Journal of Computational Physics 228 (2009) 2676–2689

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Discontinuity detection in multivariate space for stochastic simulations

Rick Archibald [a,*], Anne Gelb [b], Rishu Saxena [b], Dongbin Xiu [c]

[a] Computer Science and Mathematics, Oak Ridge National Laboratory, Oak Ridge, TN 37831, United States
[b] Department of Mathematics and Statistics, Arizona State University, Box 1804, Tempe, AZ 85287, United States
[c] Department of Mathematics, Purdue University, West Lafayette, IN 47907, United States

ARTICLE INFO

Article history:
Received 6 June 2008
Received in revised form 18 December 2008
Accepted 1 January 2009
Available online 13 January 2009

Keywords:
Stochastic partial differential equations
Multivariate edge detection
Generalized polynomial chaos method

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.

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 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

* Corresponding author.
E-mail addresses: ArchibaldRK@ORNL.gov (R. Archibald), ag@math.asu.edu (A. Gelb), saxena@mathpost.asu.edu (R. Saxena), dxiu@math.purdue.edu (D. Xiu).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.01.001

become computationally prohibitive. One of the most widely used methods is based on generalized polynomial chaos (gPC) [23], which is an extension of the classical polynomial chaos (PC) method pioneered by Ghanem in [10]. In gPC, stochastic quantities are expressed as convergent orthogonal polynomial series in terms of the input random variables. This closely resembles spectral methods [11,6,13], 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, see, for example [3,9,10,14–16,19,20,22,23]. 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 [12,14], 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 [3,14,19]. 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. 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. To the best of our knowledge, there have thus far been no attempts to address the issue of detecting discontinuities in stochastic simulations, or on how to decompose the random space in the event that discontinuities exist. Such is the focus of this paper.

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. [5,17], 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. Furthermore, as is demonstrated in the current investigation and is relevant to the gPC solution, the polynomial annihilation edge detection method can be reformulated in terms of the coefficients of any (orthogonal) polynomial expansion. 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 2.2 reviews the polynomial annihilation edge detection method, which is extended in Section 3 so that it can be applied to stochastic simulation results in the form of gPC expansions. In Section 4 we propose an adaptive algorithm to completely classify the domain based on the regions of smoothness of the function. Section 5 demonstrates our techniques on some numerical experiments, and we conclude with ideas for future investigations in Section 6. The terms 'edge' and 'jump discontinuity' are used interchangeably throughout.

2. Problem background and formulation

This section provides brief reviews of the generalized polynomial chaos method for stochastic simulations [23], and the polynomial annihilation method for edge detection [1]. We then formulate the problem under investigation in this paper.

2.1. Generalized polynomial chaos for stochastic equations

Generalized polynomial chaos (gPC) has become one of the most widely used methods for solutions of stochastic problems, particularly in the context of uncertainty quantification. Here we review the basic ideas behind gPC.

Let $D \subset \mathbb{R}^\ell, \ell = 1,2,3$, be a physical domain with boundary ∂D and coordinates $x = (x^{(1)}, \ldots, x^{(\ell)})$ and let $T > 0$ be a real number. We consider the following general (scalar) stochastic partial differential equation

$$\begin{cases} u_t(x,t,y) = \mathcal{L}(u), & D \times (0,T] \times \mathbb{R}^d, \\ \mathcal{B}(u) = 0, & \partial D \times [0,T] \times \mathbb{R}^d, \\ u = u_0, & D \times \{t = 0\} \times \mathbb{R}^d, \end{cases} \tag{1}$$

where $\mathcal{L}$ is a (nonlinear) differential operator, $\mathcal{B}$ is the boundary condition operator, u_0 is the initial condition, and $y = (y^{(1)}, \ldots, y^{(d)}) \in \mathbb{R}^d$, $d \geqslant 1$, are a set of independent random variables characterizing the random inputs to the governing equation from various sources, e.g. boundary condition, system parameters, etc.

The solution is therefore a stochastic quantity,

$$u(x,t,Z) : \overline{D} \times [0,T] \times \mathbb{R}^d \to \mathbb{R}. \tag{2}$$

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 $y = (y^{(1)}, \ldots, y^{(d)})$ has a joint PDF

$$\rho(y) = \prod_{i=1}^{d} \rho^{(i)}(y^{(i)}). \tag{3}$$

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

$$\mathcal{J}_N \triangleq \{\mathbf{i} \in \mathbb{N}_0^d : |\mathbf{i}| \leqslant N\} \tag{4}$$

for a given integer $N \geqslant 0$.

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

$$u_N(t,x,y) = \sum_{\mathbf{i} \in \mathcal{J}_N} \hat{u}_{\mathbf{i}}(t,x)\Phi_{\mathbf{i}}(y), \tag{5}$$

where

$$\hat{u}_{\mathbf{i}}(t,x) = \int u(t,x,y)\Phi_{\mathbf{i}}(y)\rho(y)dy = \mathbb{E}[u(x,y)\Phi_{\mathbf{i}}(y)], \quad \forall \mathbf{i} \in \mathcal{J}_N, \tag{6}$$

are the expansion coefficients, $\mathbb{E}$ is the expectation operator, and

$$\Phi_{\mathbf{i}}(y) = \prod_{k=1}^{d} \phi_{i_k}(y^{(k)})$$

are the polynomial basis functions, defined as products of univariate orthogonal polynomials in the $y^{(k)}$ dimension, $k = 1, \ldots, d$, satisfying

$$\int \phi_m(y^{(k)})\phi_n(y^{(k)})\rho^{(k)}(y^{(k)})dy^{(k)} = \delta_{mn}, \quad 0 \leqslant m, n \leqslant N. \tag{7}$$

Here 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 satisfying

$$\mathbb{E}[\Phi_{\mathbf{i}}(y)\Phi_{\mathbf{j}}(y)] = \int \Phi_{\mathbf{i}}(y)\Phi_{\mathbf{j}}(y)\rho(y)dy = \delta_{\mathbf{ij}}, \quad \forall \mathbf{i}, \mathbf{j} \in \mathcal{J}_N, \tag{8}$$

where $\delta_{\mathbf{ij}} = \prod_{k=1}^{d} \delta_{i_k j_k}$. From (8), the distribution of Z_k will determine the polynomial type. For example, Hermite polynomials are associated with the Gaussian distribution, Jacobi polynomials with the beta distribution, Laguerre polynomials with the gamma distribution, etc. For a detailed discussion of these correspondences and their resulting computational efficiency, see [23].

Following classical approximation theory, the gPC expansion (5) converges when u is square integrable with respect to $\rho(y)$, that is, for any fixed $(t,x) \in [0,T] \times \overline{D}$,

$$\|u - u_N\|_{L^2_\rho}^2 \triangleq \int (u - u_N)^2 \rho(y)dy \to 0, \quad N \to \infty. \tag{9}$$

In practice, however, the expansion coefficients defined in (6) are not available, as the equation requires knowledge of the (unknown) solution u. Numerical strategies are needed to approximate the coefficients from (6). Two popular approaches are stochastic Galerkin (SG) method and stochastic collocation (SC) method. In SG, the coefficients in (5) are obtained by satisfying the governing Eq. (1) in a weak form in the random space; whereas in SC, (1) is satisfied in a strong form at a set of prescribed nodes in the random space. For detailed discussions on various aspects of gPC based SG and SC methods, see [21].

When the stochastic solution u is relatively smooth with respect to y, the convergence rate of the gPC expansion (5) can be large. This implies that a relatively low-degree expansion can achieve high accuracy and is advantageous in practical stochastic simulations.

However, when the stochastic solution contains discontinuities in the random space, the convergence of gPC methods deteriorates [12,14]. To circumvent this difficulty, it is important to use a set of piecewise smooth gPC basis functions, (cf. [14,19]), 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 [3,14,19]. 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.

2.2. Deterministic 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]

$$S = \{y_1, y_2, \ldots, y_{\overline{Q}^{(1)}}\} \subset [-1,1]. \tag{10}$$

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

$$[f](y) = f(y^+) - f(y^-) = \begin{cases} 0, & \text{if } y \neq \xi, \\ [f](\xi), & \text{if } y = \xi. \end{cases} \tag{11}$$

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 $m < \overline{Q}^{(1)} - 1$, we choose a local stencil

$$S_y = \{y_j | y_j \in S\} = \{y_0, \ldots, y_m\} \tag{12}$$

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

$$L_m f(y) = \frac{1}{q_m(y)} \sum_{y_j \in S_y} c_j(y) f(y_j), \tag{13}$$

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

$$\sum_{y_j \in S_y} c_j(y) p_i(y_j) = p_i^{(m)}(y), \quad \forall i = 0, \ldots, m. \tag{14}$$

Here $p_i^{(m)}(y)$ denotes the mth derivative of $p_i(y)$. Notice that the solution to (14) exists and is unique. Next we define

$$S_y^+ = \{y_j \in S_y | y_j \geqslant y\} \quad \text{and} \quad S_y^- = S_y \setminus S_y^+. \tag{15}$$

The normalization factor in (13), given by

$$q_m(y) = \sum_{y_j \in S_y^+} c_j(y), \tag{16}$$

ensures that $L_m f(y)$ has correct value at the jump discontinuities. Note that (16) is non-zero by design. In [1] it was shown that if the maximum separation $h(y)$ is defined as

$$h(y) = \max\{|y_i - y_{i-1}| : y_{i-1}, y_i \in S_y\}, \tag{17}$$

then (13) satisfies the following property:

$$L_m f(y) = \begin{cases} [f](\xi) + \mathcal{O}(h(y)), & \text{if } y_{j-1} \leqslant \xi,\ y \leqslant y_j, \\ \mathcal{O}(h^{\min(m,k)}(y)), & \text{if } f \in \mathcal{C}^k(I_y),\ k > 0. \end{cases} \tag{18}$$

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 (13), and more traditional edge detection methods [5,17]. The polynomial annihilation edge detection method is a high order reconstruction of the *jump function*, $[f](y)$, defined in (11). 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 (13) 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.

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

Furthermore, as is highlighted in this investigation, the polynomial annihilation edge detection method can be applied directly to orthogonal polynomial expansion coefficients.

The order m in (13) affects the resolution of the jump function both close to and away from any discontinuities. Small m might cause a steep gradient to be misidentified 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. These oscillations may also be misinterpreted as edges. To prevent inaccuracies due to either reason, the *minmod* function, which is typically utilized in flux limiting methods to reduce oscillations when solving numerical conservation laws, was used in [1] to enhance the performance of the polynomial annihilation method.[2] Specifically, we apply

$$MM(L_m f(y)) = \begin{cases} \min\limits_{m\in\mathcal{M}} L_m f(y), & \text{if } L_m f(y) > 0,\ \forall m \in \mathcal{M}, \\ \max\limits_{m\in\mathcal{M}} L_m f(y), & \text{if } L_m f(y) < 0,\ \forall m \in \mathcal{M}, \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

where $\mathcal{M} \subset \mathbb{N}$ of positive integers. 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. Examples of its effectiveness can be found in [1].

Finally, we would like to point out the the polynomial annihilation edge detection method was originally developed 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.

2.3. Problem formulation

The objective of this paper is to discuss an efficient methodology for discontinuity detection for stochastic solutions in term of the gPC expansion. More precisely, let

$$f_N(y) = \sum_{\mathbf{i}\in\mathcal{J}_N} \hat{f}_{\mathbf{i}} \Phi_{\mathbf{i}}(y), \quad y \in \mathbb{R}^d, \quad d \geqslant 1, \tag{20}$$

be the N-th degree gPC approximation to an unknown stochastic solution $f(y)$, where the expansion coefficients $\{\hat{f}\}$ are obtained via either a stochastic Galerkin or a stochastic collocation method. Assuming f_N is an approximation to f with sufficient accuracy, our goal is to determine numerically, given $\{\hat{f}_{\mathbf{i}}\}$: (i) if discontinuities exist in the random space; and (ii) if discontinuities do exist, what are their locations and how can the random space be decomposed into piecewise continuous subdomains.

For the method to work effectively on the gPC expansion, we require that: (i) it should employ the coefficients $\{hatf_{\mathbf{i}}\}$, which are obtained by potentially costly stochastic simulations, as the only inputs and not incur more computations of the original stochastic governing equations; and (ii) it should scale well in high dimension random spaces where $d \gg 1$.

In the following section we present a discontinuity detection method based on applying the polynomial annihilation method to the gPC expansion of the stochastic solution. The key difference, compared to the standard deterministic polynomial annihilation, is that the new method is applied to the gPC expansion coefficients directly to achieve the important requirements listed above.

3. Discontinuity detection for gpc stochastic solution

Hereafter we restrict our discussions to bounded random space, and with proper scaling, consider random space of a hypercube, i.e., $y \in [-1,1]^d, d \geqslant 1$. We first discuss the method in one-dimensional random space with $d = 1$, and then extend it to multi-dimensional space $d > 1$.

3.1. Formulation in one dimension

For one dimensional random space, the multi-index $\mathbf{i}$ becomes a single index and the gPC expansion (20) can be written as

$$f_N(y) = \sum_{k=0}^{N} \hat{f}_k \phi_k(y), \tag{21}$$

where $\{\phi_k\}_{k=0}^N$ is a univariate gPC basis satisfying (7). The jump function $[f](y)$ can be approximated by $[f_N](y)$ as

$$L_m f(y) \approx L_m f_N(y) = \frac{1}{q_m(y)} \sum_{y_j \in S_y} c_j(y) f_N(y_j) = \frac{1}{q_m(y)} \sum_{y_j \in S_y} c_j(y) \sum_{k=0}^{N} \hat{f}_k \phi_k(y_j), \tag{22}$$

[2] The *minmod* technique was introduced in the context of edge detection in bauer [4,8].

where the coefficients $c_j(y)$ can be determined from (14) and S_y is the stencil defined in (12) 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, $\overline{Q}^{(1)} = N$ in (10).

Here we approximate the jump function by directly using the gPC expansion coefficients.

To this end, we first note that the system that determines the annihilation coefficients (14), must hold for any polynomials of degree up to m. By making the annihilation polynomials the same as the gPC basis polynomials, $p_i(y) = \phi_i(y)$, (14) becomes

$$\sum_{y_j \in S_y} c_j(y)\phi_i(y_j) = \phi_i^{(m)}(y), \quad i = 0, \ldots, m, \tag{23}$$

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, (21). We now rearrange (22) so that

$$L_m f_N(y) = \frac{1}{q_m(y)} \sum_{k=0}^{N} \hat{f}_k \sum_{y_j \in S_y} c_j(y)\phi_k(y_j). \tag{24}$$

Further, since the first $m-1$ polynomials are annihilated by (23), (24) reduces to

$$L_m f_N(y) = \frac{1}{q_m(y)} \sum_{k=m}^{N} \hat{f}_k \sum_{y_j \in S_y} c_j(y)\phi_k(y_j). \tag{25}$$

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 (25) 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 (25), we consider the following example:

Example 3.1. Let $f(y) : [-1,1] \to \mathbb{R}$ be defined as

$$f(y) = \begin{cases} \cos 3\pi y, & y < 0, \\ \frac{2}{1+3e^{-5(2y-1)}} - 1 & y \geqslant 0. \end{cases} \tag{26}$$

Example 3.1 has a jump discontinuity at $y = 0$. Fig. 1(a) depicts the function (3.1) and the gPC expansion (20) using Legendre polynomial basis functions with $N = 32$. Fig. 1(b)–(e) illustrate the results using (25) for orders $m = 2$ through $m = 5$. Fig. 1(f) displays the jump function approximation after the minmod algorithm (19), is applied for $\mathcal{M} = \{1,2,3,4,5\}$.

3.2. Extension to higher dimensions

Suppose now that $f(y) : [-1,1]^d \to \mathbb{R}, d > 1$, is a piecewise smooth function with corresponding Nth-order gPC approximation (20). It is now important to 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

$$y^{[\ell]}(b) = (b^{(1)}, \ldots, b^{(\ell-1)}, y^{(\ell)}, b^{(\ell+1)}, \ldots, b^{(d)}) \tag{27}$$

be a one dimensional coordinate in $[-1,1]^d$ where all but the ℓth-dimension are fixed by the given coordinate b. Then (5) becomes

$$f_N(y^{[\ell]}(b)) = \sum_{\mathbf{i} \in \mathcal{J}_N} \hat{f}_{\mathbf{i}} \Phi_{\mathbf{i}}(y^{[\ell]}(b)). \tag{28}$$

The polynomial annihilation edge detection method, (13), in the $y^{(\ell)}$ direction is then

$$L_m f_N(y^{(\ell)}) = \frac{1}{q_m(y^{(\ell)})} \sum_{y_j^{[\ell]}(b) \in S^{[\ell]}(b)} c_j(y^{(\ell)}) f_N(y_j^{[\ell]}(b)) = \frac{1}{q_m(y^{(\ell)})} \sum_{y_j^{[\ell]}(b) \in S^{[\ell]}(b)} c_j(y^{(\ell)}) \sum_{\mathbf{i} \in \mathcal{J}_N} \hat{f}_{\mathbf{i}} \Phi_{\mathbf{i}}(y_j^{[\ell]}(b)) \tag{29}$$

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

$$S^{[\ell]}(b) = \{y_0^{[\ell]}(b), \ldots, y_m^{[\ell]}(b)\}. \tag{30}$$

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

$$\sum_{y_j^{[\ell]}(b) \in S^{[\ell]}(b)} c_j(y^{(\ell)}) \phi_{k_\ell}(y_j^{[\ell]}(b)) = \phi_{k_\ell}^{(m)}(y^{(l)}), \quad k_\ell = 0, \ldots, m, \tag{31}$$

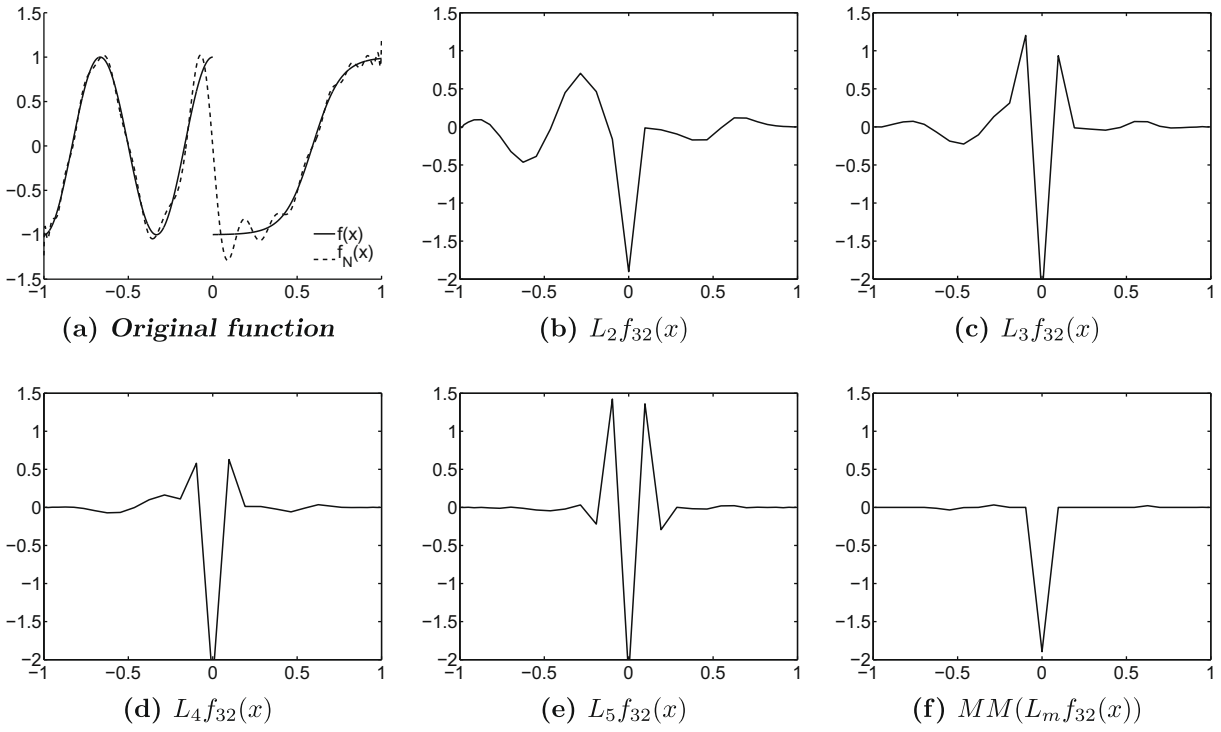

Fig. 1. Example 3.1. (a) underlying function on $N = 32$ Legendre Gauss points; (b)-(e) Jump function approximation using various orders m; (f) Minmod results (19), for $\mathcal{M} = \{1, 2, 3, 4, 5\}$.

where $\phi_{k_\ell}^{(m)}(y^{(l)})$ denotes the mth derivative of the polynomial $\phi_{k_\ell}(y^{(l)})$. Rearranging (29) therefore yields

$$L_m f_N(y^{(\ell)}) = \frac{1}{q_m(y^{(\ell)})} \sum_{\mathbf{i} \in \mathcal{J}_m^{(\ell)}} \hat{f}_{\mathbf{i}} \sum_{y_j^{(\ell)} \in S^{[\ell]}(b)} c_j(y^{(\ell)}) \Phi_{\mathbf{i}}(y_j^{[\ell]}(b)), \tag{32}$$

where

$$\mathcal{J}_m^{(\ell)} = \{\mathbf{i} \in \mathbb{N}_0^d : \mathbf{i} \in \mathcal{J}_N \text{ and } i_\ell \geqslant m\} \tag{33}$$

is an index set obtained by removing the ℓth multi-index components less than m from the index set $\mathcal{J}_N$.

3.3. Remarks

The key ingredient of the proposed discontinuity detection method is the direct annihilation of the gPC basis polynomials. Hence the only inputs required are the gPC expansion coefficients. Note that all of the function evaluations of (32) are in fact simple polynomial evaluations of the gPC expansion (20) at the given points in the point set $S^{[\ell]}$. Therefore they do not involve solutions of the original stochastic equations which can be time consuming. Compared to the computational cost of solving large-scale stochastic problems in high dimensional random spaces, the cost of the discontinuity detection algorithm can be negligible and regarded as a "post-processing" procedure. If a discontinuity is detected by the proposed method, then the gPC computation using a global polynomial basis needs to be refined because of the accuracy lost due to the discontinuity. The most natural way of refinement is to employ piecewise continuous gPC bases. To this end, it is highly desirable to decompose the computational domain into as few subdomains as possible, with interfaces preferably being the discontinuity. In the following section we discuss a domain classification strategy to address this.

4. Domain classification

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.

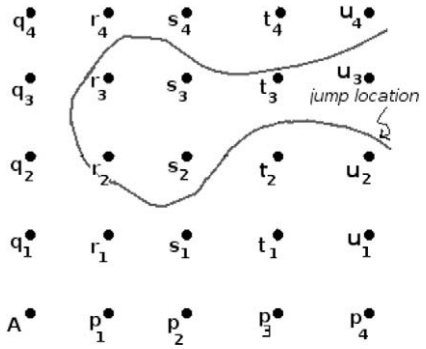

Fig. 2. A jump function running through a two dimensional domain.

4.1. 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. Fig. 2 shows a jump function on part of a bounded two dimensional domain, $[-1,1]\times[-1,1]$. The jump function approximation, (32), is followed by the minmod algorithm (19), and is reconstructed on $\overline{Q}^{(1)}\times\overline{Q}^{(2)}$ points. For simplicity, and without loss of generality, we will from now on assume that $\overline{Q}^{(1)}=\overline{Q}^{(2)}=\cdots=\overline{Q}^{(d)}=\overline{Q}$ represent both the number of reconstruction points and the number of quadrature points in each dimension. The jump function in Fig. 2 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 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 Fig. 2, 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 $\overline{Ap_4}$, 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 $\overline{Aq_4}$, 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 Fig. 2), 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.

4.2. 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 $\overline{Q}^d$, it is possible to build an adaptive procedure that obtains very high

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

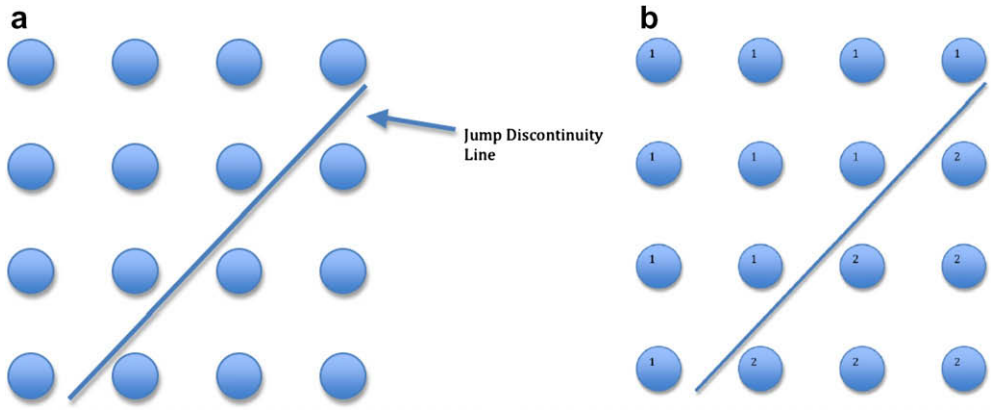

Fig. 3. (a) Initial set-up for a two dimensional problem using $\overline{Q} = 4$ uniform quadrature points in each direction. The jump discontinuity splits the domain into two parts. (b) Initial domain classification results.

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 (17), at each level up to an arbitrary number of iterations, $\mu \in \mathbf{N}$. 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 4.1. (Adaptive Domain Classification) Consider the initial domain, the hyper-cube $[-1, 1]^d$. Specify the maximum number of iterations, μ, the number of reconstruction (quadrature) points in each direction for the stochastic collocation gPC method, $\overline{Q}$, and the maximum order of the polynomial edge detection method, m, where $m < \overline{Q} - 1$.

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

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

for each identified hyper-cube

1. Apply the stochastic collocation gPC method with $\overline{Q}$ 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 (33), combined with the minmod algorithm, (19), on the gPC approximation, (28).
3. Classify the domain as described in Section 4.1.
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 $\overline{Q}^d$ 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

Figs. 3 and 4 provide a schematic for Algorithm 4.1 for a simple two dimensional example. Fig. 3(a) displays the initial set-up where there are $\overline{Q} = 4$ uniform quadrature points in each direction. In this example, the initial one dimensional maximum separation distance in (17) is (constant) $h_1(y) = h_1 = \frac{2}{\overline{Q}-1}$, and each subsequent level of refinement is $h_i = \frac{2}{(\overline{Q}-1)^i}$, $i = 1, \ldots, \mu$.[4] Suppose that the gPC solution and subsequent processing by the edge detection method produces a line discontinuity as shown in Fig. 3(a) (Steps 1 and 2 in Algorithm 4.1.) The initial domain classification (Step 3) is illustrated in Fig. 3(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 Fig. 4(a).

At this stage the first iteration of Algorithm 4.1 is completed. The convergence of the polynomial annihilation edge detection method (18), 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 4.1 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.

[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 (27) 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.

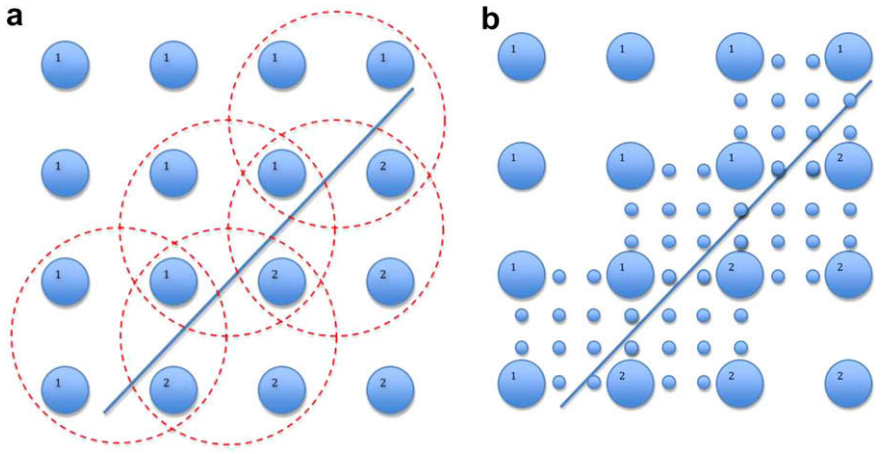

Fig. 4. (a) The dotted circles contain cells that require further refinement (Step 4 of Algorithm 4.1). (b) Cell refinement using $\overline{Q} = 4$ uniform quadrature points in each direction. Note that the coarse grid point classification is passed on to the refined grid.

As demonstrated in Fig. 4(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 $h_2 = \frac{2}{(\overline{Q}-1)^2}$. Algorithm 4.1 is iterated until either no 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 $h_i = \frac{2}{(\overline{Q}-1)^i}$, $i = 1, \ldots, \mu$. 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, $h_\mu(y)$.

4.3. Remarks on the use of tensor grids

The proposed domain classification algorithm relies on examinations of the function values on tensor grids. This facilitates the classification procedure by allowing dimension-by-dimension operations. One obvious drawback of using tensor grids is the rapid growth of the total number of points in high dimensional space, which prevents its use for simulations in dimension more than ~ 5. Our domain classification procedure reduces the severity of this problem, however. Because the function evaluations are conducted for the gPC approximation f_N (20), they do not involve simulations of the original system and only require polynomial evaluations at the grid. Therefore even though the total number points may become very large in high dimensions, their function evaluations can still be conducted relatively fast.

For further refinement of the subdomain interface, as illustrated in Fig. 4, we need function values on a fine set of nodes in regions close to the discontinuity. In this case, using f_N from the coarser grid to determine the jump locations (and subsequent domain decomposition) may not be satisfactory because of the Gibbs phenomenon. It is thus desirable to use the "true function values of f" at the finer nodes, or in our case, a more refined approximation of the original stochastic problem on the tensor grids. This would require additional computations, but only at the nodes close to the discontinuity and not in the entire domain. Therefore the computational cost will not be as prohibitive as performing stochastic simulations on tensor grids in the entire domain.

In summary, the use of tensor grids for domain classification suffers much less from the curse-of-dimensionality than it usually does. However, it is still desirable to develop a domain classification method utilizing non-tensor type grids. This will be pursued in future study.

5. Numerical examples

5.1. Domain decomposition

The following example demonstrates use of the adaptive domain classification, Algorithm 4.1:

Example 5.1. Let $f(y^{(1)}, y^{(2)}) : \Omega \to R, \Omega = [-1, 1] \times [-1, 1]$ be defined as

$$f(y) = \begin{cases} -1, & \text{if } y^{(1)} < y^{(2)}, \\ 1, & \text{else.} \end{cases} \tag{34}$$

Example 5.1 is continuous everywhere in the domain except along the line $y^{(1)} + y^{(2)} = 0$. Fig. 5 shows the results after applying Algorithm 4.1. In this example we used $\overline{Q} = 5$ Legendre Gauss Lobatto collocation points, defined as $\{(y^{(1)}, y^{(2)}) :$

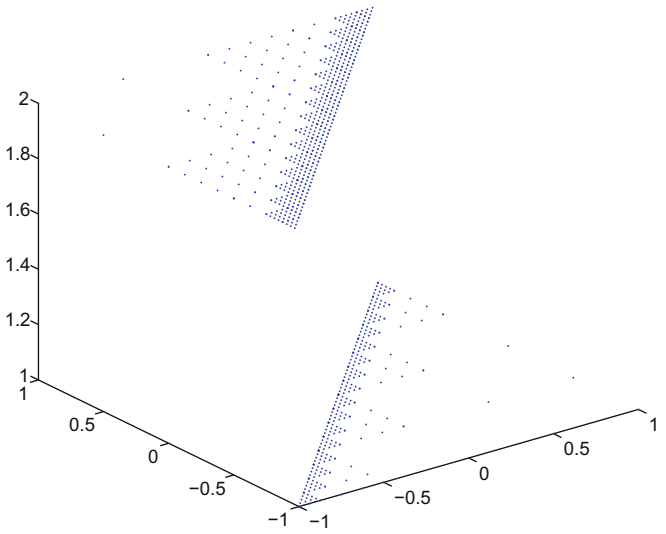

Fig. 5. Algorithm 4.1 applied to Example 5.1 on the Legendre Gauss Lobatto grid.

$y^{(1)}, y^{(2)} = \pm 1, \pm\frac{1}{2}, 0\}$. 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.

5.2. Multidimensional application

We now consider a stochastic differential-algebraic system of equations

$$\begin{aligned} \frac{du}{dt} &= \frac{\alpha_1}{1+v^\beta} - u, \\ \frac{dv}{dt} &= \frac{\alpha_2}{1+w^\gamma} - v, \\ w &= \frac{u}{(1+[\mathrm{IPTG}]/K)^\eta}, \end{aligned} \tag{35}$$

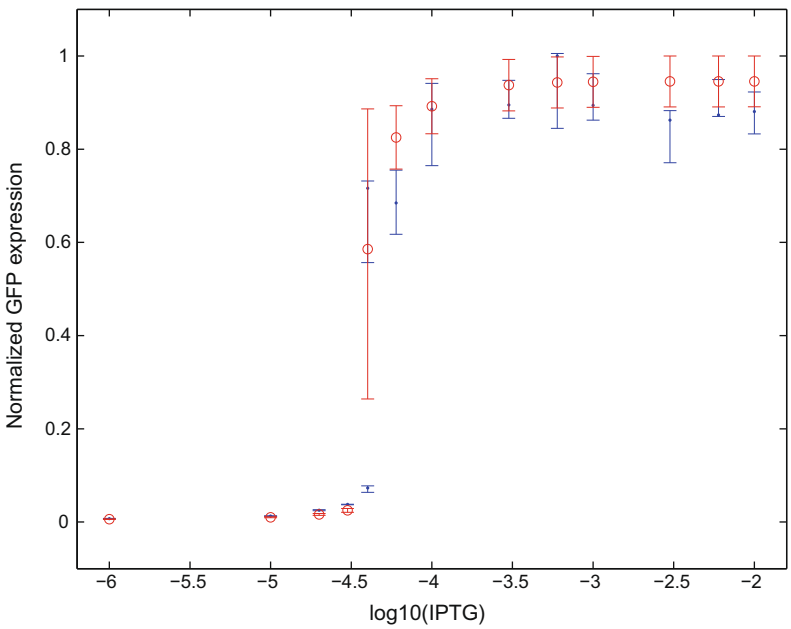

Fig. 6. Stochastic steady-state solution of v in (35). The light (red) error bars centered around the open circles display the numerical results determined by the gPC. The dark (blue) error bars centered around the dots depict the corresponding experimental measurements. (For interpretation of the references in color in this figure legend, the reader is referred to the web version of this article.)

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

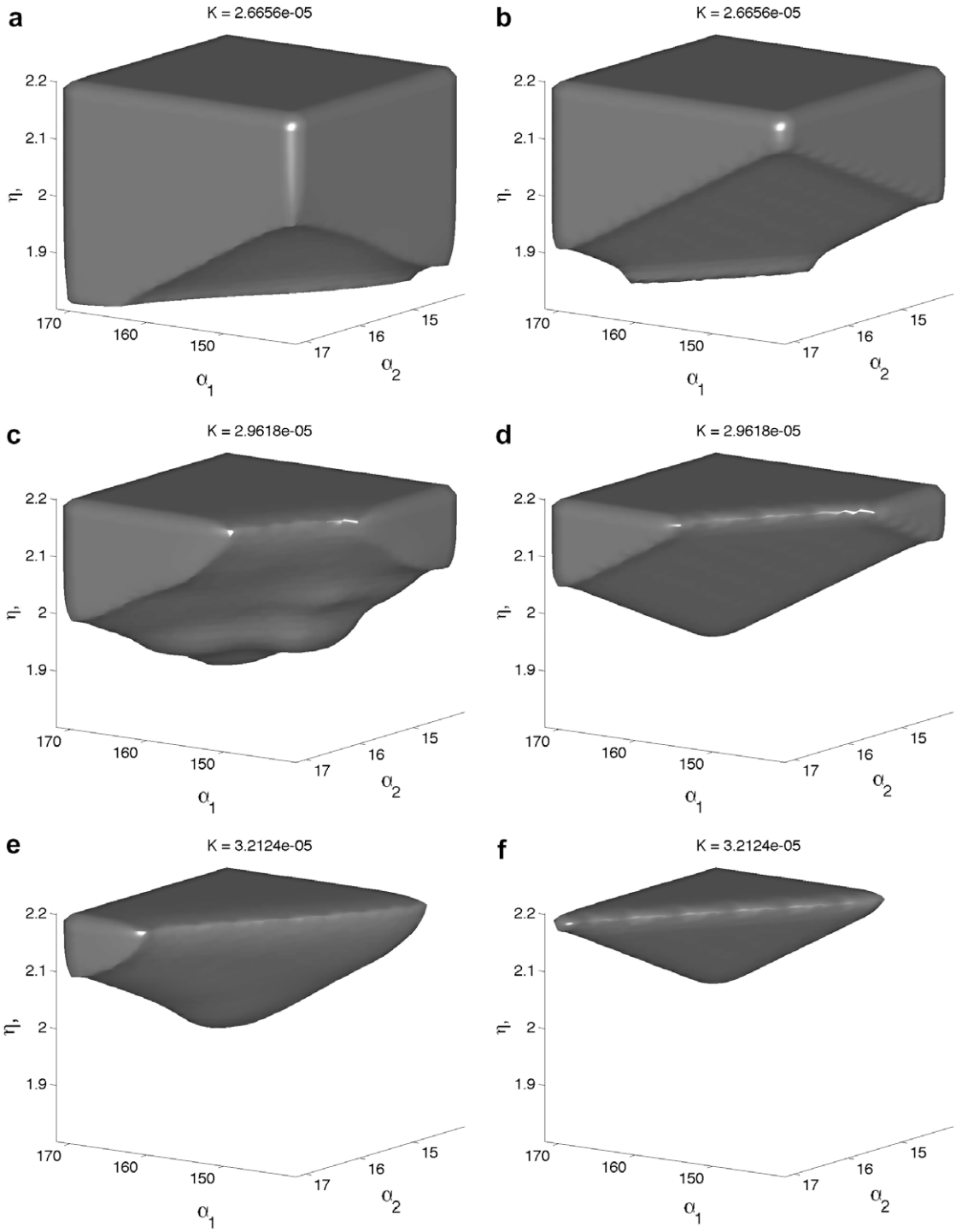

Fig. 7. Visualization of the adaptive domain classification for the stochastic differential-algebraic system (35) for the fixed parameters $\beta = 2.5$, $\gamma = 1$, $\sigma = 0.1$, and $[IPTG] = 4.0 \times 10^{-5}$. The gPC solution has two identifiable domain classes, visualized here by fixing the 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.

$p = \langle p \rangle (1 + \sigma y)$, 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 carried out in [20] by the stochastic collocation method. The solution v in (35), termed as the "Normalized GFP expression" of simulation and experimental measurement, is displayed as error bars in Fig. 6. We observe good agreement with the experimental measurements. (Details of the computation can be found in [20].)

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 (35), is shown in Fig. 7(a), (c), and (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 [18]. 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

$$f(\alpha_1, \alpha_2, \eta, K) = -0.1188\alpha_1 + 1.5849\alpha_2 + 18.5336\eta - 8.7464 \times 10^5 K - 18.49. \tag{36}$$

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 (35). 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 (36) is a crude approximation of the edge that yields a relatively accurate characterization of (35). In fact, comparison of the classification given by (36) with the steady state solution of (35) 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, Fig. 7 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 (36) provides a simple intuitive representation without compromising accuracy.

6. 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 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 subdomains 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 LLC, 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 AFOSR FA9550-08-1-0353, DOE DE-FC52-08NA28617 and NSF CAREER DMS-0645035.

References

[1] R. Archibald, A. Gelb, J. Yoon, Polynomial fitting for edge detection in irregularly sampled signals and images, SIAM J. Numer. Anal. 43 (1) (2005) 259–279.

[2] R. Archibald, A. Gelb, J. Yoon, Determining the locations and discontinuities in the derivatives of functions, Appl. Numer. Math. 58 (5) (2008) 577–592.
[3] I. Babuska, R. Tempone, G.E. Zouraris, Galerkin finite element approximations of stochastic elliptic differential equations, SIAM J. Numer. Anal. 42 (2004) 800–825.
[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. Mach. Intell. 8 (1986) 679–698.
[6] C. Canuto, M. Hussaini, A. Quarteroni, T. Zang, Spectral Methods in Fluid Dynamics, Springer Verlag, 1988.
[7] T. Gardner, C. Cantor, J. Collins, Construction of a genetic toggle switch in *Escherichia coli*, Nature 403 (2000) 339–342.
[8] A. Gelb, E. Tadmor, Adaptive edge detectors for piecewise smooth data based on the minmod limiter, J. Sci. Comput. 28 (2-3) (2006) 279–306.
[9] B. Ganapathysubramanian, N. Zabaras, Sparse grid collocation schemes for stochastic natural convection problems, J. Comput. Phys. 225 (2007) 652–685.
[10] R. Ghanem, P. Spanos, Stochastic Finite Elements: A Spectral Approach, Springer Verlag, 1991.
[11] D. Gottlieb, S. Orszag, Numerical Analysis of Spectral Methods: Theory and Applications, CBMS-NSF, SIAM, Philadelphia, PA, 1977.
[12] D. Gottlieb, D. Xiu, Galerkin method for wave equations with uncertain coefficients, Commun. Comput. Phys. 3 (2) (2008) 505–518.
[13] J. Hesthaven, S. Gottlieb, D. Gottlieb, Spectral Methods For Time-Dependent Problems, Cambridge University Press, 2007.
[14] O. Le Maitre, O. Knio, H. Najm, R. Ghanem, Uncertainty propagation using Wiener–Haar expansions, J. Comput. Phys. 197 (2004) 28–57.
[15] O. Le Maitre, O. Knio, H. Najm, R. Ghanem, Multi-resolution analysis of Wiener-type uncertainty propagation schemes, J. Comput. Phys. 197 (2004) 502–531.
[16] G. Lin, C.-H. Su, G.E. Karniadakis, The stochastic piston problem, Proc. Natl. Acad. Sci. 101 (2004) 15840–15845.
[17] I. Sobel, An isotropic 3×3 image gradient operator, in: H. Freeman (Ed.), Machine Vision for Three-Dimensional Scenes, Academic Press, Boston, 1990.
[18] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, 1995.
[19] X. Wan, G. Karniadakis, Multi-element generalized polynomial chaos for arbitrary probability measures, SIAM J. Sci. Comput. 28 (2006) 901–928.
[20] D. Xiu, Efficient collocational approach for parametric uncertainty analysis, Commun. Comput. Phys. 2 (2) (2007) 293–309.
[21] D. Xiu, Fast numerical methods for stochastic computations: a review, Commun. Comput. Phys. 5 (2009) 242–272.
[22] D. Xiu, J.S. Hesthaven, High-order collocation methods for differential equations with random inputs, SIAM J. Sci. Comput. 27 (2005) 1118–1139.
[23] D. Xiu, G. Karniadakis, The Wiener–Askey polynomial chaos for stochastic differential equations, SIAM J. Sci. Comput. 24 (2) (2002) 619–644.