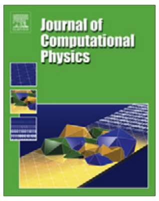
Journal of Computational Physics 230 (2011) 3977–3997

Contents lists available at ScienceDirect

Journal of Computational Physics

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



Characterization of discontinuities in high-dimensional stochastic problems on adaptive sparse grids

John D. Jakeman [a,*], Richard Archibald [b], Dongbin Xiu [c]

[a] Department of Mathematics, Australian National University, Australia
[b] Computer Science and Mathematics, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[c] Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA

ARTICLE INFO

Article history:
Received 25 June 2010
Received in revised form 31 January 2011
Accepted 16 February 2011
Available online 22 February 2011

Keywords:
Adaptive sparse grids
Stochastic partial differential equations
Multivariate discontinuity detection
Generalized polynomial chaos method
High-dimensional approximation

ABSTRACT

In this paper we present a set of efficient algorithms for detection and identification of discontinuities in high dimensional space. The method is based on extension of polynomial annihilation for discontinuity detection in low dimensions. Compared to the earlier work, the present method poses significant improvements for high dimensional problems. The core of the algorithms relies on adaptive refinement of sparse grids. It is demonstrated that in the commonly encountered cases where a discontinuity resides on a small subset of the dimensions, the present method becomes "optimal", in the sense that the total number of points required for function evaluations depends linearly on the dimensionality of the space. The details of the algorithms will be presented and various numerical examples are utilized to demonstrate the efficacy of the method.

1. Introduction

The behaviour of financial, chemical, mechanical, environmental and many other processes is often characterized by a large number of variables. The relationship between the variables that drive the system (inputs) and the system response (outputs) can be highly non-linear, discontinuous and correlated. Knowledge of the relationships between the model inputs and outputs can be extremely useful for understanding and simulation of such systems. Often we seek an approximation of such input–output relations based on a finite number of system responses. And this is one of the most studied topics of classical approximation theory.

Approximation theory in low dimensions, in fact mostly in univariate cases, is rather mature, with extensive results available on polynomial interpolation, orthogonal projections, etc. The theory for approximation in high dimensional problems is much less developed, and many fundamental mathematical issues remain unresolved.

One of the applications that requires advanced numerical methods for analysing such input–output relations is uncertainty quantification, where the inputs are random variables representing uncertain inputs for a given (potentially complex) system. Uncertainty in the input data, such as model coefficients, forcing terms, etc., is usually represented through a finite number of random variables with some known probability distribution. The goal is then to compute the system response, and in many cases, to calculate the statistics of the response. The most widely adopted numerical approaches for this problem are based on generalized polynomial chaos (gPC) expansions [1,2]. This is essentially an approximation of the high dimensional input–output relations of the system utilizing multivariate orthogonal polynomials. For a recent review, see

* Corresponding author.
E-mail addresses: john.jakeman@anu.edu.au (J.D. Jakeman), ArchibaldRK@ORNL.gov (R. Archibald), dxiu@purdue.edu (D. Xiu).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.02.022

[3]. Even when the probability distribution of the input variables is not known, similar polynomial approximation ideas can be applied [4].

While the gPC approach offers significant improvements over the traditional methods, it requires sufficient smoothness of the input–output function. If a discontinuous dependence on the input variables exists, the global polynomial approximations will suffer from the well known Gibbs phenomenon [5–8]. To circumvent the difficulty, the piecewise continuous gPC approach has been proposed and analyzed in [9,6,8]. The idea is to discretize the high-dimensional space into regions of high regularity and construct and apply a continuous approximation to each sub-domain separately to obtain a globally convergent solution [10,8]. 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. Recent efforts have used adaptivity in an attempt to allievate the difficulties arising from approximation of non-smooth functions [10,7].

Efficient identification and location of discontinuities is extremely useful. Ideally, if the presice of knowledge of the location of the discontinuities is available, one can avoid the tensor structure of the sub-domain decomposition and seek to discretize the high-dimensional domain into a minimal number of sub-domains, whose interfaces lie exactly along the discontinuities.

An initial attempt to identity and locate discontinuities in high-dimensional spaces was made in [11], where the polynomial annihilation edge detection method [12] was used. The polynomial annihilation edge detection is a high-order method and has certain advantages over the traditional methods [13–15]. It was originally developed for edge detection in one and two dimensions but can be extended to higher dimensions by applying the algorithm dimension by dimension directly to a gPC approximation [11]. However, the procedure proposed in [11] relies on examinations of the function values on (local) tensor grids and thus restricts the dimensionality that can be handled.

In this paper we present a much improved algorithm over that of [11]. The present algorithm targets the high-dimensional stochastic simulations typically encountered in uncertainty quantification. In particular, it is closely related to one of the most popular numerical implementations for gPC—sparse grid stochastic collocation [16]. Sparse grids [17–19] have been frequently used to provide approximations of high-dimensional functions, especially in stochastic simulations in recent years. The efficiency of the sparse grid collocation method has been demonstrated by many authors [20–24]. However, accurate construction of the overall solution response, either via interpolation or integration of the nodal solutions on the sparse grid, still requires sufficient smoothness of the solution. In case of the presence of discontinuities in the parameter space, the accuracy of the global sparse grids will greatly deterioriate.

The primary contribution of this paper is the development of a highly efficient method for detecting the presence of discontinuities in high dimensional spaces, based on nodal solutions of sparse grids. If a discontinuity is detected, the method can accurately identify the location and resolve the geometry of the discontinuity by employing a locally adaptive sparse grid algorithm. In many practical cases, the discontinuity resides on a manifold of smaller dimensionality. In such cases, the proposed algorithm becomes "optimal", in the sense the number of function evaluations grows linearly as the dimensionality of the space increases. Therefore, the present algorithm represents a significant improvement over the earlier work in [11] and is suitable for high dimensional problems.

2. Polynomial annihilation edge detection

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

$$S = \{x_1, x_2, \ldots, x_{\overline{Q}^{(1)}}\} \subset [-1,1]. \tag{1}$$

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

$$[f](x) = f(x^+) - f(x^-) = \begin{cases} 0, & \text{if } x \neq \xi, \\ [f](\xi), & \text{if } x = \xi. \end{cases} \tag{2}$$

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

The polynomial annihilation discontinuity detection method, introduced in [12], seeks an approximation to $[f](x)$ that converges rapidly to zero away from the jump discontinuities. It can be described as follows: Assume x is a point inside the domain, i.e., $x \in (-1,1)$. For a given positive integer $m < \overline{Q}^{(1)} - 1$, we choose a local stencil

$$S_x = \{x_j | x_j \in S\} = \{x_0, \ldots, x_m\} \tag{3}$$

[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.

of the nearest $m+1$ grid points around x. 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 discontinuity detection method is given by

$$L_m f(x) = \frac{1}{q_m(x)} \sum_{x_j \in S_x} c_j(x) f(x_j), \tag{4}$$

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

$$\sum_{x_j \in S_x} c_j(x) p_i(x_j) = p_i^{(m)}(x), \quad \forall\ i = 0, \dots, m. \tag{5}$$

Here $p_i^{(m)}(x)$ denotes the mth derivative of $p_i(x)$. Notice that the solution to (5) exists and is unique. Next we define

$$S_x^+ = \{x_j \in S_x | x_j \geqslant x\} \quad \text{and} \quad S_x^- = S_x \setminus S_x^+. \tag{6}$$

The normalization factor in (4), given by

$$q_m(x) = \sum_{x_j \in S_x^+} c_j(x), \tag{7}$$

ensures that $L_m f(x)$ has correct value at the jump discontinuities. Note that (7) is non-zero by design. In [12] it was shown that if the maximum separation $h(x)$ is defined as

$$h(x) = \max\{|x_i - x_{i-1}| : x_{i-1}, x_i \in S_x\}, \tag{8}$$

then (4) satisfies the following property:

$$L_m f(x) = \begin{cases} [f](\xi) + \mathcal{O}(h(x)), & \text{if } x_{j-1} \leqslant \xi,\ x \leqslant x_j, \\ \mathcal{O}(h^{\min(m,k)}(x)), & \text{if } f \in \mathcal{C}^k(I_x),\ k > 0. \end{cases} \tag{9}$$

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

We pause to note two important distinctions between the polynomial annihilation discontinuity detection method (4), and more traditional edge detection methods [13,14]. The polynomial annihilation discontinuity detection method is a high order reconstruction of the *jump function* $[f](x)$ defined in (2). 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 (4) is well suited when the underlying structure of the piecewise-smooth function has some variability. 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 discontinuity detection method over traditional techniques is that it does *not* require a uniform point distribution. This is particularly useful when combined with adaptive methods which do not possess an equidistant point distribution.

The order m in (4) 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 discontinuity (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 discontinuities. To prevent inaccuracies due to either of these problems, the *minmod* function, which is typically utilized in flux limiting methods to reduce oscillations when solving numerical conservation laws, was used in [12] to enhance the performance of the polynomial annihilation method.[2] Specifically, we apply

$$MM(L_m f(x)) = \begin{cases} \min\limits_{m \in \mathcal{M}} L_m f(x), & \text{if } L_m f(x) > 0, \quad \forall m \in \mathcal{M}, \\ \max\limits_{m \in \mathcal{M}} L_m f(x), & \text{if } L_m f(x) < 0, \quad \forall m \in \mathcal{M}, \\ 0, & \text{otherwise}, \end{cases} \tag{10}$$

where $\mathcal{M} \subset \mathbb{N}$ of positive integers. The minmod function controls the oscillations while still maintaining a 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 [12].

The polynomial annihilation method presented here will only capture jump discontinuities in a function. If one wishes to detect discontinuities in derivatives the method must be reformulated as discussed in [27]. However, the adaptive algorithm, which facilitates the application of discontinuity detection to high dimensional applications, does not need to be changed. The local stencils generated by the adaptive algorithm, and employed by the polynomial annihilation method (4), can still be used, only the coefficients $c_j(x)$ and normalisation factor $q_m(x)$ must be recalculated.

[2] The *minmod* technique was introduced in the context of edge detection in [25,26].

Finally, we would like to point out that the polynomial annihilation discontinuity detection method was originally developed to determine multi-dimensional jump discontinuities from randomly distributed grid points. However, the sparse grid collocation method uses structured grids in each dimension. 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.

3. Discontinuity tracking

The one-dimensional polynomial annihilation method presented in Section 2 can easily be extended to two dimensions using a triangulation procedure which identifies jump discontinuities from its surrounding points (triangle vertices) [12]. In principle, similar element-based techniques could be used for $d > 2$, but in practice such algorithms are very complicated and difficult to implement. To overcome this difficulty, Archibald et al. [11] applied one-dimensional polynomial annihilation to tensor product grids in a line-by-line fashion. Due to the curse of dimensionality inherent in tensor product formulations such an approach was limited to small and moderate number of dimensions. This problem was alleviated slightly by applying the detection algorithm on a gPC stochastic collocation approximation that can be often evaluated much more quickly than the underlying function being represented.

Here we attempt to extend this method by improving the refinement algorithm used to resolve discontinuities, whilst maintaining the tensor product structure. Starting with the hypercube $[-1,1]^d$, Archibald's discontinuity detection method identifies any hypercube containing a jump discontinuity. Every cell consisting of 2^d points and containing a discontinuity is identified and flagged for refinement. When such a cell is identified a tensor product quadrature rule is applied to this cell, and each sub cell consisting of 2^d points that contains a discontinuity is identified and the process repeated.

Here we propose an alternate procedure for detecting jump discontinuities in a moderate number of dimensions. Specifically we use a low-order sparse grid to provide an initial grid to be used for discontinuity detection. Polynomial annihilation is then used to refine this grid using a discontinuity tracking procedure.

3.1. Sparse grids

The discontinuity detection method proposed here is closely related to sparse grid approximation. Sparse grids are frequently used to reduce the curse of dimensionality and the associated computational complexity for high dimensional interpolation and quadrature. The sparse grid approximation solution is approximated by a combination of solutions to numerous but lower-dimensional regular grids. Traditionally the sub-dimensional grid solutions are constructed using Lagrange polynomials based upon a predetermined set of points [17,18]. However local basis functions can also be used [28,7].

Let us review the basic concepts of sparse grid approximation. Consider the interpolation of a function $u(\mathbf{x})$, defined over the d-dimensional hypercube $[-1,1]^d$. The sparse grid approximation is based upon the one-dimensional interpolation formula for the univariate function $u^{(1)}$

$$\mathcal{I}_k u^{(1)} = \sum_{j=1}^{m_k} u^{(1)}(x_k^j) \cdot \Psi_k^j(x) \tag{11}$$

for a set of abscissas $\mathbf{x}_k = (x_k^1, \dots, x_k^{m_k})$ and univariate bases $\{\Psi_k^j\}_{j=1}^{m_k}$. Superficially we can approximate u using Smolyak's algorithm [29]

$$\mathcal{I}_l u = \sum_{l-d+1 \leqslant |\mathbf{k}| \leqslant l} (-1)^{l-|\mathbf{k}|} \cdot \binom{d-1}{l-|\mathbf{k}|} \cdot (\mathcal{I}_{k_1} \otimes \cdots \otimes \mathcal{I}_{k_N}) u(\mathbf{x}), \tag{12}$$

where $\mathbf{k} = (k_1, \dots, k_d)$ and $|\mathbf{k}|_1 = \sum_{j=1}^d k_j$. Of all the possible tensor products of one-dimensional formulas only the combinations associated with the indices $l-d+1 \leqslant |\mathbf{k}|_1 \leqslant l$ are considered. This defines an isotropic sparse grid which treats all dimensions equally assigning decreasing importance to higher-order interactions terms. In contrast a full tensor product formulation treats all dimensions and interactions equally. The sparse grid construction significantly delays the curse of dimensionality in comparison to a full tensor product approach.

To construct the sparse grid approximation the function must be evaluated at all the abscissa in the sparse grid

$$\mathcal{G}_l = \bigcup_{l-d+1 \leqslant |k|_1 \leqslant l} (\mathbf{x}_{k_1} \times \cdots \times \mathbf{x}_{k_d}). \tag{13}$$

Typically the nodes are chosen to ensure that the set of abscissa $\mathbf{x}_k = (x_k^1, \dots, x_k^{m_k})$ chosen to interpolate each univariate component of u are nested, i.e. $\mathbf{x}_{k-1} \subset \mathbf{x}_k$. This allows the level of interpolation to be increased from $k-1$ to k by evaluating the function only at the $m_k^\Delta = m_{k-1} - m_k$ points $\mathbf{x}_k^\Delta = \mathbf{x}_k \setminus \mathbf{x}_{k-1}$. Similarly the level of interpolation of the d-dimensional sparse grid can be increased from level $l-1$ to l by evaluating the function at the abscissa

$$\Delta\mathcal{G}_l = \bigcup_{|k|_1 = l} \left(\mathbf{x}_{k_1}^\Delta \times \cdots \times \mathbf{x}_{k_d}^\Delta\right). \tag{14}$$

3.2. Discontinuity tracking algorithm

Sparse grids provide an efficient means of sampling a high-dimensional input space. Here we use sparse grids to provide a set of points which provide a reasonable coverage of the input space, and which can be used for initially identifying any discontinuities. The intervals of the sparse grid can then be refined in a line-by-line fashion, using polynomial annihilation, until a discontinuity is resolved up to a pre-specified tolerance δ. This typically occurs first only at one or a small number of locations within the domain. At these locations a small tensor product grid is generated and assessed to determine the local extension of the discontinuity. Every time an interval is created with a length smaller than δ its local neighbourhood is searched for neighbouring discontinuity intervals. Such a discontinuity tracking procedure removes the need for some points further away from the discontinuity which are necessary when refining in a top down fashion, as with the cell division method. As the local procedure evolves, the larger grid is continually adapted to identify additional discontinuity intervals for local refinement. This discontinuity tracking procedure is outlined in Algorithm 1.

We remark that, when constructing the local stencils $S_{x,i}$ and $S_{x,j}$ used to build the local stencil $\mathcal{L}$, special care must be taken in regions adjacent to the boundaries of the computational domain. In such situations any point in $S_{x,i}$ or $S_{x,j}$ outside the domain is discarded. It is possible that during refinement some points may have already been added to the sparse grid, so adequate book keeping is required to keep track of the points added.

Example 3.1. Consider the following function:

$$f(\mathbf{x}) = \begin{cases} 1, & \sum_{i=1}^{d} x_i^2 < r^2 \quad \text{and} \quad x_1 < 0 \\ x_1, & \text{otherwise}. \end{cases} \tag{15}$$

Fig. 1 displays the evolution of the discontinuity tracking algorithm as it is applied to Example 3.1 for $d = 2$, $r = 0.5$, $\delta = 2^{-4}$, $m = 2$, and $l_{\text{init}} = 3$. Fig. 1(a) depicts the initial grid and Fig. 1(b) displays the adaptive grid after three iterations. Here discontinuity points are displayed in green, and new midpoints to perform discontinuity detection are displayed in red. At this iteration, the first set of discontinuity points are identified and a local stencil is added in the neighborhood of these discontinuity points. Fig. 1(c) displays the adaptive grid after eight iterations. Notice that the adaptive grid tracks the location of the

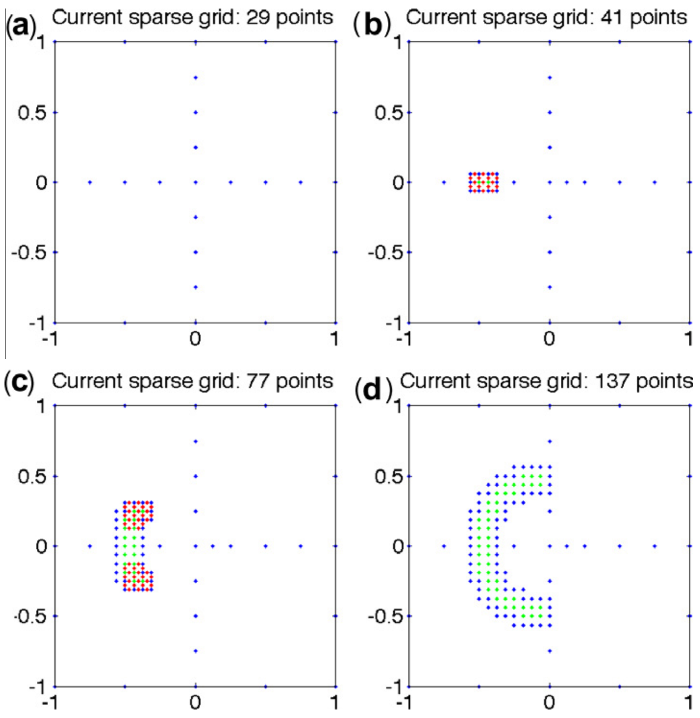


Fig. 1. (a) Initial grid with the limit of resolution set to $\delta = 2^{-4}$; (b) adaptive grid after three iterations – here discontinuity points are displayed in green, and new midpoints to perform discontinuity detection are displayed in red; (c) adaptive grid after eight iterations; (d) final grid. To obtain the same resolution using an isotropic sparse grid requires 7169 function evaluations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

discontinuity and only the midpoints at the front of the discontinuity tracking are evaluated for discontinuities. Fig. 1(d) depicts the final sparse grid, where it can be seen that the discontinuity in Example 3.1 is captured with a uniform grid with a width of order $m + 1$. Some 137 points are needed to capture the discontinuity to the desired resolution. To obtain the same resolution using an isotropic sparse grid requires 7169 function evaluations.

To facilitate convergence analysis we use distance to the nearest discontinuity to assess the accuracy of the representation of a discontinuity. Using the tensor-product discontinuity tracking procedure, such a distance function can be approximated by

$$D_E(\mathbf{x}) = \min\left\{\left\|\mathbf{x} - \left(y_1,\ldots,\frac{y_i+z_i}{2},\ldots,y_d\right)\right\|_2 : (\mathbf{y},\mathbf{z}) \in E\right\}. \tag{16}$$

Example 3.2. Consider the following function:

$$f(\mathbf{x}) = \begin{cases} 1, & \sum_{i=1}^{k} x_i^2 < r^2, \quad k \leqslant d, \\ 0, & \text{otherwise}. \end{cases} \tag{17}$$

Algorithm 1. Given an initial grid, a desired resolution of δ and a desired order of accuracy m, the following steps are iterated until no new grid points are determined to be added

```
1. Generate an initial isotropic sparse grid of level $l_{\text{init}}$ and add all the points to $\mathcal{S}$
2. Define $\bar{S}$ as the set of any two points in $\mathcal{S}$ that are nearest neighbours in an axial direction.
3. Find the set of new grid points to be added

   $\mathcal{A} = \left\{\mathbf{x} = \left(y_1,\cdot,\frac{y_i+z_i}{2},\cdot,y_d\right) : (\mathbf{y},\mathbf{z}) \in \bar{S},\ MM(L_m f(x_i)) > \mathcal{O}(|y_i - z_i|^m),\ \text{and } |y_i - z_i| \geqslant 2\delta\right\}$

   where $i$ is the axial direction along which the discontinuity was found
4. While $A \neq \emptyset$
   • Add the new points in $\mathcal{A}$ to the grid $\mathcal{S}$
   • Update $\bar{S}$
   • Find the set of new grid points to be added $\mathcal{A}$
   • Find the set of discontinuity points given by

     $\mathcal{E} = \left\{(\mathbf{y},\mathbf{z}) : (\mathbf{y},\mathbf{z}) \in \bar{S}, x_i = \frac{y_i+z_i}{2},\ MM(L_m f(x_i)) > \mathcal{O}(|y_i - z_i|^m),\ \text{and } |y_i - z_i| \leqslant \delta\right\}$

   • For each discontinuity point $\mathbf{z}$ in $\mathcal{E}$, refine the grid locally
     – Define $S_{x,i}$ to be the local one-dimensional four point stencil containing the discontinuity $\mathbf{z}$ $(z_i - \delta, z_i - \delta/2, z_i, z_i + \delta/2)$
     – Define $S_{x,j}$, $j = 1,\ldots,d$, $i \neq j$ to be the one-dimensional three point stencil $(z_j - \delta, z_j, z_j + \delta)$
     – Construct the tensor product grid $\mathcal{L} = S_{x,1} \otimes \cdots \otimes S_{x,i}, \otimes \cdots \otimes, S_{x,d}$ and add the $4 \cdot 3^{d-1}$ points to the grid $S$
     – identify all intervals of $\mathcal{L}$ containing a discontinuity and add the associated discontinuity point to $\mathcal{E}$.
```

which has a simple analytic expression of the distance function to the nearest discontinuity

$$D_f(\mathbf{x}) = \left| r - \left(\sum_{i=1}^{k} x_i^2\right)^{\frac{1}{2}} \right|. \tag{18}$$

Table 1 reports the number of points in the grid generated by the discontinuity tracking algorithm for Example 3.2 and the resulting error in the estimation of the distance function to the nearest discontinuity given in (18) as dimensionality is increased. Here we choose the initial grid to be a first level sparse grid, $m = 2$, $\delta = 2^{-4}$ and $k = d$. It can be seen that the error in the estimation of the distance function to the nearest discontinuity is maintained at $\mathcal{O}(\delta)$, and the number of grid points is proportional to the dimension and hyper-dimensional surface area of the functional discontinuity.

Fig. 2 further illustrates the dependency of the number of points required to resolve a discontinuity on the dimension, hyper-dimensional surface area and the resolution δ. Fig. 2(a) shows that the number of points required to resolve a discontinuity, up to a certain resolution, is dependent on the hyper-dimensional surface area which grows with dimensionality. Fig. 2(b) highlights that for fixed surface area the number of points required is inversely proportional to δ. The small

Table 1
The number of points N and the error in the approximated distance function of Example 3.2 with $\delta = 1/32$ and $r = 1/8$, for increasing dimension d.

d	N	$\epsilon_{\max}$
2	57	0.03171
3	293	0.03070
4	1585	0.02848
5	8253	0.02516
6	40617	0.03163
7	189717	0.03191

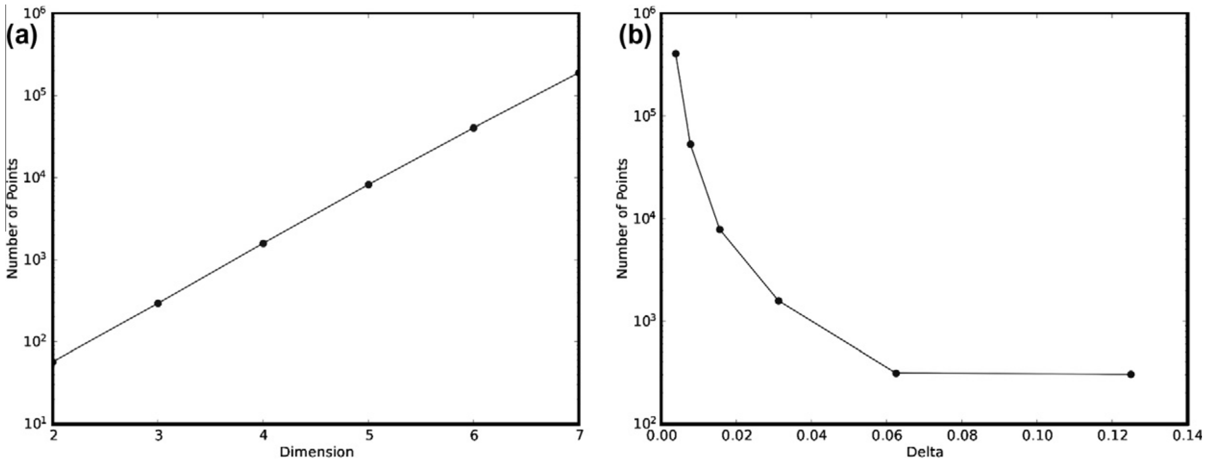


Fig. 2. (a) the number of points required to resolve the discontinuity of Example 3.2 ($r = 1/8, \delta = 1/32$) as dimensionality increases; (b) the number of points required to resolve the discontinuity of Example 3.2 ($r = 1/8, d = 4$) as the size of δ decreases.

difference in the number of points required for $\delta = 1/16$ and $\delta = 1/32$ is because the small discontinuity arising from a radius of 1/16 is indistinguishable at these coarser resolutions.

If the discontinuity exists along all axial directions the dependence of the number of points necessary on the hyper-dimensional surface area cannot be broken. However, if a discontinuity only exits in a small subset of dimensions, this dependence can be broken and the curse of dimensionality inherent in the tensor product formulation can be delayed significantly. Consider Example 3.2 with $k = 2$ and $d = 3$ the discontinuity manifold is a cylinder. Fig. 3 shows the results of the discontinuity tracking procedure applied under these conditions. For any value of x_2 (the perpendicular axis) a discontinuity can be found along the remaining axial directions. However after searching along lines parallel to the perpendicular axis no discontinuity is present. In such situations the number of points needed to capture the discontinuity manifold could be drastically reduced by preventing refinement in the perpendicular axis. This proposition is the basis of the discontinuity detection method presented in the following section.

4. High-dimensional discontinuity detection

In a d-dimensional space any discontinuity will lie on a $d-1$ manifold. The number of points needed to characterise such a manifold is proportional to its volume. Thus the number of points needed to represent a discontinuity will increase rapidly with dimensionality. The main aim of the discontinuity detection methods proposed here is to find the lower-dimensional structure of the manifold embedded in the higher-dimensional space. Specifically the proposed algorithms attempt to work in the low-dimensional latent space of the manifold to provide an efficient characterisation of the discontinuity manifold.

Dimension-wise decompositions of the input–output relationship have arisen as a successful means of delaying or even breaking the curse of dimensionality for certain applications. Such an approach represents the model outputs as a high-dimensional function $f(\mathbf{x})$ dependent on the model inputs $\mathbf{x} = (x_1,\ldots,x_d)$. It uses an ANOVA type decomposition

$$f(x_1,\ldots,x_d) = f_0 + \sum_{n=1}^{d} f_i(x_i) + \sum_{i,j=1}^{d} f_{i,j}(x_i,x_j) + \cdots + f_{i,\ldots,d}(x_i,\ldots,x_d), \tag{19}$$

which separates the function into a sum of subproblems. The first term is the zeroth order effect which is a constant throughout the d-dimensional variable space. The $f_i(x_i)$ terms are the first-order terms which represent the effect of each variable

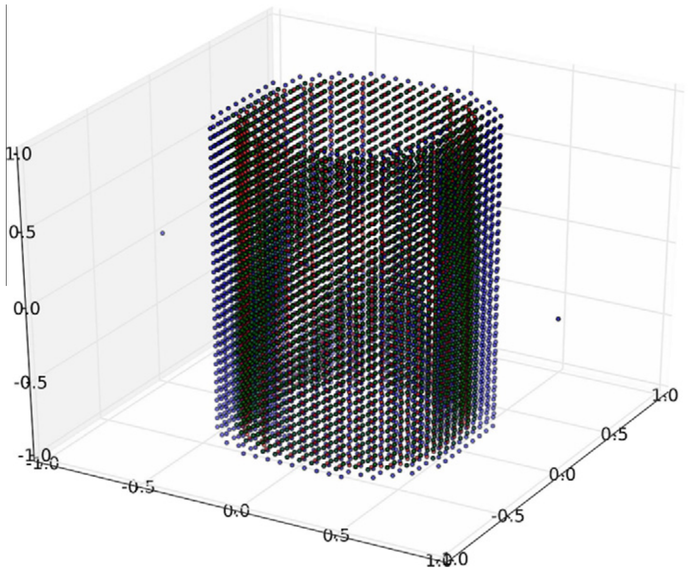


Fig. 3. Discontinuity tracking scheme applied to Example 3.2 with $k = 2$, $d = 3$, $r = 0.5$ and $\delta = 1/16$.

acting independently of all others. The $f_{i,j}(x_i, x_j)$ terms are the second-order terms which are the contributions of x_i and x_j acting together, and so on.

In practice only a small number of interactions contribute significantly to the system response and consequently only a low-order approximation is needed to represent the input–output mapping accurately [30]. Moreover only a small subset of the remaining terms may actually control the location of discontinuities and/or kinks.

In this section we propose a locally-adaptive approach to determine and resolve the location of any discontinuities. The method only refines in dimensions deemed to contribute to the location of the discontinuity, thereby expending more effort in regions which likely contain a discontinuity. The polynomial annihilation method is used to guide refinement that invests the majority of function evaluations in regions surrounding discontinuities and neglects the grid points in the smooth regions.

4.1. Adaptive sparse grids

The discontinuity detection method proposed here is closely related to adaptive sparse grid approximation. In comparison to isotropic sparse grids, presented in Section 3.1, which treat all dimensions and regions of the input space equally, adaptive sparse grids spend more effort sampling a function in areas and dimensions of interest. Here we extend the adaptive sparse grid framework used for interpolation and integration [28,7] to discontinuity detection.

The sparse grid interpolant $\mathcal{I}_l u$ can be written using a hierarchical formulation

$$\mathcal{I}_l u = \sum_{|\mathbf{k}|_1 \leqslant l} \sum_{\mathbf{j} \in B_\mathbf{k}} w_\mathbf{k}^\mathbf{j} \cdot \left(\Psi_{k_1}^{j_1} \otimes \cdots \otimes \Psi_{k_d}^{j_d} \right)(\mathbf{x}). \tag{20}$$

The coefficients $w_\mathbf{k}^\mathbf{j}$ are calculated in a hierarchical manner such that

$$w_\mathbf{k}^\mathbf{j} = u\left(\mathbf{x}_\mathbf{k}^\mathbf{j}\right) - \mathcal{I}_{l-1} u\left(\mathbf{x}_\mathbf{k}^\mathbf{j}\right), \tag{21}$$

where the function is evaluated at the d-variate node $\mathbf{x}_\mathbf{k}^\mathbf{j} = (x_{k_1}^{j_1}, \ldots, x_{k_d}^{j_d})$ if its associated index $\mathbf{j}$ belongs to

$$B_\mathbf{k} = \left\{ \mathbf{j} \in \mathbb{N}^d \middle| x_{k_i}^{j_i} \in \mathbf{x}_{k_i}^{\Delta} \text{ and } |w_\mathbf{k}^\mathbf{j}| > \epsilon \text{ for } j_i = 1, \ldots, m_{k_i}^{\Delta}, \quad i = 1, \ldots, d \right\} \tag{22}$$

If $\epsilon = 0$ the hierarchical formulation (20) is equivalent to (12). To increase the level of interpolation of the d-dimensional sparse grid from $l - 1$ to l one must evaluate the function at the abscissa

$$\Delta \mathcal{G}_l = \left\{ \mathbf{x}_\mathbf{k}^\mathbf{j} \middle| |\mathbf{k}|_1 = l \text{ and } \mathbf{j} \in B_\mathbf{k} \right\}. \tag{23}$$

The hierarchical formulation (20) lends itself well to local refinement. Setting $\epsilon > 0$ adaptively selects which nodes are added to the sparse grid. Subsequently the points in the sparse grid become concentrated in discontinuous or rapidly varying regions. The number of points in the spare grid depends on the value of ϵ and the level of interpolation l.

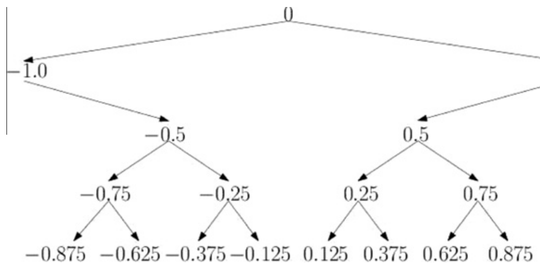


Fig. 4. Tree like structure of the 1D sparse grid coordinates.

When adaptive sparse grid approximation is applied to functions in regions of high regularity the hierarchical surpluses w decay rapidly as the level of interpolation l increases. However in regions containing discontinuities the hierarchical surpluses decay much slower. Consequently the amplitude of the hierarchical surpluses provides an estimate of the local error and can be used as means for determining the need for refinement. Such a procedure is able to resolve discontinuities but it cannot directly identify their location.

In this paper we will consider sparse grids based upon equidistant 1D abscissa. We utilise the observation of Ma and Zabaras [7] that these 1D equidistant points can be considered as a tree like-data structure as shown in Fig. 4. We can consider the refinement level of the sparse grid as the depth of the tree $D(\mathbf{y})$. For example the level of point 0.25 is three. Subsequently the abscissa of the sparse grid $\mathcal{G}_l$ given by (13) can be written equivalently as

$$\mathcal{G}_l = \left\{ \mathbf{x} = (x_1, \dots, x_d) \middle| \sum_{i=1}^{d} D(x_i) \leqslant l \right\}. \tag{24}$$

Denoting the parent of a 1D grid point x as $F(x)$ the $2d$ children of a d-dimensional point $\mathbf{x}$ are

$$\{\mathbf{y} | (F(y_1), y_2, \dots, y_d) = \mathbf{x} \text{ or } (y_1, F(y_2), \dots, y_d) = \mathbf{x} \text{ or } (y_1, y_2, \dots, F(y_d)) = \mathbf{x}\}, \tag{25}$$

where the parent of the root is itself, i.e. $F(0) = 0$. Special treatment is required to progress from level 1 to 2. The boundary nodes on level 1 have only one child.

The sparse grid approximation procedure is presented in Algorithm 2. The goal is to find the set of points such that the corresponding interpolation error is as small as possible for a given amount of work. We start with the root point located at the middle of the hypercube $[-1,1]^d$. We then determine the $2d$ neighbours and evaluate the function at these points. If the hierarchical surplus at any of these points is above the threshold, i.e. $w_{\mathbf{k}}^{\mathbf{j}} \geqslant \epsilon$, the point is added to the active point set $\mathcal{A}$. Otherwise it is added to the sparse grid $\mathcal{S}$. Once all points have been assessed, the level of interpolation is increased from $l = 1$ to $l + 1$, the points in $\mathcal{O}$ are added to the sparse grid $\mathcal{S}$, the active point set is added to the old index set $\mathcal{O}$, and $\mathcal{A}$ is cleared. The $2d$ children of each of the points in $\mathcal{O}$ are determined and checked for refinement and the appropriate points added to $\mathcal{A}$. This process is repeated until the maximum level of interpolation $l_{\max}$ is reached or no points with $w_{\mathbf{k}}^{\mathbf{j}} \geqslant \epsilon$ remain.

Algorithm 2. Adaptive sparse grid refinement algorithm

```
$l := 1$
$\mathbf{x} := (0,\dots,0)$
$\mathcal{S} := \emptyset$
$\mathcal{O} := \emptyset$
$\mathcal{A} := \{\mathbf{x}\}$
**while** ( $\mathcal{A} \neq \emptyset$ and $l \leqslant l_{\max}$ ) **do**
  $\mathcal{S} := \mathcal{O}$
  $\mathcal{O} := \mathcal{A}$
  **for** ( $\mathbf{x} \in \mathcal{O}$) **do**
    **for** (dir $\in \{1,\dots,d\}$) **do**
      $\mathcal{C}$=FindAxialChildren ($\mathbf{x}$, dir)
      **for** ($\mathbf{y} \in \mathcal{C}$) **do**
        Refine ($\mathbf{y}$, dir)
      **end for**
    **end for**
  **end for**
  $l$++
**end while**
```

Algorithm 3. Refine($\mathbf{y}$,dir)

```
if ($\mathbf{y} \notin \mathcal{S} \cup \mathcal{A}$) then
  $\mathbf{s}$, $\mathbf{f}$, $i_{\text{cur}}$ = FindLocalAxialStencil ($\mathbf{y}$,dir,$m$)
  if NearEdge($\mathbf{s},\mathbf{f},i_{\text{cur}},m,\epsilon$) then
    if (Support ($\mathbf{y}$,dir) $\leqslant 2\delta$) then
      $\mathcal{E} := \mathcal{E} \cup \{\mathbf{y}\}$
    else
      $\mathcal{A} := \mathcal{A} \cup \{\mathbf{y}\}$
    end if
  end if
end if
```

4.2. Sparse grid discontinuity detection

When endeavouring to locate discontinuities, we are no longer concerned with representing a function with some level of accuracy but rather determining the location of possible discontinuities in u. With this aim we propose using the polynomial annihilation method to determine if a point is close to a discontinuity instead of using the hierarchical surplus $w_{\mathbf{k}}^{\mathbf{j}}$ or similar measures.

One major advantage of employing the polynomial annihilation method is that it does not require a function evaluation to determine if a point is close to a discontinuity. A function evaluation is only performed once the choice to refine has been made. This is in contrast to many other refinement strategies, locally adaptive sparse grid interpolation for example, which must evaluate the function at a point to determine whether refinement is necessary.

The fundamental heuristic of the proposed procedure is to use the basic adaptive sparse grid procedure (Algorithm 2), in conjunction with the polynomial annihilation method, to determine whether the refinement of a point in the sparse grid will lead to a more accurate characterization of any discontinuities. This can be achieved by simply modifying the `Refine ()` procedure presented in Algorithm 3.

Algorithm 4. The following outlines the procedure used to combine polynomial annihilation with sparse grid adaptivity to capture discontinuities up to a pre-defined resolution δ. We call this method the sparse grid discontinuity detection (SGDD) method

1. Initialise the active coordinate index set $\mathbf{u} = \emptyset$
2. Construct the first level of the sparse grid
   - Add the root point (0,...,0) to the sparse grid $\mathcal{S}$
   - Evaluate and add the 2d children of the root point to the active point set $\mathcal{A}$
   - Set the level of interpolation to one, i.e. $l = 1$
3. While $l \leqslant l_{\max}$ and $\mathcal{A} \neq \emptyset$
   - Copy the points in $\mathcal{A}$ to $\mathcal{O}$ and clear $\mathcal{A}$
   - For each point in $\mathcal{O}$
     - Generate its 2d children
     - Use polynomial annihilation to determine which children are near a discontinuity
     - If a point is near a discontinuity along an axial direction i, set $\mathbf{u} = \mathbf{u} \cup \{i\}$
     - If a point is near a discontinuity and its support is larger than δ, evaluate and add the point to $\mathcal{A}$
     - If a point is near a discontinuity and its support is not larger than δ, add the point to $\mathcal{E}$
   - Find, evaluate and add any missing ancestors of points in $\mathcal{A}$ to $\mathcal{A}$.
   - Copy the points in $\mathcal{O}$ to the sparse grid $\mathcal{S}$ and clear $\mathcal{O}$
   - Increment the level of interpolation i.e. $l = l + 1$

A fundamental property of sparse grids is that for each point in the sparse grid, all points in the hierarchical tree of ancestors must exist. To ensure this condition is met, whenever a point is added to the sparse grid all its missing ancestors must be added. This can easily be done recursively.

Note an mth order univariate PA method requires a local stencil with at least $m + 1$ points with known function values. Special treatment is needed when refining any point $\mathbf{x}$ in the axial direction $i = 1,\dots,d$, if the ith coordinate of the point x_i is 0. The children of these points will only have one neighbouring point (its parent) in one or more of the axial directions. When refining the root node $(0,\dots,0)$ we simply add every child. However if the children, grandchildren, etc. were refined in a similar manner, the dimensionality of the sparse grid would grow indefinitely and the number of points evaluated would grow

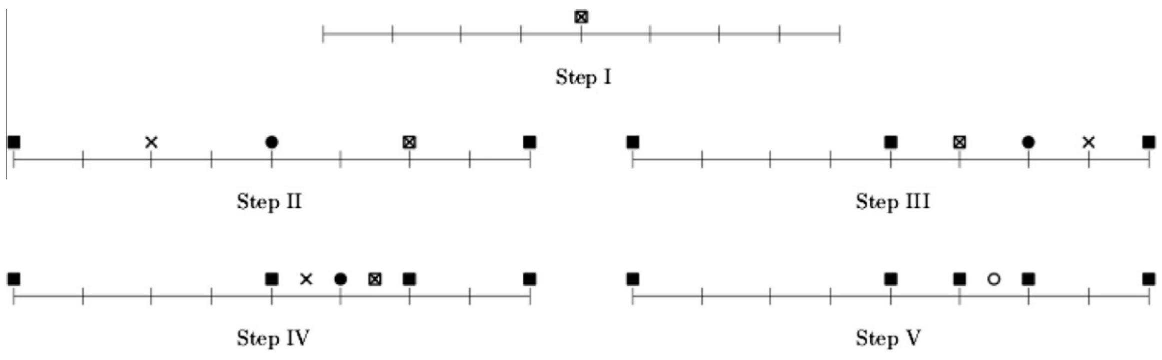


Fig. 5. Five steps of the discontinuity detection algorithm applied to a 1D domain with a discontinuity at $x = 1/3$ and $\delta = 0.25$. Squares represent points in the current sparse grid $\mathcal{S}$, black circles represent points in the old points set, $\mathcal{O}$ crosses represent the children of the points in $\mathcal{O}$, and the square surrounding the cross indicates the mid point of the interval containing the discontinuity which is added to $\mathcal{A}$. Empty circles represent the points in the discontinuity point set $\mathcal{E}$.

exponentially. To avoid this problem we do not refine any boundary point ($\{\mathbf{x}|\exists i \text{such that } x_i = -1 \text{ or } x_i = 1\}$) in the jth direction if $x_j = 0$.

It is possible that during refinement some points may have been already been added to the sparse grid so adequate book keeping is required to keep track of the points added. Furthermore the points in the local stencil may not be equally spaced. So the data structures used to store the points must allow efficient searching and inserting of points. With this aim we use hash-tables with unique keys based upon the coordinates of each point.

Once an interval is created whose length is smaller than the pre-specified threshold δ, the midpoint of the interval is added to the set of discontinuity points $\mathcal{E}$. This means that no interval will be created that has a length smaller than δ.

This algorithm starts with a first-level sparse grid with $2d + 1$ points. The algorithm will identify and locate discontinuities provided this initial grid possesses at least one interval with a discontinuity. If no discontinuity is found the level of the initial sparse grid can be increased to provide a more comprehensive search for a discontinuity.

As the algorithm evolves it identifies all the dimensions along which a discontinuity can be found. We refer to these as active dimensions, the coordinate indices of which are stored in $\mathbf{u}$.

An example of the adaptive-discontinuity detection algorithm in 1D is shown in Fig. 5 and an example in 2D is shown in Fig. 6. The adaptive nature of the proposed algorithm automatically selects and refines the dimensions that cause a discontinuity. Boundary points are added in the other dimensions to determine if a discontinuity is present in those directions but the overwhelming number of function evaluations is spent in the aforementioned dimensions.

Fig. 7 shows the grid points used by Algorithm 2 to characterise the discontinuity of the function given in Example 3.2 with $k = 2$ and $d = 3$. The sparse grid detection scheme only resolves the discontinuity along 2D slices at $x_2 = (-1,0,1)$. Consequently the number of points used, 747, is much less than the 7007 points used by the discontinuity tracking scheme Algorithm 1. See also Fig. 3.

5. High-dimensional discontinuity tracking

To further reduce the number of points required to identify the location of a discontinuity a heuristic can be used to identify and truncate the unimportant dimensions. Here we suggest that, if a minimum level of refinement has been reached in a certain axial direction and no discontinuity has yet been found, it is likely no discontinuity exists in that direction. As each such direction is identified they should be flagged so that no more points are invested in these un-important dimensions.

In this vein we propose a high-dimensional discontinuity tracking scheme that combines the strength of the adaptive sparse grid discontinuity detection Algorithm 2, to detect and refine active dimensions, and the efficiency of the tensor-product discontinuity tracking Algorithm 1.

The discontinuity tracking Algorithm 1 proposed in Section 3 uses local tensor product grids around known discontinuity points to determine neighbouring discontinuity points. This method is very efficient when the dimensionality of the computational domain is small. However the tensor-product construction suffers the curse of dimensionality, which limits the practical application to a moderate number of dimensions. The algorithm uses a low-order sparse grid to determine an initial set of intervals containing discontinuities, which are then refined up to a pre-defined size. Once a discontinuity point is identified the tracking scheme is invoked to propagate knowledge of the discontinuity location out from this point.

We can extend the utility of the discontinuity tracking scheme by only tracking a discontinuity in active dimensions, i.e. dimensions along which a discontinuity has been discovered. The initial sparse grid can be used to identify these dimensions. However the complete identification of a discontinuity relies on the intervals between the points containing at least one discontinuity in each active dimension. This is possible provided the initial level of the sparse grid is high. As the dimensionality is increased, however, generating the number of points in the sparse grid quickly becomes infeasible. We propose using the

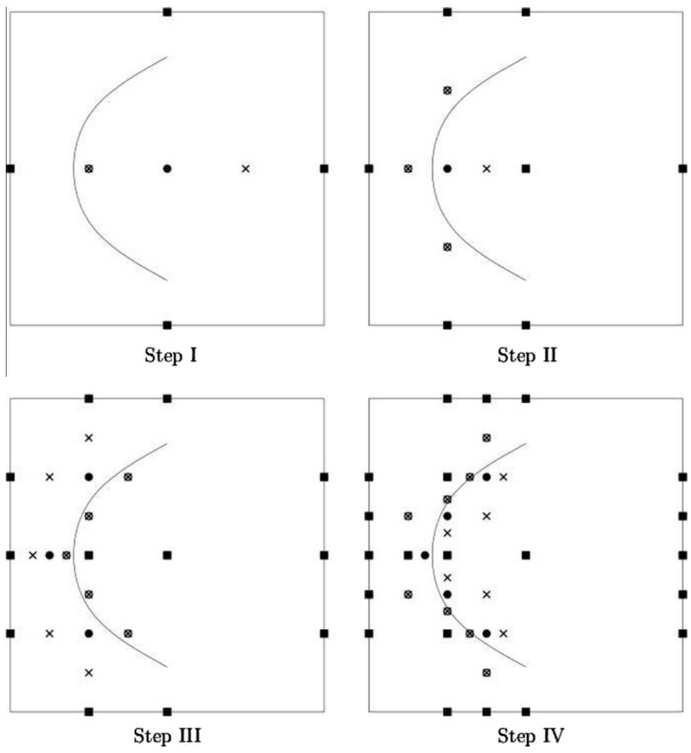


Fig. 6. Four steps of the discontinuity detection algorithm. The solid line represents the discontinuity, the squares represent points in the current sparse grid $\mathcal{S}$, black circles represent points in the old points set $\mathcal{O}$, crosses represent the children of the of points in $\mathcal{O}$ and a circle surrounding a cross indicates the mid point of an interval containing the discontinuity which is to be added to $\mathcal{A}$.

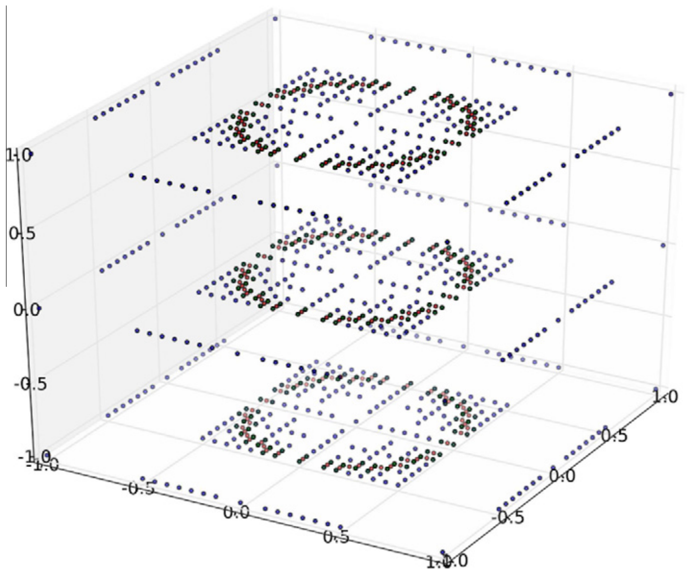


Fig. 7. Edge tracking scheme applied to Example 3.2 with $k = 2$, $d = 3$, $r = 0.5$, and $\delta = 1/16$.

sparse grid detection Algorithm 4 to provide the initial grid and then use the discontinuity tracking Algorithm 1 to resolve the discontinuity completely.

Algorithm 5 proceeds in the manner specified by Algorithm 4, except that now every time a new discontinuity point is found the discontinuity tracking Algorithm 1 is used to track the discontinuity in the adjacent regions.

The set $\mathbf{u}$ can be used to reduce the number of points invested in the inactive dimensions. Instead of generating the $2d$ children of every active point, we only generate $2|\mathbf{u}|$ children by refining only in the active dimensions whose coordinate indices are in $\mathbf{u}$. The cardinality of $\mathbf{u}$ is referred to as the truncation dimension in the sparse grid approximation literature [31]. Accordingly we will refer to the process of restricting refinement to active dimensions as truncation.

Algorithm 5. The following outlines the procedure used to combine polynomial annihilation with sparse grid adaptivity to capture discontinuities up to a pre-defined resolution δ. We call this method the Sparse Grid Discontinuity Detection (SGDD) method

```
1. Initialise the active coordinate index set $\mathbf{u} = \emptyset$
2. Construct the first level of the sparse grid
   • Add the root point $(0,\dots,0)$ to the sparse grid $\mathcal{S}$
   • Evaluate and add the $2d$ children of the root point to the active point set $\mathcal{A}$
   • Set the level of interpolation to one, i.e. $l = 1$
3. While $l \leqslant l_{\max}$ and $\mathcal{A} \neq \emptyset$
   • Copy the points in $\mathcal{A}$ to $\mathcal{O}$ and clear $\mathcal{A}$
   • For each point in $\mathcal{O}$
         –Generate its children
            ∗If $\mathbf{u}$ is complete find children in each direction $i \in \mathbf{u}$
            ∗If $\mathbf{u}$ is not complete find children in each direction $i \in \{1,\dots,d\}$
         –Use polynomial annihilation to determine which children are near a discontinuity
         –If a point is near a discontinuity along an axial direction $i$, set $\mathbf{u} = \mathbf{u} \cup \{i\}$
         –If a point is near a discontinuity and its support is larger than $\delta$, evaluate and add the point to $\mathcal{A}$
         –If a point is near a discontinuity and its support is not larger than $\delta$
            ∗Add the point to $\mathcal{E}$
            ∗Define $S_{x,i}$ to be the local one-dimensional four point stencil containing the discontinuity $\mathbf{z}$ $(z_i - \delta, z_i - \delta/2, z_i, z_i + \delta/2)$
            ∗Define $S_{x,j}$, $j = 1,\dots,d$, $i \neq j$ to be the one-dimensional three point stencil $(z_j - \delta, z_j, z_j + \delta)$
            ∗Construct the tensor product grid $\mathcal{L} = S_{x,1} \otimes \cdots \otimes S_{x,i}, \otimes \cdots \otimes, S_{x,d}$and add the $4 \cdot 3^{d-1}$ points to the grid $S$.
            ∗Identify all intervals of $\mathcal{L}$ containing a discontinuity and add the associated discontinuity point to $\mathcal{E}$.
   • If $\mathbf{u}$ is not complete find, evaluate and add any missing ancestors of points in $\mathcal{A}$ to $\mathcal{A}$.
   • Copy the points in $\mathcal{O}$ to the sparse grid $\mathcal{S}$ and clear $\mathcal{O}$
   • Increment the level of interpolation, i.e. $l = l + 1$
```

Initially the set $\mathbf{u}$ is empty so we cannot apply truncation immediately. We must wait until all the important dimensions have been identified. When this occurs $\mathbf{u}$ is said to be complete. We can use the active dimension coordinate index set when $\mathbf{u}$ is complete. Here we deem $\mathbf{u}$ complete when every dimension, whose coordinate index is in $\mathbf{u}$, has at least one associated interval with a length δ.

To further reduce the number of points needed to characterise a discontinuity we must prevent refinement on unimportant boundaries. A boundary is unimportant if the ith coordinate $i = 1,\dots,d$ is equal to -1 or 1 and $i \notin \mathbf{u}$. Fig. 7 illustrates how the sparse grid detection Algorithm 5 refines boundaries of unimportant dimensions, the top and bottom faces of the cube.

Once truncation is invoked the missing ancestors of an active point are no longer added. These points are necessary when refining in a top-down fashion like that of Algorithm 4 but are not necessary when discontinuity tracking is used concurrently. However it must be noted that because missing ancestors are no longer added, situations may arise when the local stencil used by the polynomial annihilation method contains less than the required three points. In such cases the missing points must be added. Denote the d-dimensional coordinate of the child being assessed by $\mathbf{x}$ and let i be the coordinate index of the dimension along which the polynomial annihilation is begin applied. Add any of the following points which are not in the grid $(x_1,\dots,x_{i-1},-1,x_{i+1},\dots,x_d)$, $(x_1,\dots,x_{i-1},0,x_{i+1},\dots,x_d)$ and $(x_1,\dots,x_{i-1},1,x_{i+1},\dots,x_d)$ The algorithm is complete when no active points and no new discontinuity points can be found.

Fig. 8 shows four steps of the high-dimensional discontinuity tracking algorithm applied to Example 3.2 with $k = d = 2$, $r = 0.5$ and $\delta = 1/16$. Fig. 9 shows the final grid when HDT is applied to Example 3.2 with $k = 2$, $d = 3$, $r = 0.5$ and $\delta = 1/16$. The number of points in this grid, 379, is smaller than the number of points in the grid resulting from sparse grid detection. We also see that points are no longer wasted on the top and bottom faces of the cube, that is, on the unimportant boundaries.

Note that the proposed method does not guarantee that the existence of a discontinuity will be determined. If a discontinuity is not found by the computational algorithm the resolution of the initial grid was too coarse. If information on the existence of the discontinuity becomes available a posteriori, that is a location though which the discontinuity passes through is found, then the proposed discontinuity tracking algorithm can be used to determine track the discontinuity throughout the entire domain. This information can be obtained by reducing the size of the computation domain to an area which the modeller believes the discontinuity to exist or by using a higher level initial sparse grid to initiate the algorithm.

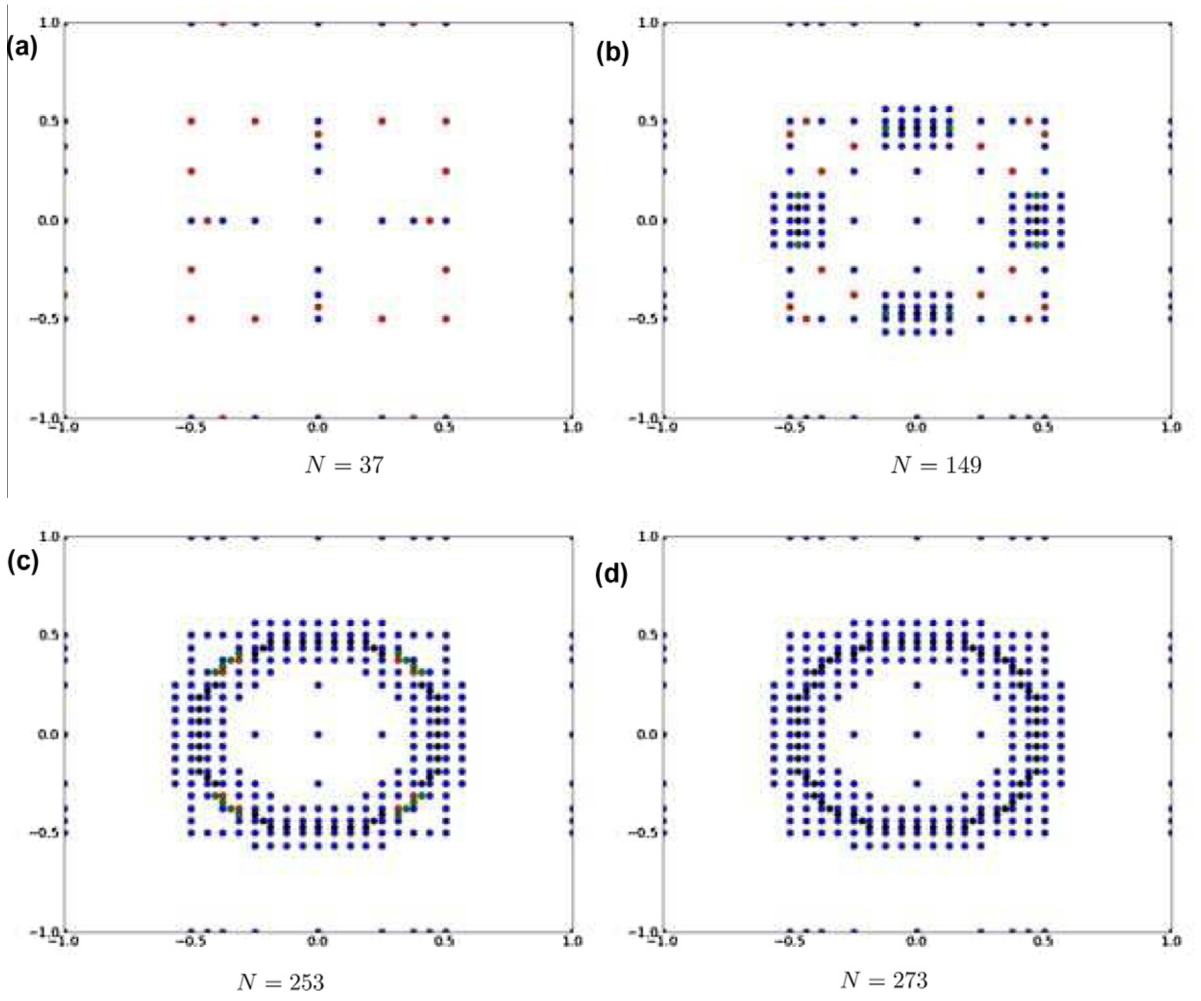


Fig. 8. (a) adaptive grid with the limit of resolution set to $\delta = 2^{-4}$ at level 5; (b) adaptive grid at level 7; (c) adaptive grid at level 9; (d) final grid. Here discontinuity points $\mathcal{E}$ are displayed in black, discontinuity points to be used for tracking are shown in green, active sparse grid points $\mathcal{A}$ are displayed in red, and points in the sparse grid $\mathcal{S}$ are displayed in blue. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

6. Post-processing

6.1. Estimating distance functions

The distance function

$$\mathrm{dist}_{\mathcal{E}}(\mathbf{x}) = \inf_{\mathbf{x}_{\mathcal{E}} \in \mathcal{E}} \|\mathbf{x} - \mathbf{x}_{\mathcal{E}}\|_2, \tag{26}$$

measures the shortest distance from a point $\mathbf{x}$ to the surface of the discontinuity. Using the set of discontinuity points determined by the proposed algorithm the distance function to the nearest discontinuity can be approximated by

$$\mathrm{dist}_{\mathcal{E}}(\mathbf{x}) = \inf_{\mathbf{x}_{\mathcal{E}} \in \mathcal{E}} \left(\sum_{i \in \mathbf{u}} (\mathbf{x}_i - \mathbf{x}_{\mathcal{E},i})^2 \right)^{\frac{1}{2}}. \tag{27}$$

Here $\mathbf{u} \subseteq \{1, \ldots, d\}$ denotes the set of coordinate indices of the dimensions along which a discontinuity exists. In the following we will consider the convergence of the maximum error

$$\epsilon_{\max} = \max_{\mathbf{x} \in \mathcal{X}} \mathrm{dist}_{\mathcal{E}}(\mathbf{x}), \tag{28}$$

which is closely related to the Hausdorff distance of two curves. Here the test set $\mathcal{X}$ is a set of 1000 points uniformly sampled from the hypercube $[-1, 1]^d$.

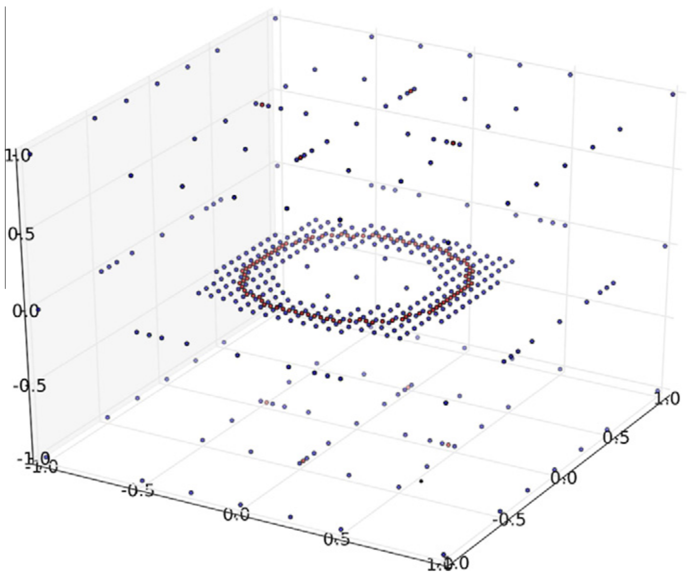


Fig. 9. HDT applied to Example 3.2 with $k = 2$, $d = 3$, $r = 0.5$ and $\delta = 1/16$.

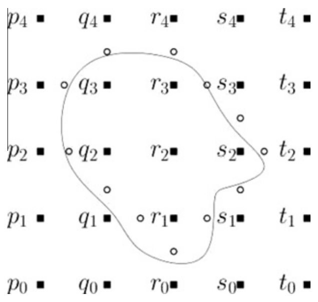


Fig. 10. A discontinuity in a two-dimensional domain. Grid points are represented using black boxes and discontinuity points by empty circles.

A slight improvement in the accuracy of (27) can be obtained by adding certain discontinuity points in the δ neighbourhood of previously identified discontinuity points. Specifically, if a discontinuity point $\mathbf{x}$ identifies a discontinuity in the ith direction, search each axial dimension $j \neq i$ for neighbouring discontinuity points. If a neighbouring discontinuity point $\mathbf{y}$ is a distance of 2δ away, i.e. $\mathbf{y} = (x_1,\dots,x_{j-1},x_j \pm 2\delta,x_{j+1},\dots,x_d)$, add the point $(x_1,\dots,x_{j-1},x_j \pm \delta,x_{j+1},\dots,x_d)$ to $\mathcal{E}$. In Fig. 11 these points correspond to the discontinuity points with no associated interval points.

6.2. Domain classification

The sparse grid discontinuity detection method can be used to classify the computational domain into its continuous subdomains. Once the points in the grid have been classified, any number of existing classification techniques can be used to construct a classification function to make predictions for new, yet unknown, data.

Here we present a modification of the classification algorithm presented in [11]. The objective of this algorithm is to classify each grid point into an appropriate class. 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. The points are classified iteratively. The algorithm starts by assigning the root point $(0,\dots,0)$ a class value of 1. We then proceed to classify its neighbouring points in each active axial direction $i \in \mathbf{u}$ to the left and to the right. Recall a dimension is deemed active if a discontinuity was found in that direction during the evolution of the discontinuity detection algorithm. If a discontinuity point exists between a point and its neighbour, the neighbour remains unclassified and the search in that direction is terminated. As each point is classified its neighbours are determined and then classified. This procedure is repeated until every point in Class 1 has been classified. This occurs when the algorithm stops generating new points for classification. The algorithm then proceeds to search for points in Class 2. A point of the yet unclassined points, which has non-zero coordinates only in the active dimensions, is marked as belonging to Class 2. The same search procedure used to classify points in Class 1 is then used to find and classify neighbours of points in Class 2. After all points in Class 2 have been classified we check to see if any unclassified points remain. If so, we choose and assign a point to Class 3 and continue otherwise until the classification is complete. The pseudo-code for this procedure is outlined in Algorithm 6.

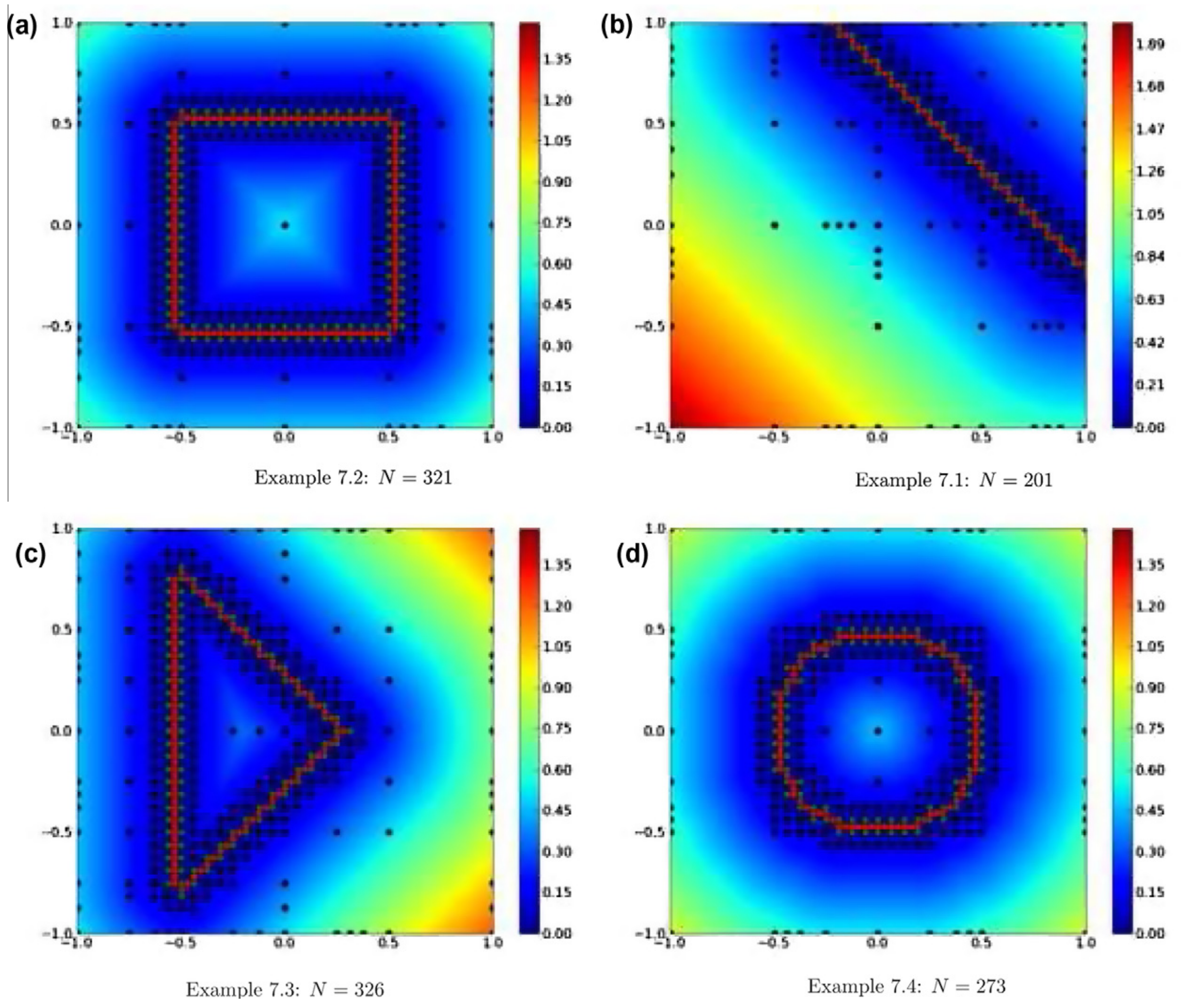


Fig. 11. Estimated distance functions, determined from Algorithm 2 in conjunction with Algorithm 3, for $d = 2$ and $\delta = 2^{-4}$, as it is applied to Example 7.1 (a), Example 7.2 (b), Example 7.3 (c), and Example 7.4 with $k = d = 2$ (d). Black points represent points in the sparse grid $\mathcal{S}$, red points represent discontinuity points $\mathcal{E}$, and the green points the intervals on which the discontinuity points are defined. Discontinuity interval points are also in $\mathcal{S}$.

Algorithm 6. Classify()

```
k = 1;
$\mathcal{U} := \mathcal{S}$
  **for** $(\mathbf{x} \in \mathcal{U})$ **do**
    $\mathcal{N} := \{\mathbf{x}\}$;
    $\mathcal{U} := \mathcal{U} \setminus \{\mathbf{x}\}$;
    **while** $(\mathcal{N} \neq \emptyset)$ **do**
      Class $(\mathbf{x})=k$;
      **for** $(\mathbf{x} \in \mathcal{N})$ **do**
        **for** $(\text{dir} \in \{1,\ldots,d\})$ **do**
          $\mathbf{z} :=$ FindLeftDiscontinuityPt $(\mathbf{x}, \text{dir})$;
          $L :=$ FindLeftNeighboursUpToDiscontinuity $(\mathbf{x}, \mathbf{z}, \text{dir})$;
          $\mathbf{z} :=$ FindRightDiscontinuityPt $(\mathbf{x}, \text{dir})$;
          $R :=$ FindRightNeighboursUpToDiscontinuity $(\mathbf{x}, \mathbf{z}, \text{dir})$;
          $\mathcal{N} := \mathcal{N} \cup (L \cup R)$
        **end for**
      **end for**
    **end while**
    $k = k + 1$;
**end for**
```

Note that not all points in the grid will be classified. Only the grid points in the hyperplane of the active dimensions are considered. The unclassified points can be discarded. To classify a new data point only the coordinates of the active dimensions will determine how the point is classified.

Without loss of generality consider the discontinuity, shown in Fig. 10, that divides the two-dimensional domain into two mutually exclusive regions. The same procedure can be extended to higher dimensions and multiple function classes. We pick a first point from the list of all unclassified points, say the point p_0, and assign it a class value 1. We consider all the points to the right of p_0, i.e., the points along the straight line $p_0 - t_0$, consisting of the group of points $\{q_0, r_0, s_0, t_0\}$. There are no discontinuities 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 $p_0 - p_4$, i.e., the group $\{p_1, p_2, p_3, p_4\}$. Once again every point in this group gets classified as Class 1. Nine of a total of 25 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 p_0. For instance, considering the vertical direction of the point q_0 would determine that the point q_1 belongs to the first class. The point q_2 does not belong to Class 1 because it is separated from q_1 by a discontinuity point (represented by an empty circle). Considering the horizontal direction of the point q_0 would determine that the points $\{p_0, r_0, s_0, t_0\}$ belong to Class 1. Naturally there will be points that are classified more than once, but they will be classified into the same class. Therefore this algorithm need only be concerned with the first classification.

Once all the grid points have been classified, any number of classification algorithms can be used to construct a classification function to make predictions for new, yet unknown, data. Support vector machines [32] and sparse grid classification [33,34] are two such techniques. Principal manifold learning [35] could also be used to construct an approximation of the lower-dimensional manifold. In this paper we use a simple heuristic to classify points. The class of a new point is simply the class of its nearest neighbour in the grid. This will incorrectly classify points in the δ neighbourhood of the discontinuity but this is also a failing of the aforementioned methods. We remark again that the inactive dimensions have no influence on the classification of new data points. The discontinuity detection Algorithm 5 is a dimension reduction technique that allows important, active dimensions to be identified and the unimportant dimensions to be ignored. Subsequently any post-processing procedure, be it classification or estimating distances to a discontinuity, can be performed on the lower-dimensional latent space.

7. Numerical results

In this section we provide several numerical tests to illustrate the implementation and convergence of the proposed methodology. While several examples are presented in earlier sections, here we focus on the performance of the algorithms in high-dimensional spaces.

7.1. Benchmark problems

We first consider the following examples.

Example 7.1 (*Square*).

$$f(\mathbf{x}) = \begin{cases} 1, & |x_1| < \frac{1}{2} \quad \text{and} \quad |x_2| < \frac{1}{2}, \\ 0, & \text{otherwise}. \end{cases} \tag{29}$$

Example 7.2. Plane

$$f(\mathbf{x}) = \begin{cases} 1, & x_1 + x_2 > \frac{3}{4}, \\ 0, & \text{otherwise}. \end{cases} \tag{30}$$

Example 7.3. Triangle

$$f(\mathbf{x}) = \begin{cases} 0, & x_1 + x_2 > \frac{1}{4} \quad \text{and} \quad x_1 - x_2 > \frac{1}{4} \quad \text{and} \quad x_1 < -0.5, \\ 1, & \text{otherwise}. \end{cases} \tag{31}$$

Example 7.4. Sphere

$$f(\mathbf{x}) = \begin{cases} 1, & \sum_{i=1}^{k} x_i^2 < r^2, \quad k \leqslant d, \\ 0, & \text{otherwise}. \end{cases} \tag{32}$$

Fig. 11 plots the final grid generated by Algorithm 5 and the associated approximate distance function for Examples 7.1, 7.2, 7.3, 7.4.

First let us investigate the error of the approximated distance function using the HDT method, as δ decreases. Table 2 shows the number of points needed to capture the discontinuity present in Example 7.4 with $k = 3 \leqslant d$ and $r = 1/8$ for varying dimensionality d of the input space. The number of points required increases only linearly with the dimensionality of the input space. The vast majority of points are invested in the active dimensions. A graphical illustration of the linear increase in the number of points with dimension is shown in Fig. 12.

Table 3 displays the number of points required by the HDT method as δ decreases, when applied to Example 7.4 with $k = 3$ and $d = 20$. The error in the distance function is maintained at $O(\delta)$.

Fig. 13 (a) plots the $\epsilon_{\max}$ error of the HDT method when applied to Example 7.4 with $k = 3$ and $d = 20$. To illustrate the efficiency of the HDT method we also plot the error of the SGDD method and the truncated SGDD method, TSGDD. For all

Table 2
$\epsilon_{\max}$ error in the approximated distance function of the 3-dimensional spherical discontinuity as the dimensionality of the input space is increased for Example 7.4 with $k = 3 \leqslant d$ and $r = 1/8$.

d	N	$\epsilon_{\max}$
5	573	0.03206
10	883	0.03160
15	1193	0.03156
20	1503	0.03075
25	1813	0.03055
50	3363	0.03137
100	6463	0.03128

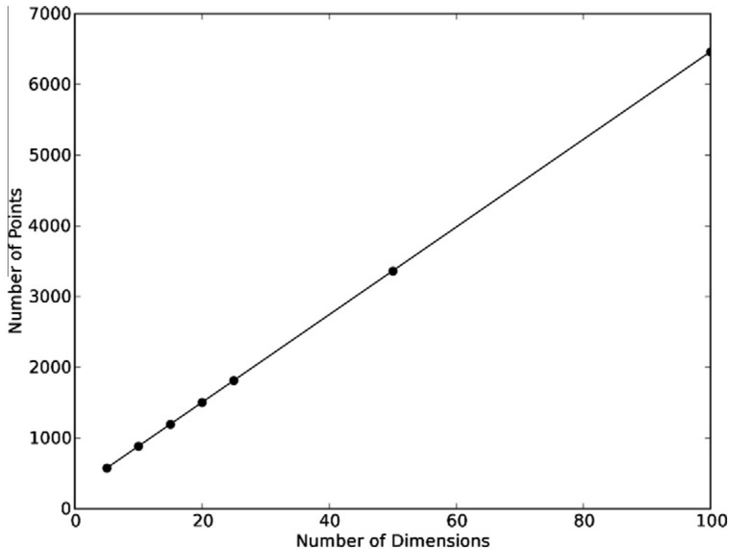


Fig. 12. Example 7.4 with $k = 3 \leqslant d$ and $r = 1/8$ for varying dimensionality d of the input space. Here the discontinuity resides on a small subset of the dimensions and the total number of points required grows linearly with the dimensionality.

Table 3
$\epsilon_{\max}$ error in the approximated distance function as δ is decreased for the 3-dimensional spherical discontinuity embedded in a larger input space, Example 7.4 with $k = 3 \leqslant d$ and $r = 1/8$.

δ	N	$\epsilon_{\max}$
$\frac{1}{8}$	1027	0.09112
$\frac{1}{16}$	1503	0.03164
$\frac{1}{32}$	2513	0.01577
$\frac{1}{64}$	17,747	0.007779
$\frac{1}{128}$	37,633	0.003872

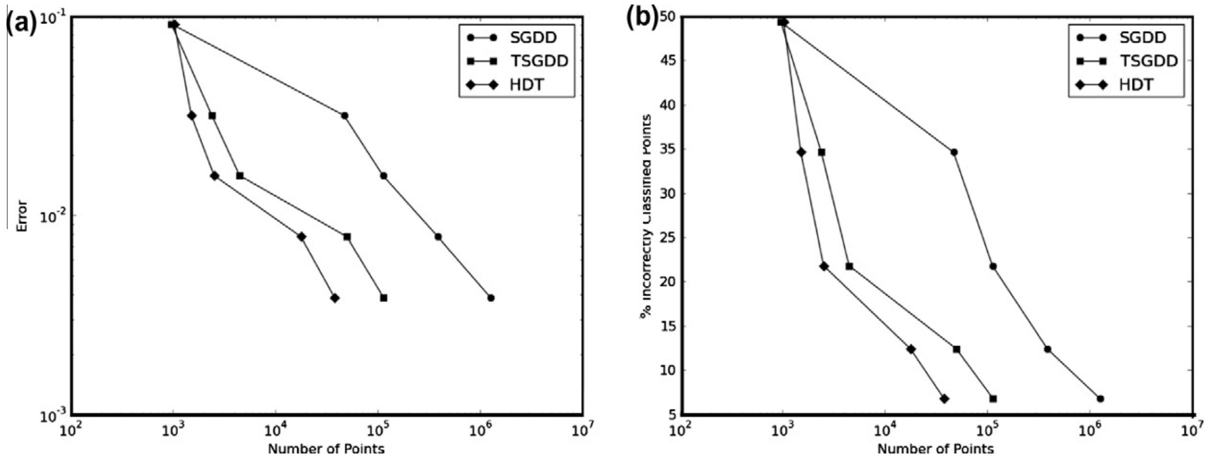


Fig. 13. (a) the $\epsilon_{\max}$ error of the HDT, TSGDD, and SGDD methods when applied to Example 7.4 with $k = 3$ and $d = 20$; (b) the percentage of new points that are classified incorrectly as the size of δ decreases.

methods the error in the approximated distance function decreases with δ. The HDT method is by far the most efficient. TSGDD is the second-most efficient method and SGDD requires the most amount of points to characterise the discontinuity.

Now we analyse the ability of the three methods to classify a set of randomly generated data points. Here we randomly generate 1000 points that lie a maximum distance of 0.125 away from the true discontinuity and classify each point according to the classification algorithm presented in Section 6.2. Fig. 13(b) plots the percentage of new points that are classified incorrectly as the size of δ decreases. Again HDT is the most efficient method, followed by the TSGD and SGD methods. Note that if 1000 sample points were simply generated uniformly in $[-1,1]^{20}$, all three methods classify all the points correctly.

7.2. Multidimensional stochastic application

We now consider a stochastic differential-algebraic system of equations

$$\begin{aligned} \frac{du}{dt} &= \frac{\alpha_1}{1+v^{\beta}} - u, \\ \frac{dv}{dt} &= \frac{\alpha_2}{1+w^{\gamma}} - v, \\ w &= \frac{u}{(1+[\mathrm{IPTG}]/K)^{\eta}}, \end{aligned} \tag{33}$$

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* [15]. Stochastic simulation of this system was conducted in [23], where the parameters $p = (p_1,\ldots,p_6) = (\alpha_1,\alpha_2,\beta,\gamma,\eta,K)$ are modelled as random variables in the form of $p = \langle p\rangle(1+\sigma y)$, where $\langle p\rangle = (156.25, 15.6, 2.5, 1, 2.0015, 2.9618\times10^{-5})$ are the mean values and $y = (y^{(1)},\ldots,y^{(6)})$ are random variables uniformly distributed in $[-1,1]^6$.

Provided the duration of the simulation of the genetic toggle is long enough (the exact length depends on the values of the input parameters), the system will reach one of only two possible steady states. The steady state reached during a simulation depends on the intensity of the input [IPTG]. The "on" and "off" switch type behavior suggests that the solution, 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. A discontinuity detection method such as the one proposed in this manuscript can be used to identify which parameters and combinations thereof cause a jump in behaviour.

Here we utilise the HDT method to classify the input space of the genetic toggle model into regions that produce "on" behaviour and regions that produce "off" behaviour. "On" behaviour is defined to occur when the system reaches the steady state $v \geqslant 0$ and "off" behaviour refers to simulations that reach the alternative steady state. An associated aim of this section is to provide a comparison of the adaptive-domain classification method proposed by Archibald et al. [11] with the HDT method. To ensure consistency with the classification procedure used by Archibald et al. we carry out discontinuity detection for $v(y)$ in four dimensions neglecting the influence of β and γ which are fixed at their nominal values.

Using the high-dimensional discontinuity tracking method Algorithm 5, 31,104 points were needed to capture the discontinuity to a resolution of $\delta = 2^{-4}$. Applying the classification procedure outlined in Section 6.2 to 1000 point randomly generated in $[-1,1]^4$, 98.1% of points were classified correctly. The computational cost of the Algorithm 5 is better than the adaptive-domain classification method [11], where the number of nodes in the HDT generated grid was 31,379 as compared to 91,250 grid points for the adaptive-domain classification method.

The numerical cost of the proposed algorithm is dominated by the computational effort required to evaluate the function at each of the points in the adaptive grid. When the time required to evaluate the function is even moderately large (greater than 1 s), the cost of applying the polynomial annihilation method to detect discontinuities, and the adaptive method for determining the necessary stencils, is negligible.

Here we have limited our attention to accurately decomposing the input domain into regions of high-regularity. Combining the proposed discontinuity detection method with multi-element interpolation to facilitate a more traditional uncertainty quantification analysis is a complex task and is beyond the scope of this manuscript. Once the domain has been decomposed into regions of high-regularity an appropriate high-order interpolation method must be used to interpolate in the smooth regions. Currently the majority of such methods can only be applied to hyper-rectangular regions. Other methods that do not suffer from this limitation exist, however these methods, in their current form, are not well suited to high-dimensional interpolation. Further refinement of these methods is necessary before they can be used to solve the problem at hand. The development of a high-order, high-dimensional interpolation method that can be applied to arbitrarily shaped regions is the subject of future investigations.

8. Conclusions

In this paper we present a method for identifying and locating discontinuities in high-dimensional spaces. The method extends, and improves significantly, the previous algorithms using polynomial annihilation for detecting discontinuities in low dimensions.

The proposed discontinuity detection method proposed is closely related to adaptive sparse grid approximation. A locally adaptive approach is used to determine and resolve the location of any discontinuities. The method only refines in dimensions deemed to contribute to the location of those discontinuity, thereby spending more effort in regions which likely contain a discontinuity. The polynomial annihilation method is used to guide refinement that invests the majority of function evaluations in regions surrounding discontinuities and neglects the grid points in the smooth regions.

Upon detecting the discontinuities, the adaptive sparse grid approach is further combined with a tracking scheme to efficiently capture the location and geometry of the discontinuities which often reside on a subset of the dimensions. The tracking scheme uses local tensor product grids around known edge points to determine neighbouring edge points. This process is performed iteratively, thereby efficiently investing points along the discontinuity and out from the original edge points.

The grid generated by the discontinuity method can be used to classify the computational domain into its continuous subdomains. Once the points in the grid have been classified, any number of existing classification techniques can be used to construct a classification function to make predictions for new, yet unknown, data. We developed a simple heuristic that classifies new data based upon the classification of the nearest grid point. Such a procedure is shown to provide an accurate classification function. The discontinuity grid can also be used as a distance function that calculates the distance of a new data point from a discontinuity.

The accuracy of the classification and distance functions is proportional to the resolution δ used to resolve the discontinuity. The number of grid points is proportional to the dimension and hyper-dimensional surface area of the functional edge. Analogously, for fixed surface area the number of points required is inversely proportional to the resolution δ used to resolve the discontinuity. When the dimensionality of the discontinuity resides in a small subset of dimensions of the input space the proposed method is very efficient. In these situations the algorithm becomes "optimal," that is the total number of points required grows linearly with the dimensionality.

Acknowledgments

The submitted manuscript has been authored by contractors [UT-Battelle LLC, manager of Oak Ridge National Laboratory (ORNL)] of the US Government under Contract No. DE-AC05-00OR22725. Accordingly, the US 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 US Government purposes. Dongbin Xiu was partially supported by AFOSR FA9550-08-1-0353, DOE/NNSA DE-FC52-08NA28617, and NSF CAREER DMS-0645035 and IIS-0914447.

References

[1] R. Ghanem, P. Spanos, Stochastic Finite Elements: A Spectral Approach, Springer-Verlag, New York, Inc.,, New York, NY, USA, 1991.
[2] D. Xiu, G. Karniadakis, The Wiener–Askey polynomial chaos for stochastic differential equations, SIAM J. Sci. Comput. 24 (2) (2002) 619–644. http://dx.doi.org/10.1137/S1064827501387826.
[3] D. Xiu, Fast numerical methods for stochastic computations: a review, Commun. Comput. Phys. 5 (2009) 242–272.
[4] J. Jakeman, M. Eldred, D. Xiu, Numerical approach for quantification of epistemic uncertainty, J. Comput. Phys. 229 (12) (2010) 4648–4663, doi:10.1016/j.jcp.2010.03.00. http://www.sciencedirect.com/science/article/B6WHY-4YM7FRC-1/2/4dd64b16ceaef199ff2c8545106ea6e6.
[5] D. Gottlieb, D. Xiu, Galerkin method for wave equations with uncertain coefficients, Commun. Comput. Phys. 3 (2) (2008) 505–518.
[6] O. Le Maitre, O. Knio, H. Najm, R. Ghanem, Uncertainty propagation using Wiener–Haar expansions, J. Comput. Phys. 197 (1) (2004) 28–57. http://dx.doi.org/10.1016/j.jcp.2003.11.033.
[7] X. Ma, N. Zabaras, An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations, J. Comput. Phys. 228 (2009) 3084–3113.

[8] X. Wan, G. Karniadakis, Multi-element generalized polynomial chaos for arbitrary probability measures, SIAM J. Sci. Comput. 28 (3) (2006) 901–928. http://dx.doi.org/10.1137/050627630.
[9] I. Babuska, R. Tempone, G. Zouraris, Galerkin finite element approximations of stochastic elliptic partial differential equations, SIAM J. Numer. Anal. 42 (2) (2004) 800–825, doi:10.1137/S003614290241868. http://link.aip.org/link/?SNA/42/800/1.
[10] J. Foo, X. Wan, G. Karniadakis, The multi-element probabilistic collocation method (ME–PCM): error analysis and applications, J. Comput. Phys. 227 (22) (2008) 9572–9595.
[11] R. Archibald, A. Gelb, R. Saxena, D. Xiu, Discontinuity detection in multivariate space for stochastic simulations, J. Comput. Phys. 228 (7) (2009) 2676–2689.
[12] R. Archibald, J. Gelb, A. Yoon, Polynomial fitting for edge detection in irregularly sampled signals and images, SIAM J. Numer. Anal. 43 (1) (2005) 259–279.
[13] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Machine Intell. 8 (1986) 679–698.
[14] I. Sobel, An isotropic 3 image gradient operator, in: H. Freeman (Ed.), Machine Vision for Three-Dimensional Scenes, Academic Pres, Boston, 1990.
[15] T. Gardner, C. Cantor, J. Collins, Construction of a genetic toggle switch in *Escherichia coli*, Nature 403 (2000) 339–342.
[16] D. Xiu, J. Hesthaven, High-order collocation methods for differential equations with random inputs, SIAM J. Sci. Comput. 27 (3) (2005) 1118–1139.
[17] V. Barthelmann, E. Novak, K. Ritter, High dimensional polynomial interpolation on sparse grids, Adv. Comput. Math. 12 (2000) 273–288.
[18] T. Gerstner, M. Griebel, Numerical integration using sparse grids, Numer. Algor. 18 (3–4) (1998) 209–232.
[19] T. Gerstner, M. Griebel, Dimension-adaptive tensor-product quadrature, Computing 71 (1) (2003) 65–87, doi:10.1007/s00607-003-0015-5.
[20] I. Babuska, F. Nobile, R. Tempone, A stochastic collocation method for elliptic partial differential equations with random input data, SIAM J. Numer. Anal. 45 (3) (2007) 1005–1034, doi:10.1137/050645142.
[21] B. Ganapathysubramanian, N. Zabaras, Modeling diffusion in random heterogeneous media: data-driven models stochastic collocation and the variational multiscale method, J. Comput. Phys. 226 (1) (2007) 326–353, doi:10.1016/j.jcp.2007.04.009.
[22] F. Nobile, R. Tempone, C. Webster, An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data, SIAM J. Numer. Anal. 46 (5) (2008) 2411–2442, doi:10.1137/070680540. http://link.aip.org/link/?SNA/46/2411/1.
[23] D. Xiu, Efficient collocation approach for parametric uncertainty analysis, Commun. Comput. Phys. 2 (2) (2007) 293–309.
[24] N. Zabaras, B. Ganapathysubramanian, A scalable framework for the solution of stochastic inverse problems using a sparse grid collocation approach, J. Comput. Phys. 227 (9) (2008) 4697–4735. http://dx.doi.org/10.1016/j.jcp.2008.01.019.
[25] R. Bauer, Band pass filters for determining shock locations, Ph.D. thesis, Applied Mathematics, Brown University, 1995.
[26] 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.
[27] R. Archibald, J. Gelb, A. Yoon, Determining the locations of discontinuities in the derivatives of functions, Appl. Numer. Math. 58 (5) (2008) 577–592.
[28] M. Griebel, Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences, Computing 61 (2) (1998) 151–179. http://dx.doi.org/10.1007/BF0268441.
[29] S. Smolyak, Quadrature and interpolation formulas for tensor products of certain classes of functions, Soviet Math. Dokl. 4 (1963) 240–243.
[30] X. Wang, I. Sloan, Why are high-dimensional finance problems often of low effective dimension, SIAM J. Sci. Comput. 27 (2005) 159–183.
[31] M. Griebel, M. Holtz, Dimension-wise integration of high-dimensional functions with applications to finance, J. Complex. 26 (5) (2010) 455–489, doi:10.1016/j.jco.2010.06.001. http://www.sciencedirect.com/science/article/B6WHX-50F8BT9-1/2/b1c278f2e0e3ad9714b17aee72497c74.
[32] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, Cambridge, United Kingdom, 200.
[33] J. Garcke, M. Griebel, M. Thess, Data mining with sparse grids, Computing 67 (2001) 225–253.
[34] M. Hegland, J. Garke, V. Challis, The combination technique and some generalisations, Linear Algebra Appl. 420 (2007) 249–275.
[35] C. Feuersanger, M. Griebel, Principal manifold learning by sparse grids, Computing 85 (4) (2009) 267–299. http://dx.doi.org/10.1007/s00607-009-0045-8.