint PDF

$$\Gamma \triangleq \prod_{i=1}^{d} \Gamma^{(i)} \subset \mathbb{R}^d, \qquad \rho(y) = \prod_{i=1}^{d} \rho^{(i)}(y^{(i)}), \tag{2}$$

respectively. This allows us to construct numerical formulations in the finite dimensional (d-dimensional) random space Γ in replacement of Ω. Subsequently the governing equation (1) should be valid for all $y \in \Gamma$, and we seek a solution $u(x, y) : \bar{D} \times \Gamma \to \mathbb{R}$ such that (1)

is satisfied for all $x \in \bar{D}$ and $y \in \Gamma$. Although the methodology presented here applies to random variables with unbounded support, we will restrict our exposition to bounded support for notational convenience. Upon proper scaling, the finite dimensional probability space Γ is a hypercube $[-1,1]^d$.

In the following we will adopt the multi-index notation: Let $\mathbf{i} = (i_1, \ldots, i_d) \in \mathbb{N}_0^d$ be a multi-index, with $|\mathbf{i}| = \sum_{k=1}^d i_k$, and $\mathbf{i} = \mathbf{j}$ iff. $i_k = j_k, \forall k = 1, \ldots, d$. We also define an index set

$$\mathcal{J}_N \triangleq \{\mathbf{i} \in \mathbb{N}_0^d : |\mathbf{i}| \leq N\} \tag{3}$$

for a given integer $N \geq 0$.

2.2 Generalized Polynomial Chaos

An Nth-order generalized polynomial chaos (gPC) expansion to the solution of (1) takes the form

$$u_N(x,y) = \sum_{\mathbf{i} \in \mathcal{J}_N} \hat{u}_{\mathbf{i}}(x) \Phi_{\mathbf{i}}(y), \qquad \forall x \in D, \tag{4}$$

where the expansion coefficients

$$\hat{u}_{\mathbf{i}}(x) = \int u(x,y) \Phi_{\mathbf{i}}(y) \rho(y) dy = \mathbb{E}[u(x,y)\Phi_{\mathbf{i}}(y)], \quad \mathbf{i} \in \mathcal{J}_N, \tag{5}$$

and $\Phi_{\mathbf{i}}(y) = \prod_{k=1}^d \phi_{i_k}(y^{(k)})$ are the polynomial basis functions. Here $\phi_{i_k}(y^{(k)})$ are the i_k-th order orthogonal polynomials in the $y^{(k)}$ dimension satisfying

$$\mathbb{E}_k \left[\phi_m(y^{(k)}) \Phi_n(y^{(k)})\right] \triangleq \int \phi_m(y^{(k)}) \phi_n(y^{(k)}) \rho^{(k)}(y^{(k)}) dy^{(k)} = \delta_{mn}, \quad 0 \leq m, n \leq N, \tag{6}$$

where δ_{mn} is the Kronecker delta function and the polynomials are normalized. Therefore $\{\Phi_{\mathbf{i}}(y)\}_{\mathbf{i} \in \mathcal{J}_N}$ are d-variate orthogonal polynomials of total degree up to N constructed as the tensor product of the one dimensional polynomials in each $y^{(k)}, k = 1, \ldots, d$, direction. They are orthonormal so that

$$\mathbb{E}[\Phi_{\mathbf{i}}(y)\Phi_{\mathbf{j}}(y)] \triangleq \int \Phi_{\mathbf{i}}(y)\Phi_{\mathbf{j}}(y)\rho(y)dy = \delta_{\mathbf{ij}}, \quad \forall \mathbf{i}, \mathbf{j} \in \mathcal{J}_N, \tag{7}$$

where $\delta_{\mathbf{ij}} = \prod_{k=1}^d \delta_{i_k j_k}$.

From an implementation point of view, it is convenient to construct a numbering scheme so that all polynomial bases are indexed with only one index. The gPC expansion, (4), can then be written as

$$u_N(x,y) = \sum_{k=1}^K \hat{u}_k(x) \Phi_k(y), \tag{8}$$

where each index k corresponds to a unique sequence $\mathbf{k} = (k_1, \ldots, k_d)$ such that

$$\Phi_k(y) = \phi_{k_1}(y^{(1)}) \cdots \phi_{k_d}(y^{(d)}), \qquad k_1 + \cdots + k_d \leq N, \tag{9}$$

and the orthogonality relation, (7), can be written as

$$\mathbb{E}[\Phi_j(y)\Phi_k(y)] \triangleq \int \Phi_j(y)\Phi_k(y)\rho(y)dy = \delta_{jk}, \quad \forall 1 \le j,k \le K, \tag{10}$$

where K is the total number of basis functions

$$K = \dim\left(\Pi_N^d\right) = \binom{N+d}{d}. \tag{11}$$

Here Π_N^d denotes the linear space of d-variate polynomials of degree up to N. Similar one-to-one correspondence exists between the generalized Fourier coefficients $\{\hat{u}_k\}_{k=1}^K$ and $\{\hat{u}_{\mathbf{k}}\}_{|\mathbf{k}|=0}^N$. Classical approximation theory guarantees that this is the best approximation in Π_N^d, i.e., for any $x \in D$ and $u \in L_\rho^2(\Gamma)$,

$$\|u - u_N\|_{L_\rho^2(\Gamma)} = \inf_{\Psi \in \Pi_N^d} \|u - \Psi\|_{L_\rho^2(\Gamma)}. \tag{12}$$

2.3 Stochastic Galerkin and Collocation Methods

In practice, the expansion coefficients $\{\hat{u}_k\}_{k=1}^K$ in (8) are not available. Two often used approaches to numerically resolve them are the stochastic Galerkin (SG) method and stochastic collocation (SC) method.

The stochastic Galerkin approach seeks an approximate gPC solution in the form of (8) as

$$v_N(x,y) = \sum_{k=1}^K \hat{v}_k(x)\Phi_k(y), \tag{13}$$

where K is given by (11). The expansion coefficients $\{\hat{v}_k\}$ are obtained by satisfying (1) in the weak form; that is for all $k = 1, \ldots, K$,

$$\begin{array}{rcll} \int L(x, v_N; y)\Phi_k(y)\rho(y)dy & = & 0, & \text{in } D, \\ \int B(x, v_N; y)\Phi_k(y)\rho(y)dy & = & 0, & \text{on } \partial D. \end{array} \tag{14}$$

The resulting equations are a set of (usually coupled) *deterministic* PDEs for $\{\hat{v}_k\}$, and standard numerical techniques can be applied.

Another approach is to employ a pseudo-spectral stochastic collocation approach, [18]. We again seek an approximate solution in the form of gPC expansion, similar to (8), i.e., for any $x \in D$,

$$w_N(x,y) = \sum_{k=1}^K \hat{w}_k(x)\Phi_k(y), \tag{15}$$

where the expansion coefficients are determined as

$$\hat{w}_k(x) = \sum_{j=1}^Q u(x, y_j)\Phi_k(y_j)\alpha_j, \qquad k = 1, \ldots, K. \tag{16}$$

Here $\{y_j, \alpha_j\}_{j=1}^{Q}$ are a set of nodes and weights, and $u(x, y_j)$ is the deterministic solution of (1) with fixed y_j. The nodes and weights should be chosen from a cubature rule such that

$$\hat{w}_k(x) \approx \int u(x,y)\Phi_k(y)\rho(y)dy = \hat{u}_k(x) \tag{17}$$

with sufficient accuracy. (Here the last equality follows from (5).) Subsequently (15) becomes an approximation of the exact gPC expansion, (4). The difference between the two is caused by the integration error from (17) and is termed as "aliasing error" in [18], following the similar terminology from the classical deterministic spectral methods. (cf. [6, 10]).

2.4 Summary of gPC Based Methods

In summary, all gPC based methods seek to approximate the stochastic solution of (1) in the form of (4), where the expansion coefficients are obtained approximately via either a Galerkin approach, (13), or a collocation approach, (15). Depending on the probability distribution of the random variables y, different orthogonal polynomials can be employed for better performance, [19]. Whenever the solution is relatively smooth in the random space Γ, the gPC methods based on global orthogonal polynomials exhibits fast convergence and are highly effective.

However, when the stochastic solution contains discontinuities in the random space Γ, the convergence of gPC methods deteriorates, [11, 13]. To circumvent this difficulty, it is important to use a set of piecewise smooth gPC basis functions, (cf. [3, 13, 17]), so that the undesirable effects caused by the discontinuities are locally confined.

While the idea of using piecewise basis functions is simple, as it is a straightforward extension of the classical numerical methods such as the finite element method, a significant difficulty arises in practice for high dimensional random spaces. Whenever the random space is decomposed, the usual approach is to split each direction into elements and construct the elements via tensor product, [3, 13, 17]. This will incur a prohibitively large number of elements for high dimensional random space. Furthermore, since the unknown gPC expansion coefficients must be solved inside each element, the ensuing computational burden can be tremendously high. If, however, the location of the jump discontinuities (if there are any) can be identified from a given gPC simulation result, then the entire random space can be decomposed into as few as possible elements. The following sections are devoted to a strategy based on an extension of the deterministic polynomial annihilation edge detection methodology to enable such a decomposition.

3 Polynomial Annihilation Edge Detection

Let us first consider a piecewise continuous function $f(y) : [-1, 1] \to \mathbb{R}$ in one dimensional (random) space known only on the set of discrete points [1]

$$S = \{y_1, y_2, ..., y_{\bar{Q}^{(1)}}\} \subset [-1, 1]. \tag{18}$$

[1] There are $\bar{Q}^{(1)}$ reconstruction points in one dimension. We note, however, that in higher dimensions, $\bar{Q}^{(d)}$ may not be related to the total number of quadrature points in (16), Q.

Assume that f has well defined one sided limits, $f(y^{\pm})$, at any point y in the domain. We denote by J the set of the points of discontinuity of f, that is, $J = \{\xi : -1 < \xi < 1\}$, where ξ is a jump discontinuity point in f. The local jump function is defined as

$$[f](y) = f\left(y^{+}\right) - f\left(y^{-}\right) = \begin{cases} 0, & \text{if } y \neq \xi, \\ [f](\xi), & \text{if } y = \xi. \end{cases} \tag{19}$$

Hence if f is continuous at y, the jump function $[f](y) = 0$; if y is a point of discontinuity of f, then $[f](y)$ is equal to the jump value there.

The polynomial annihilation edge detection method, introduced in [1], seeks an approximation to $[f](y)$ that converges rapidly to zero away from the jump discontinuities. It can be described as follows: Assume y is a point inside the domain, i.e., $y \in (-1, 1)$. For a given positive integer $m < \bar{Q}^{(1)} - 1$, we choose a local stencil

$$S_y = \{y_j | y_j \in S\} = \{y_0, ..., y_m\} \tag{20}$$

of the nearest $m + 1$ grid points around y. Let $\{p_i\}_{i=0}^{m}$ be a basis for Π_m^1, the univariate polynomial space of degree up to m. The polynomial annihilation edge detection method is given by

$$L_m f(y) = \frac{1}{q_m(y)} \sum_{y_j \in S_y} c_j(y) f(y_j), \tag{21}$$

where the coefficients $c_j(y)$ are chosen to annihilate polynomials of degree up to $m - 1$ and are determined by solving the linear system

$$\sum_{y_j \in S_y} c_j(y) p_i(y_j) = p_i^{(m)}(y), \qquad \forall i = 0, \cdots, m. \tag{22}$$

Here $p_i^{(m)}(y)$ denotes the mth derivative of $p_i(y)$. Notice that the solution to (22) exists and is unique. Next we define

$$S_y^{+} = \{y_j \in S_y | y_j \geq y\} \quad \text{and} \quad S_y^{-} = S_y \setminus S_y^{+}. \tag{23}$$

The normalization factor in (21), given by

$$q_m(y) = \sum_{y_j \in S_y^{+}} c_j(y), \tag{24}$$

ensures that $L_m f(y)$ has correct value at the jump discontinuities. Note that (24) is non-zero by design. In [1] it was shown that if the maximum separation $h(y)$ is defined as

$$h(y) = \max\{|y_i - y_{i-1}| : y_{i-1}, y_i \in S_y\}, \tag{25}$$

then (21) satisfies the following property:

$$L_m f(y) = \begin{cases} [f](\xi) + \mathcal{O}(h(y)), & \text{if } y_{j-1} \leq \xi, y \leq y_j, \\ \mathcal{O}(h^{\min(m,k)}(y)), & \text{if } f \in \mathcal{C}^k(I_y), k > 0. \end{cases} \tag{26}$$

Here I_y is the smallest closed interval such that $S_y \subset I_y$. Hence the polynomial annihilation edge detection method converges to $[f](y)$ with a rate depending on m and the local smoothness of f.

We pause to note two important distinctions between the polynomial annihilation edge detection method, (21), and more traditional edge detection methods, [5, 15]. The polynomial annihilation edge detection method is a high order reconstruction of the *jump function*, $[f](y)$, defined in (19). Other techniques more commonly used in image processing are *edge detectors*. Specifically, they attempt to identify the set of edges J with regard to certain thresholds that are usually determined from some underlying assumptions about the particular image (typically digital) and outside influences (e.g., noise). The high order design of (21) is well suited when the underlying structure of the piecewise smooth function has some variability, which is often the case in the gPC solution. Also, fewer points are needed in each direction to resolve the corresponding jump function, which can dramatically reduce computational costs. The second critical advantage of the polynomial annihilation edge detection method over traditional techniques is that it does *not* require uniform point distribution. This is particularly useful when combined with the gPC collocation method, since using uniformly distributed points is almost never attempted.

The following example demonstrates the performance of the polynomial annihilation edge detection method:

Example 3.1 *Consider the function on* $[-2, 6\pi]$

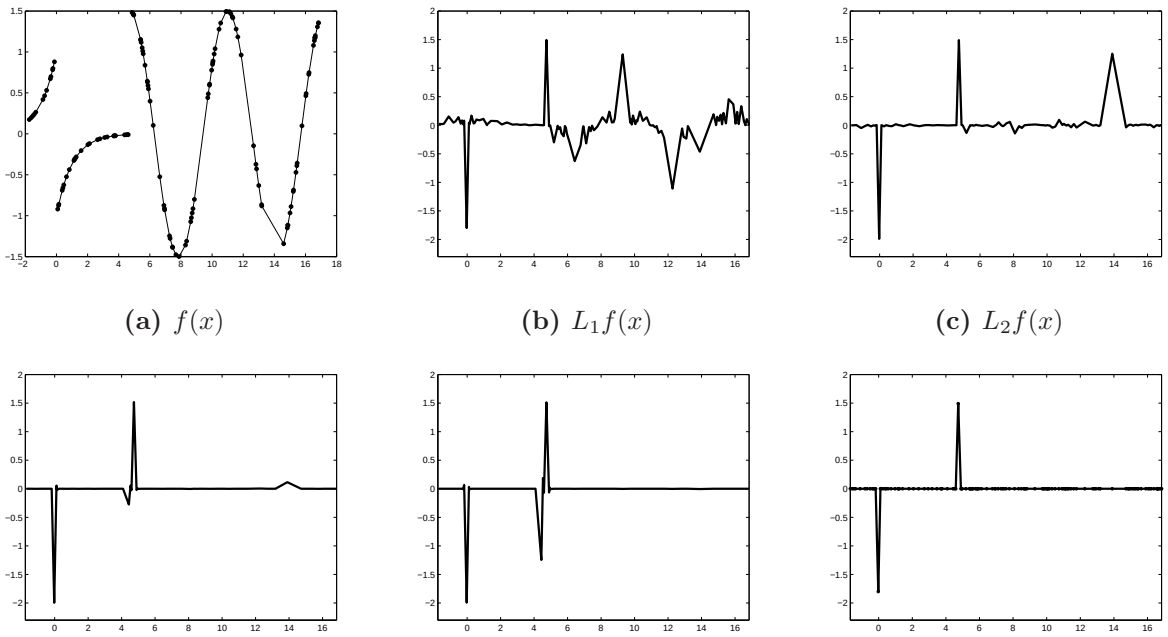
$$f(y) = \begin{cases} e^y, & y < 0, \\ -e^{-y}, & 0 \le y < \frac{3\pi}{2}, \\ -1.5\sin(y), & \frac{3\pi}{2} \le y \le 6\pi. \end{cases} \tag{27}$$

Example 3.1 has jump discontinuities at $y = 0$ and $y = 3\pi/2$ and is continuous everywhere else in the domain. The results of applying (21) to randomly distributed grid point data for various orders of m are shown in Fig. 1(b)-(e). The wide gaps in the data near $y = 9$ and $y = 12$, displayed in Fig. 1(a), can be easily mistaken for jump discontinuities in the underlying function by a low order edge detector. On the other hand, the higher order polynomial annihilation edge detection method is able to distinguish gaps in data from true discontinuities.

Fig. 1 demonstrates the importance of the order m in (21). Small order m might cause misidentification of a steep gradient as an edge (due to low resolution). On the other hand, the inherent nature of high order polynomial approximation causes oscillations to occur in the vicinity of the discontinuities when m is large. To prevent inaccuracies due to either reason, we apply the *minmod* function, which is typically used to reduce oscillations in flux limiters when solving numerical conservation laws.[2] Specifically, we apply

$$MM(L_m f(y)) = \begin{cases} \min_{m\in\mathcal{M}} L_m f(y), & \text{if } L_m f(y) > 0, \quad \forall m \in \mathcal{M}, \\ \max_{m\in\mathcal{M}} L_m f(y), & \text{if } L_m f(y) < 0, \quad \forall m \in \mathcal{M}, \\ 0, & \text{otherwise}, \end{cases} \tag{28}$$

[2]The *minmod* technique was introduced in the context of edge detection in [4, 8] and was used in [1] to enhance the performance of the polynomial annihilation method.



(a) $f(x)$ **(b)** $L_1 f(x)$ **(c)** $L_2 f(x)$

(d) $L_4 f(x)$ **(e)** $L_5 f(x)$ **(f)** $MM(L_m f(x)), m \in \{1, \cdots, 5\}$

Figure 1: Example 3.1 (a) underlying function on $N = 128$ randomly distributed grid points; (b)-(e) Jump function approximation of Example 3.1 using various orders m; (f) Minmod results, (28).

where $\mathcal{M} \subset \mathbb{N}$ of positive integers. As shown in Fig. 1(f) for $\mathcal{M} = \{1, 2, 3, 4, 5\}$, the minmod function controls the oscillations while still maintaining high order of convergence away from the jump discontinuities. It furthermore reduces the need for outside thresholding.

Finally, we would like to point out the the polynomial annihilation edge detection method was originally developed in [1] to determine multi-dimensional jump discontinuities from randomly distributed grid points. However, the gPC collocation method uses structured grids in each dimension (typically Gaussian). Hence to simplify programming and increase computational efficiency, we choose to use a dimension by dimension approach. As an additional advantage, the method easily lends itself to parallel processing.

4 Determining Jump Discontinuities in a GPC Stochastic Solution

The gPC solution to a stochastic partial differential equation is in the form of a polynomial series, (4), which is known either via its expansion coefficients, obtained by the stochastic Galerkin (13) or collocation (15) method, or via the stochastic collocation solution at the given nodes in the corresponding (multivariate) random space. The polynomial annihilation edge detection method is well suited to either situation.

4.1 Formulation in One Dimension

Let us first consider the one dimensional random space case, i.e., where $f(y) : [-1, 1] \to \mathbb{R}$ is a piecewise smooth function and $\{\phi_k(y)\}_{k=0}^N$ are the gPC basis functions of degree up to N. In this case, the total number of basis functions in (11) is $K = N + 1$. For example, when y has a beta distribution in $[-1, 1]$, the corresponding gPC basis functions are the Jacobi polynomials, see [19]. Consider a truncated gPC approximation for $f(y)$:

$$f_N(y) = \sum_{k=0}^{N} \hat{f}_k \phi_k(y), \tag{29}$$

where $\hat{f}_k$ is obtained either via stochastic Galerkin or collocation, (13) or (15) respectively. We assume that the number of coefficients in the stochastic Galerkin method or the number of quadrature points in the stochastic collocation procedure is sufficient to preserve any discontinuities in $f(y)$. The jump function $[f](y)$ can be approximated via (21) using (29) as

$$L_m f(y) \approx L_m f_N(y) = \frac{1}{q_m(y)} \sum_{y_j \in S_y} c_j(y) f_N(y_j) = \frac{1}{q_m(y)} \sum_{y_j \in S_y} c_j(y) \sum_{k=0}^{N} \hat{f}_k \phi_k(y_j), \tag{30}$$

where the coefficients $c_j(y)$ are determined from (22) and S_y is the stencil defined in (20) with $m < N - 1$. For simplicity, we assume in one dimension that the number of reconstruction points is the same as the degree of the polynomial expansion, that is, $\bar{Q}^{(1)} = N$ in (18).

Another possible approach is to approximate the jump function by directly using the gPC expansion coefficients. This is particularly useful when the gPC solution is obtained by the stochastic Galerkin method. To this end, we first note that the system that determines the annihilation coefficients, (22), must hold for any polynomials of degree up to m. By making the annihilation polynomials the same as the gPC basis polynomials, $p_i(y) = \phi_i(y)$, (22) becomes

$$\sum_{y_j \in S_y} c_j(y) \phi_i(y_j) = \phi_i^{(m)}(y), \quad i = 0, \cdots, m. \tag{31}$$

where again $\phi_i^{(m)}(y)$ denotes the mth derivative of $\phi_i(y)$. Maintaining such consistency between the basis corresponding to the given coefficients and the basis to be annihilated helps in avoiding Gibbs oscillations inherent upon function reconstruction, (29). We now rearrange (30) so that

$$L_m f_N(y) = \frac{1}{q_m(y)} \sum_{k=0}^{N} \hat{f}_k \sum_{y_j \in S_y} c_j(y) \phi_k(y_j). \tag{32}$$

Further, since the first $m - 1$ polynomials are annihilated by (31), (32) reduces to

$$L_m f_N(y) = \frac{1}{q_m(y)} \sum_{k=m}^{N} \hat{f}_k \sum_{y_j \in S_y} c_j(y) \phi_k(y_j). \tag{33}$$

In this way, an actual reconstruction of the function is completely avoided, yielding both a savings in computational costs as well as a potential improvement in accuracy since no

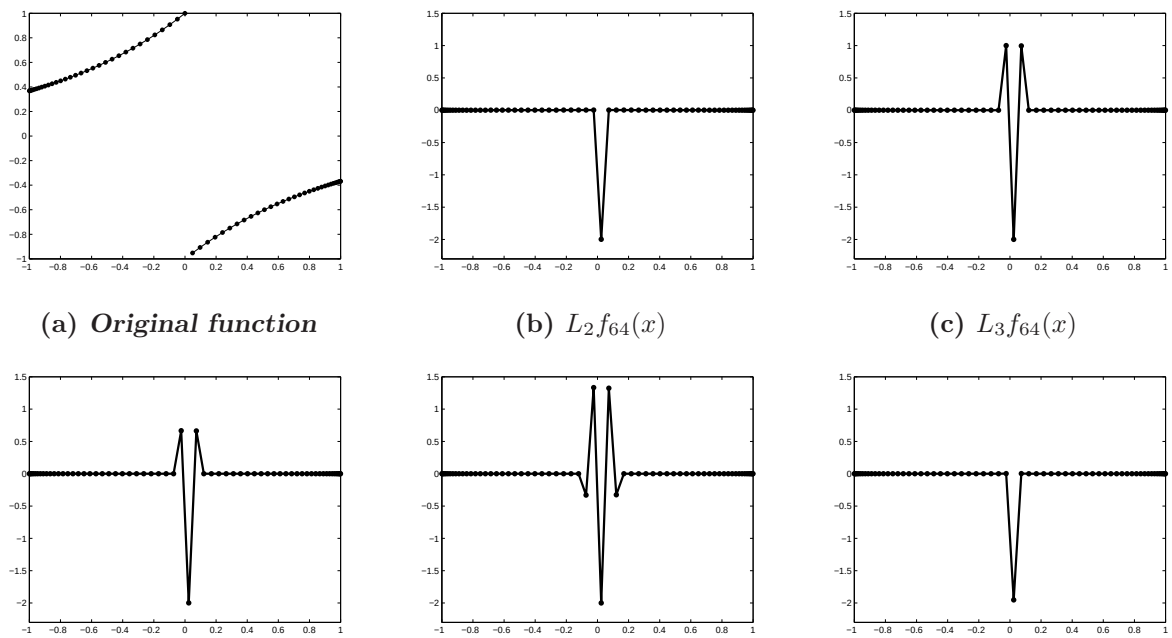
interpolatory error is introduced. Note that (33) is quite intuitive as the smoothness of a function is determined by its high end coefficients in a spectral partial sum.

To demonstrate use of (33), we consider the following example:

Example 4.1 *Let* $f(y) : [-1, 1] \to \mathbb{R}$ *be defined as*

$$f(y) = \begin{cases} e^{y}, & y < 0, \\ -e^{-y}, & y \geq 0. \end{cases} \tag{34}$$

Example 4.1 has a jump discontinuity at $y = 0$. Figures 2(b)-(e) illustrate the results using (33) for orders $m = 2$ through $m = 5$. Figure 2(f) displays the jump function approximation after the minmod algorithm, (28), is applied for $\mathcal{M} = \{1, 2, 3, 4, 5\}$.



(a) ***Original function*** **(b)** $L_2 f_{64}(x)$ **(c)** $L_3 f_{64}(x)$

(d) $L_4 f_{64}(x)$ **(e)** $L_5 f_{64}(x)$ **(f)** $MM(L_m f_{64}(x))$

Figure 2: Example 4.1. (a) underlying function on $N = 64$ Legendre Gauss points; (b)-(e) Jump function approximation using various orders m; (f) Minmod results, (28), for $\mathcal{M} = \{1, 2, 3, 4, 5\}$.

4.2 Extension to Higher Dimensions

Suppose now that $f(y) : [-1, 1]^d \to \mathbb{R}, d > 1$, is a piecewise smooth function with corresponding N^{th}-order gPC approximation

$$f_N(y) = \sum_{\mathbf{i} \in \mathcal{J}_N} \hat{f}_{\mathbf{i}} \Phi_{\mathbf{i}}(y). \tag{35}$$

Here the index set $\mathcal{J}_N$ is defined in (3) and the expansion coefficients $\hat{f}_{\mathbf{i}}$ are obtained either via a stochastic Galerkin or a stochastic collocation method.

As stated previously, it is possible to perform edge detection on the function data at any given values $f_N(y)$. However, the effects of high dimensional interpolation errors are unclear on the polynomial annihilation edge detection method. Further, as described in Section 4.1, we can take advantage of the formulation in (35) and apply the polynomial annihilation edge detection method directly on the coefficients in each dimension. This allows us to avoid interpolatory errors while also improving the overall computational efficiency.

Let $b = (b^{(1)}, \ldots, b^{(d)})$ be a fixed coordinate in $[-1, 1]^d$ and

$$y^{[\ell]}(b) = (b^{(1)}, \ldots, b^{(\ell-1)}, y^{(\ell)}, b^{(\ell+1)}, \ldots, b^{(d)}) \tag{36}$$

be a one dimensional coordinate in $[-1, 1]^d$ where all but the ℓth-dimension are fixed by the given coordinate b. Then (35) becomes

$$f_N(y^{[\ell]}(b)) = \sum_{\mathbf{i} \in \mathcal{J}_N} \hat{f}_{\mathbf{i}} \Phi_{\mathbf{i}}(y^{[\ell]}(b)). \tag{37}$$

The polynomial annihilation edge detection method, (21), in the $y^{(\ell)}$ direction is then

$$\begin{aligned} L_m f_N(y^{(\ell)}) &= \frac{1}{q_m(y^{(\ell)})} \sum_{y_j^{[\ell]}(b) \in S^{[\ell]}(b)} c_j(y^{(\ell)}) f_N(y_j^{[\ell]}(b)) \\ &= \frac{1}{q_m(y^{(\ell)})} \sum_{y_j^{[\ell]}(b) \in S^{[\ell]}(b)} c_j(y^{(\ell)}) \sum_{\mathbf{i} \in \mathcal{J}_N} \hat{f}_{\mathbf{i}} \Phi_{\mathbf{i}}(y_j^{[\ell]}(b)) \end{aligned} \tag{38}$$

where $S^{[\ell]}(b)$ consists of the closest $m+1$ points in the $y^{(\ell)}$ direction around $y^{[\ell]}(b)$,

$$S^{[\ell]}(b) = \{y_0^{[\ell]}(b), \cdots, y_m^{[\ell]}(b)\}. \tag{39}$$

The coefficients $c_j(y^{(\ell)})$ are obtained by annihilating one dimensional polynomials of degree up to $m-1$ in the ℓth direction, specifically, by solving the linear system

$$\sum_{y_j^{[\ell]}(b) \in S^{[\ell]}(b)} c_j(y^{(\ell)}) \phi_{k_\ell}(y_j^{[\ell)]}(b)) = \phi_{k_\ell}^{(m)}(y^{(l)}), \quad k_\ell = 0, \cdots, m, \tag{40}$$

where $\phi_{k_\ell}^{(m)}(y^{(l)})$ denotes the mth derivative of the polynomial $\phi_{k_\ell}(y^{(l)})$. Rearranging (38) therefore yields

$$L_m f_N(y^{(\ell)}) = \frac{1}{q_m(y^{(\ell)})} \sum_{\mathbf{i} \in \mathcal{J}_m^{(\ell)}} \hat{f}_{\mathbf{i}} \sum_{y_j^{(\ell)} \in S^{[\ell]}(b)} c_j(y^{(\ell)}) \Phi_{\mathbf{i}}(y_j^{[\ell]}(b)), \tag{41}$$

where

$$\mathcal{J}_m^{(\ell)} = \{\mathbf{i} \in \mathbb{N}_0^d : \mathbf{i} \in \mathcal{J}_N \text{ and } i_\ell \geq m\} \tag{42}$$

is an index set obtained by removing the ℓth multi-index components less than m from the index set $\mathcal{J}_N$. Hence we can save computational costs and reduce potential interpolation error by avoiding explicit function reconstruction. Numerical examples are provided in Section 6.

5 Domain Classification

Based on the techniques developed in Section 4, we can now identify the grid cells in which the jump discontinuities lie in each dimension, given either gPC expansion coefficients, (41), or the stochastic solution at the collocation nodes, (38). Here we demonstrate how the polynomial annihilation edge detection method can be used to completely classify the computational domain in terms of its continuous sub-domains. As explained in the introduction, this will enable more computational efficiency in any subsequent simulation.

5.1 General Domain Classification

For ease of presentation, we explain our domain classification algorithm on a two dimensional domain that has only two smooth sub-domains. The algorithm can be extended to multiple sub-domains and is limited only by the resolution of the grid. Figure 3 shows a jump function on part of a bounded two dimensional domain, $[-1,1] \times [-1,1]$. The jump function approximation, (41), is followed by the minmod algorithm, (28), and is reconstructed on $\bar{Q}^{(1)} \times \bar{Q}^{(2)}$ points. For simplicity, and without loss of generality, we will from now on assume that $\bar{Q}^{(1)} = \bar{Q}^{(2)} = \cdots = \bar{Q}^{(d)} = \bar{Q}$ represent both the number of reconstruction points and the number of quadrature points in each dimension. The jump function in Figure 3 runs across the domain, dividing it into two mutually exclusive regions, which we identify as belonging to either Class 1 or Class 2. Each point in the domain is tagged with its respective class value, 1 or 2. Our objective is to build an algorithm that automates this domain classification. We note that since the edge detection method is applied dimension by dimension, the jump discontinuity values themselves are of correspondingly different signs and magnitudes. This bears no consequence on the domain classification, however.

We assume that the problem is well resolved so that the jump discontinuity lies in the interior of the grid and each region is well connected. Initially, all the points in the domain

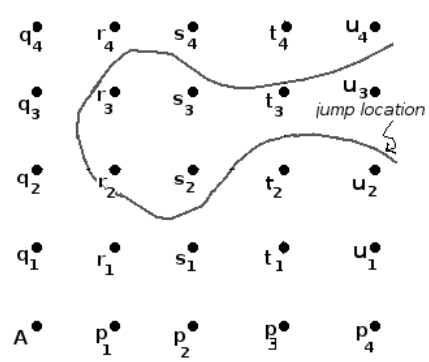


Figure 3: A jump function running through a two dimensional domain.

are listed as unclassified with a class value equal to zero. Broadly speaking, as we move on with the algorithm, each point gets identified as belonging to either Class 1 with a class value of 1, or Class 2 with a class value of 2. Let us begin with the situation where no point has been classified: We pick a first point from the list of all unclassified points, say the point A in

Figure 3, and assign it a class value 1. We consider all the points to the right of A, i.e., the points along the straight line $\bar{Ap_4}$, consisting of the group of points $\{p_1, p_2, p_3, p_4\}$. Clearly there are no edges anywhere along this line. Hence all of the points are classified as Class 1 points, and each point is assigned a class value of 1. Next we consider the group of points along the straight line $\bar{Aq_4}$, i.e., the group $\{q_1, q_2, q_3, q_4\}$. Once again every point this group gets classified as Class 1 points. Nine of a total of twenty five points have been classified thus far. The algorithm proceeds by acting on each one of these new eight points in a similar manner as for the point A. For instance, considering the vertical direction of the point p_1 would determine that points (A, p_2, p_3, p_4, r_1) all belong to the first class. Naturally there will be points that are classified more than once, but they will be classified to the same class. Therefore this algorithm needs only be concerned with the first classification.

The process continues until all the points in Class 1 have been covered; this is marked by the fact our search stops yielding new points that have not already been classified. At this juncture, we start looking for the next connected region - Class 2. The first point in the set of yet unclassified points (class value 0) is marked as belonging to Class 2 by assigning class value 2 to it. The same search procedure is followed as for Class 1 and all the points belonging to Class 2 are collected. After Class 2 has also been exhausted, we check to see if there are any more unclassified points. If there are, we define Class 3 with class value 3 and continue the same procedure. If there no points left with class value 0 (as will be for the case exhibited in Figure 3), it means all the grid points in the domain have been classified and our domain classification is complete.

The domain classification described here and the adaptive domain classification described in the next section are both by design favorable to parallel processing. High performance computing would therefore allow large scale problems to be analyzed efficiently.

5.2 Adaptive Domain Classification

Refining the entire multi-dimensional random space for more accurate edge detection and improved domain classification is unrealistic. However, any cell found to contain a jump discontinuity can be further refined to subsequently produce a new set of cells from which we can again determine jump discontinuities and classify the (sub-)domain into smooth regions.[3] If each refinement increases the resolution by $\bar{Q}^d$, it is possible to build an adaptive procedure that obtains very high accuracy both in locating the jump discontinuities and in classifying the domain after only a few iterations. The following algorithm describes how the gPC and polynomial annihilation methods can be used to characterize the solution space of the governing equations up to domain accuracy, as determined by the maximum separation distance given in (25), at each level up to an arbitrary number of iterations, $\mu \in \mathbb{N}$. In some sense, our adaptive domain classification can be viewed as an $h - p$ adaptive scheme, where h is the maximum separation and p represents the order of the polynomial annihilation edge detection method. All p refinements are pre-determined by the underlying resolution.

Algorithm 5.1 *(*Adaptive Domain Classification*) Consider the initial domain, the hypercube* $[-1, 1]^d$*. Specify the maximum number of iterations,* μ*, the number of reconstruction*

[3]We recall that the polynomial annihilation edge detection method locates jump discontinuities to within a single cell.

(quadrature) points in each direction for the stochastic collocation gPC method, $\bar{Q}$, and the maximum order of the polynomial edge detection method, m, where $m < \bar{Q} - 1$.

initialize *the only identified hyper-cube as the whole domain, $[-1, 1]^d$.*

for $i = 1, \cdots, \mu$

for *each identified hyper-cube*

1. *Apply the stochastic collocation gPC method with $\bar{Q}$ quadrature (reconstruction) points in each direction.*
2. *Determine the cells, each comprised of 2^d points, containing jump discontinuities using the polynomial annihilation edge detection, (42), combined with the minmod algorithm, (28), on the gPC approximation, (35).*
3. *Classify the domain as described in Section 5.1.*
4. *Element cells needing further refinement are identified as those in which not all of the 2^d points have the same class value. After these cells are refined using $\bar{Q}^d$ points, Steps 1 through 3 are repeated. Note that the classification of the 2^d points making up the (coarser) cell is passed to any subsequent domain classification.*

end

end

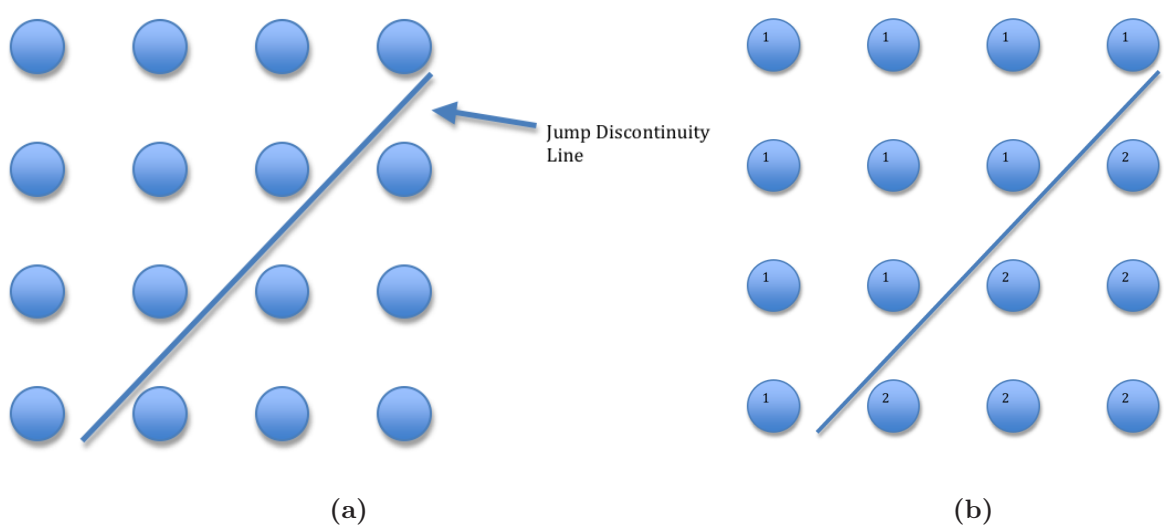


Figure 4: (a) Initial set-up for a two dimensional problem using $\bar{Q} = 4$ uniform quadrature points in each direction. The jump discontinuity splits the domain into two parts. (b) Initial domain classification results.

Figures 4 and 5 provide a schematic for Algorithm 5.1 for a simple two dimensional example. Figure 4(a) displays the initial set-up where there are $\bar{Q} = 4$ uniform quadrature

points in each direction. In this example, the initial one dimensional maximum separation distance in (25) is (constant) $h_1(y) = h_1 = \frac{2}{\bar{Q}-1}$, and each subsequent level of refinement is $h_i = \frac{2}{(\bar{Q}-1)^i}$, $i = 1, \cdots, \mu$.[4] Suppose that the gPC solution and subsequent processing by the edge detection method produces a line discontinuity as shown in Figure 4(a) (Steps 1 and 2 in Algorithm 5.1.) The initial domain classification (Step 3) is illustrated in Figure 4(b), where it is apparent that when a cell contains a discontinuity, not all of its $2^d = 4$ points will have the same class value. Such cells are identified as needing further refinement (Step 4) and are depicted in Figure 5(a).

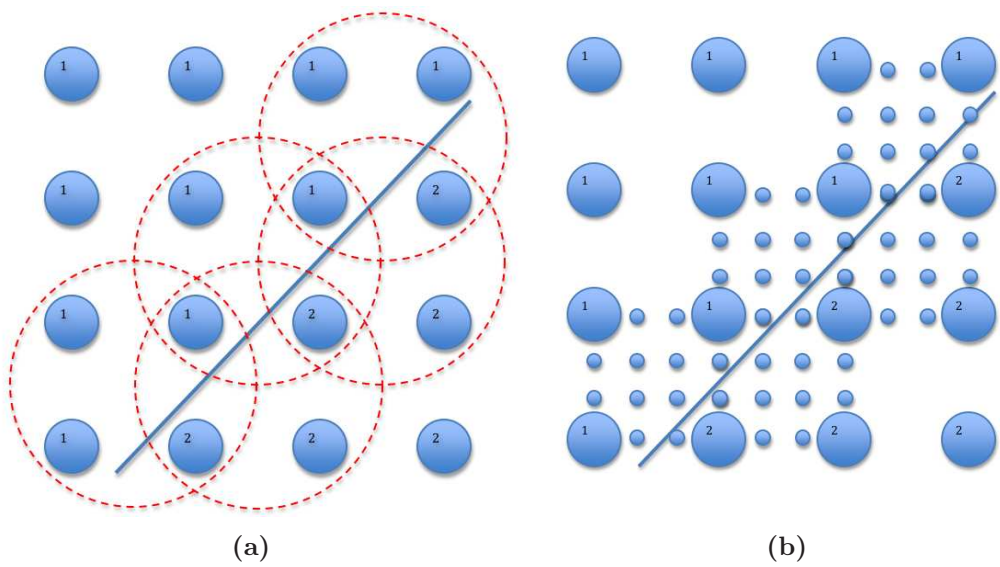


Figure 5: (a) The dotted circles contain cells that require further refinement (Step 4 of Algorithm 5.1). (b) Cell refinement using $\bar{Q} = 4$ uniform quadrature points in each direction. Note that the coarse grid point classification is passed on to the refined grid.

At this stage the first iteration of Algorithm 5.1 is completed. The convergence of the polynomial annihilation edge detection method, (26), suggests that any cell not identified as needing further refinement has a jump function value of order $\mathcal{O}(h_1^m)$.[5] In general, the accuracy of Algorithm 5.1 in classifying smooth regions at each level of refinement depends upon the maximum separation distance of the particular quadrature scheme chosen, the specified order of the polynomial annihilation edge detection method, m, and the maximum number of iterations allowed, μ.

As demonstrated in Figure 5(b), the algorithm may begin again by refining the five cells containing discontinuities. All previous coarse grid point domain classifications are passed on to the finer grid. Repeating Steps 1 through 3 will further isolate the jump discontinuity line to cells that have dimensional length $h_2 = \frac{2}{(\bar{Q}-1)^2}$. Algorithm 5.1 is iterated until either no

[4]The uniform points are for ease of presentation only. Typically the gPC solution would use a Gaussian type distribution. We also adopt the one dimensional notation for the maximum separation distance $h(y) = h(y^{[\ell]}(b))$ in (36) since all points are uniformly distributed.

[5]This will be worse as the jump discontinuity is approached, [1, 2]. The approximation also assumes that $f \in \mathcal{C}^m(\Omega)$ in smooth regions.

new cells needing refinement are identified or the maximum number of allowable iterations, μ, has expired. The algorithm produces various cells which are hyper-cube elements with dimensional lengths of $h_i = \frac{2}{(Q-1)^i}$, $i = 1, \ldots, \mu$. Each of these hyper-cubes will either have an accurate gPC expansion, or will be identified as having a non-smooth domain. Any hyper-cube that has a non-smooth domain will have a (general) dimensional length $h_\mu(y)$. In this sense we can say that the domain can be resolved by this algorithm up to an arbitrary accuracy, $h_\mu(y)$.

6 Numerical Examples

6.1 Domain Decomposition

The following example demonstrates use of the adaptive domain classification, Algorithm 5.1:

Example 6.1 *Let* $f(y^{(1)}, y^{(2)}) : \Omega \longrightarrow R, \Omega = [-1, 1] \times [-1, 1]$ *be defined as*

$$f(y) = \begin{cases} -1, & \text{if } y^{(1)} < y^{(2)}, \\ 1, & \text{else.} \end{cases} \tag{43}$$

Example 6.1 is continuous everywhere in the domain except along the line $y^{(1)} + y^{(2)} = 0$. Figure 6 shows the results after applying Algorithm 5.1. In this example we used $\bar{Q} = 5$ Legendre Gauss Lobatto collocation points, defined as $\{(y^{(1)}, y^{(2)}) : y^{(1)}, y^{(2)} = \pm 1, \pm\frac{1}{2}, 0\}$. We also chose $m = 2$ and $\mu = 3$. Note that the density of elements increases near the jump discontinuity line to provide increased accuracy where needed, while maintaining a sparse representation in smooth regions. The smallest elements have length governed by the maximum separation distance of the Legendre Gauss Lobatto quadrature points.

6.2 Multidimensional Application

Now we consider a stochastic differential-algebraic system of equations

$$\begin{aligned} \frac{du}{dt} &= \frac{\alpha_1}{1 + v^\beta} - u, \\ \frac{dv}{dt} &= \frac{\alpha_2}{1 + w^\gamma} - v, \\ w &= \frac{u}{(1 + [\text{IPTG}]/K)^\eta}, \end{aligned} \tag{44}$$

where α_1, α_2, β, γ, η, and K are parameters and [IPTG] is the system input. This model is for a genetic toggle switch in *Escherichia coli*, [7]. The parameters $p = (p_1, \ldots, p_6) = (\alpha_1, \alpha_2, \beta, \gamma, \eta, K)$ are modeled as random variables in the form of $p = \langle p \rangle (1 + \sigma y)$, where $\langle p \rangle = (156.25, 15.6, 2.5, 1, 2.0015, 2.9618 \times 10^{-5})$ are the mean values and $y = (y^{(1)}, \ldots, y^{(6)})$ are random variables uniformly distributed in $[-1, 1]^6$.

The system will reach steady state, which exhibits a switch property, i.e., "on" and "off", depending on the intensity of the input [IPTG]. Stochastic simulation of the system was

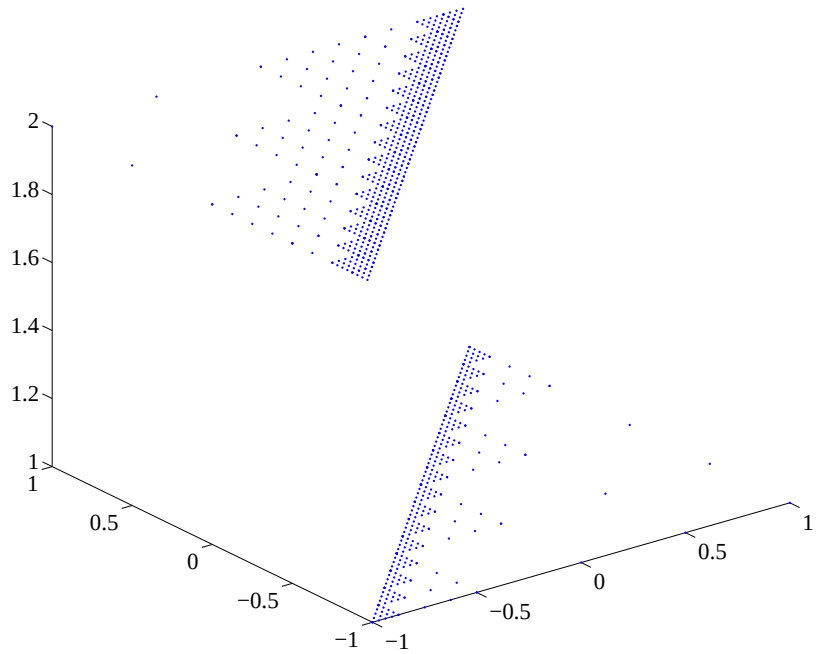


Figure 6: Algorithm 5.1 applied to Example 6.1 on the Legendre Gauss Lobatto grid.

carried out in [18] by the stochastic collocation method. The solution v in ((44)), termed as the "Normalized GFP expression" of simulation and experimental measurement, is displayed as error bars in Figure 7. We observe good agreement with the experimental measurements. (Details of the computation can be found in [18].)

The switch behavior suggests that the solution, e.g., v, depends on the parameters in a non-smooth manner. However, it is not clear either from the physics or the mathematical point of view which parameters are associated with the jump behavior. Since it is known that variations in the parameters β and γ do not effect the switching property we carry out the multi-dimensional edge detection method for $v(y), y \in [-1, 1]^4$, in $d = 4$ dimensional space to determine the location of the jump.

The results of the adaptive domain classification for the gPC solution for the stochastic differential-algebraic system, (44), is shown in Figure 8(a), Figure 8(c), and Figure 8(e) for the entire range of α_1, α_2, η, where $K = 2.9618 \times 10^{-5} + \{-\sigma, 0, \sigma\}$. Upon inspection, the adaptive domain classification indicates that the edge between the two classes can be approximated by a hyperplane. Support Vector Machines (SVM) provide powerful classification paradigms developed over the last decade in machine learning theory that operate by classifying the binary data that separate the hyperplane, or decision surface, which maximizes the margin between the two classes in the training data, [16]. Using the results of the adaptive domain classification as the training data for a linear SVM, the hyper-plane which maximizes the margin between the two identified classes is

$$f(\alpha_1, \alpha_2, \eta, K) = -0.1188\alpha_1 + 1.5849\alpha_2 + 18.5336\eta - 8.7464 \times 10^5 K - 18.49. \tag{45}$$

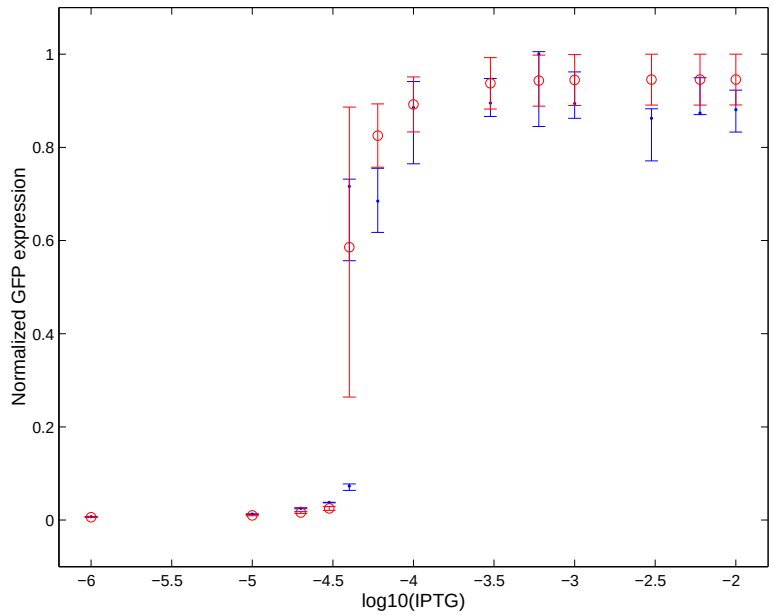


Figure 7: Stochastic steady-state solution of v in (44). The light (red) error bars centered around the open circles display the numerical results determined by the gPC. The dark (blue) error bars centered around the dots depict the corresponding experimental measurements.

This hyperplane, although not a complete characterization of the multi-dimensional edge, provides a simple rule to determine the state of the genetic toggle switch model given in (44). Specifically, if a given set of parameters satisfy $f(\alpha_1, \alpha_2, \eta, K) > 0$, then these parameters lie in the domain where the steady-state solution is "on". Conversely, if a given set of parameters yield $f(\alpha_1, \alpha_2, \eta, K) < 0$, then these parameters lie in the domain where the steady-state solution is "off". We note that (45) is a crude approximation of the edge that yields a relatively accurate characterization of (44). In fact, comparison of the classification given by (45) with the steady state solution of (44) for a thousand randomly generated realization of the parameters α_1, α_2, η and K produces an agreement rate of 99.1%. Our experiments generated steady state solutions $.2786 < v_{steady} < 12.7095$. Hence we chose $v_{steady} > 6.4941$ to classify the system as being in the "on" domain. Otherwise it was considered "off". While the hyperplane test provides high "on" and "off"agreement, Figure 8 demonstrates that the adaptive domain classification represents a more comprehensive description of the system. More sophisticated approximations can be generated using nonlinear kernels in the SVM. However, for this particular example, the hyperplane in (45) provides a simple intuitive representation without compromising accuracy.

7 Conclusion

In this paper we proposed a numerical strategy for detecting jump discontinuities from stochastic simulation results obtained in the form of global polynomial expansions in multi-dimensional random spaces. The strategy is an extension of the deterministic polynomial annihilation edge detection method, [1], and works particularly well with the high order stochastic Galerkin or

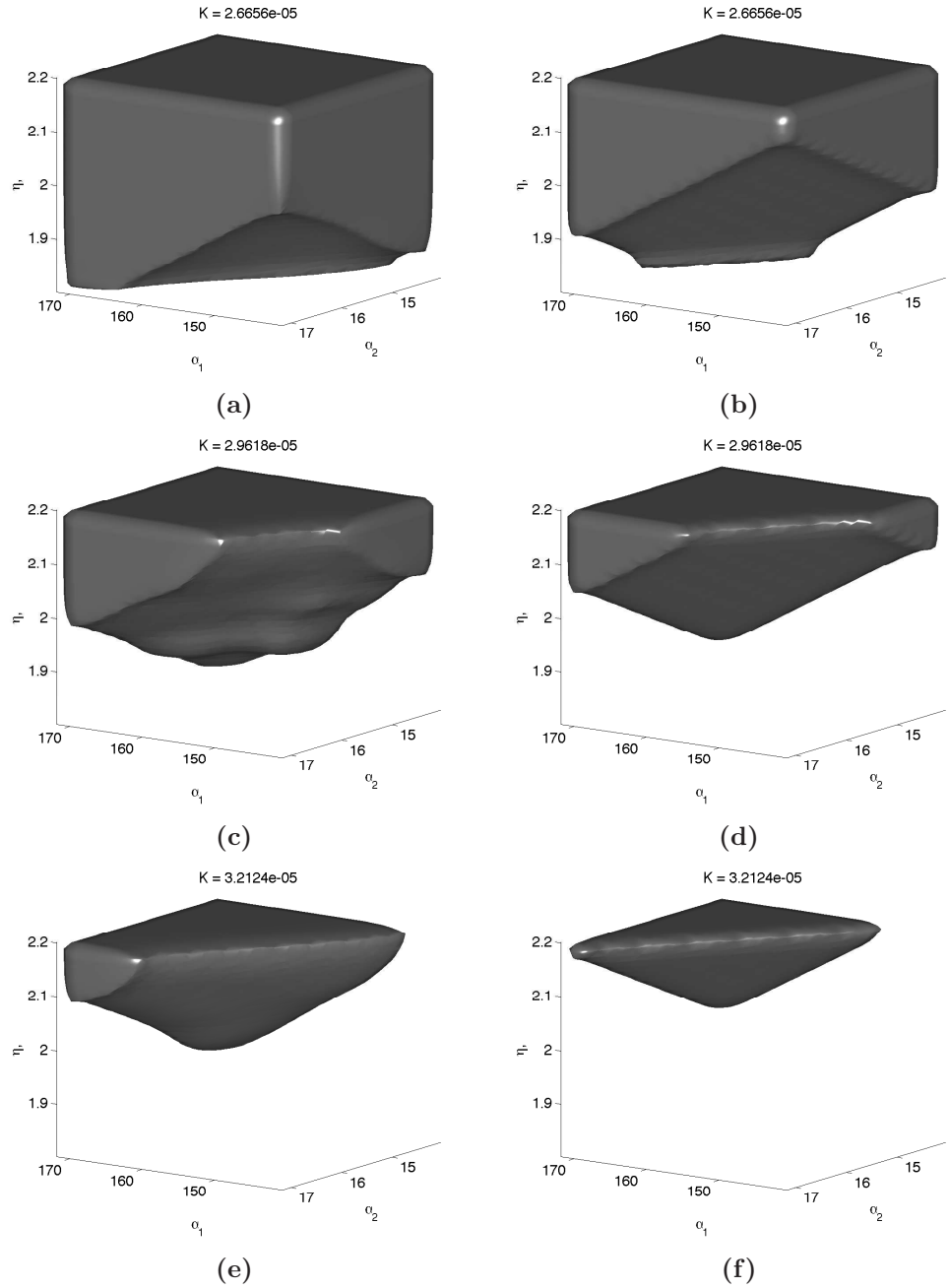


Figure 8: Visualization of the adaptive domain classification for the stochastic differential-algebraic system (44) for the fixed parameters $\beta = 2.5$, $\gamma = 1$, $\sigma = 0.1$, and $[IPTG] = 4.0 \times 10^{-5}$. The gPC solution has two identifiable domain classes, visualized here by fixing the dimension for the parameter K and displaying the remaining three parameters for (a) $K = 2.9618 \times 10^{-5} - \sigma$, (c) $K = 2.9618 \times 10^{-5}$, and (e) $K = 2.9618 \times 10^{-5} + \sigma$. The gray portion of the cube represents the "on" intensity domain and the void portion of the cube represents the "off." The figures in (b), (d), and (f) display the two classes when the boundary between the two domains is approximated by a hyperplane.

stochastic collocation methods based on generalized polynomial chaos (gPC). The algorithms and implementation were presented, with an emphasis on their applicability in high dimensional spaces. Upon correct identification of the discontinuities (if they exist), we are able to classify the problem domain in terms of its regions of smoothness. We further derived an adaptive procedure that refines the locations of the discontinuities and subsequently improves the domain classification for a reasonable computational cost. Hence we can decompose the entire random space into as few smooth sub-domains as possible and thus facilitate more accurate stochastic simulations using a multi-element approach.

We emphasize that the proposed strategy represents only one of the first attempts to identify the existence and location of discontinuities in stochastic simulations. Here we focus on the applicability and potential of the method and leave many more issues, for example, more efficient algorithms for very high dimensional random space, to further study.

Acknowledgments

The submitted manuscript has been authored by contractors [UT-Battelle, manager of Oak Ridge National Laboratory (ORNL)] of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Anne Gelb was partially supported by NSF DMS-0510813, NSF FRG DMS-0652833, NSF RUI-0608844 and NSF SCREMS DMS-0421846. Dongbin Xiu was partially supported by NSF CAREER DMS-0645035.

References

[1] R. Archibald, A. Gelb, and J. Yoon, *Polynomial Fitting for Edge Detection in Irregularly Sampled Signals and Images,* SIAM J. Numer. Anal., 43(1), pp 259-279, 2005.

[2] R. Archibald, A. Gelb, and J. Yoon, *Determining the Locations and Discontinuities in the Derivatives of Functions,* Appl. Numer. Math., available online Feb 13, 2007.

[3] I. Babuška and R. Tempone and G.E. Zouraris, *Galerkin Finite Element Approximations of Stochastic Elliptic Differential Equations,* SIAM J. Numer. Anal., 42, pp 800-825, 2004.

[4] R. Bauer, *Band Pass Filters for Determining Shock Locations,* Ph.D. thesis, Applied Mathematics, Brown University, 1995.

[5] J. Canny, *A Computational Approach to Edge Detection,* IEEE Trans. Pattern Anal. Machine Intell., 8, pp 679-698, 1986.

[6] C. Canuto, M. Hussaini, A. Quarteroni and T. Zang, *Spectral Methods in Fluid Dynamics,* published by Springer-Verlag, 1988.

[7] T. Gardner, C. Cantor and J. Collins, *Construction of a Genetic Toggle Switch in Escherichia coli*, Nature, 403, pp 339-342, 2000.

[8] A. Gelb and E. Tadmor, *Adaptive Edge Detectors for Piecewise Smooth Data Based on the Minmod Limiter,* J. Sci. Comput., 28(2-3), pp 279-306, 2006.

[9] R. Ghanem and P. Spanos, *Stochastic Finite Elements: a Spectral Approach,* published by Springer-Verlag, 1991.

[10] D. Gottlieb and S. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications,* published by CBMS-NSF, SIAM, Philadelphia, PA, 1977.

[11] D. Gottlieb and D. Xiu, *Galerkin method for Wave Equations with Uncertain Coefficients,* Comm. Comput. Phys., 3(2), pp 505-518, 2008.

[12] J. Hesthaven, S. Gottlieb and D. Gottlieb, *Spectral Methods For Time-Dependent Problems,* Cambridge University Press, 2007.

[13] O. Le Maitre, O. Knio, H. Najm, and R. Ghanem, *Uncertainty Propagation Using Wiener-Haar Expansions,* J. Comput. Phys., 197, pp 28-57, 2004.

[14] O. Le Maitre, O. Knio, H. Najm, and R. Ghanem, *Multi-Resolution Analysis of Wiener-type Uncertainty Propagation Schemes,* J. Comput. Phys., 197, pp 502-531, 2004.

[15] I. Sobel, *An Isotropic* 3 *Image Gradient Operator,* Machine Vision for Three-Dimensional Scenes (H. Freeman editor), Academic Press, Boston, 1990.

[16] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[17] X. Wan and G. Karniadakis, *Multi-Element Generalized Polynomial Chaos for Arbitrary Probability Measures,* SIAM J. Sci. Comput., 28, pp 901-928, 2006.

[18] D. Xiu, *Efficient Collocational Approach for Parametric Uncertainty Analysis*, Comm. Comput. Phys., 2(2), pp 293-309, 2007.

[19] D. Xiu and G. Karniadakis, *The Wiener-Askey Polynomial Chaos for Stochastic Differential Equations,* SIAM J. Sci. Comput., 24(2), pp 619-644, 2002.