A nonintrusive reduced order modelling approach using Proper Orthogonal Decomposition and locally adaptive sparse grids

Alsayyari, Fahad; Perkó, Zoltán; Lathouwers, Danny; Kloosterman, Jan Leen

DOI
10.1016/j.jcp.2019.108912

Publication date
2019

Document Version
Final published version

Published in
Journal of Computational Physics

Journal of Computational Physics 399 (2019) 108912

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A nonintrusive reduced order modelling approach using Proper Orthogonal Decomposition and locally adaptive sparse grids

Fahad Alsayyari *, Zoltán Perkó, Danny Lathouwers, Jan Leen Kloosterman

Delft University of Technology, Faculty of Applied Sciences, Department of Radiation Science and Technology, Mekelweg 15, Delft, 2629JB, the Netherlands

ARTICLE INFO

Article history:
Received 4 April 2019
Received in revised form 30 July 2019
Accepted 23 August 2019
Available online 28 August 2019

Keywords:
Proper Orthogonal Decomposition
Reduced order modelling
Adaptive sparse grids
Locally adaptive

ABSTRACT

Large-scale complex systems require high fidelity models to capture the dynamics of the system accurately. The complexity of these models, however, renders their use to be expensive for applications relying on repeated evaluations, such as control, optimization, and uncertainty quantification. Proper Orthogonal Decomposition (POD) is a powerful Reduced Order Modelling (ROM) technique developed to reduce the computational burden of high fidelity models. In cases where the model is inaccessible, POD can be used in a nonintrusive manner. The accuracy and efficiency of the nonintrusive reduced model are highly dependent on the sampling scheme, especially for high dimensional problems.
To that end, we study integrating the locally adaptive sparse grids with the POD method to develop a novel nonintrusive POD-based reduced order model. In our proposed approach, the locally adaptive sparse grid is used to adaptively control the sampling scheme for the POD snapshots, and the hierarchical interpolant is used as a surrogate model for the POD coefficients. An approach to efficiently update the surpluses of the sparse grids with each POD snapshots update is also introduced. The robustness and efficiency of the locally adaptive algorithm are increased by introducing a greediness parameter, and a strategy to validate the reduced model after convergence. The proposed validation algorithm can also enrich the reduced model around regions of detected discrepancies. Three numerical test cases are presented to demonstrate the potential of the proposed POD-Adaptive algorithm. The first is a nuclear reactor point kinetics, the second is a general diffusion problem, and the last is a variation of the analytical Morris function. The results show that the developed algorithm reduced the number of model evaluations compared to the classical sparse grid approach. The built reduced models captured the dynamics of the reference systems with the desired tolerances. The non-intrusiveness and simplicity of the method provide great potential for a wide range of practical large scale applications.

1. Introduction

Complex systems are described by interactions of multi-physics phenomena that occur at various scales. Capturing such inter-disciplinary interactions requires building complex coupled models. Nuclear reactors, for example, are described by interactions of radiation transport, heat transfer, and fluid dynamics. High fidelity simulation tools are often used to provide

* Corresponding author.
E-mail address: f.s.s.alsayyari@tudelft.nl (F. Alsayyari).

https://doi.org/10.1016/j.jcp.2019.108912
0021-9991/

a solution for such coupled problems. Nevertheless, even with the increasing power of today's supercomputers, high fidelity models require a tremendous amount of computational time and memory allocation. For applications such as design optimization, uncertainty quantification, and control, where many repeated model evaluations are needed, such models are too expensive.

Reduced Order Modelling (ROM) is an effective technique to reduce the dimensionality of large-scale complex systems. It replaces the high fidelity model with a low-dimensional efficient model preserving only the prominent dynamics of the system. The reduced model can then be used to provide fast solutions with a controlled level of accuracy. Different reduced order modelling techniques have been proposed in literature. Benner et al. (2015) [1] presented a comprehensive survey on ROM with numerous examples on their applications. Proper Orthogonal Decomposition (POD) is the favoured method for nonlinear systems [2]. POD was first introduced as a statistical technique to extract dominant characteristics from a set of data. As a reduced order method, the method was later developed by Lumley (1967) [3] to model coherent structures in turbulent flows.

The POD method is based on sampling the high fidelity model at several points in parameter space to construct a so-called snapshot matrix. Then, a reduced basis is created through Singular Value Decomposition (SVD). The original high fidelity model can then be projected onto the created reduced basis space by means of a Galerkin projection. The snapshot matrix and the model are built during the offline phase, which is executed only once. Afterwards, during the online phase, the reduced model can be run inexpensively at any desired parameter point. POD-Galerkin has been studied extensively in many fields, such as in fluid dynamics to model compressible flows [4–6] and incompressible flows [7–9]. In reactor physics applications, POD was applied to the eigenvalue problem of the diffusion equation [10], the time-dependent diffusion equation [11], and to model the coolant pool of the Lead-cooled Fast Reactor [12]. A variant of this approach called the Reduced Basis method [13–15] was developed by adding an a posteriori error estimator and a greedy sampling scheme to model parametrized partial differential equations.

Projection-based POD methods are code intrusive, which is a major limitation. For legacy codes where access to the governing equations is unattainable, this approach cannot be applied. In such cases, a slightly different nonintrusive POD technique can be employed. The idea is to benefit from the orthogonality of the subspace basis to generate the Galerkin expansion coefficients at the sampled points. Then, a surrogate model can be constructed to compute the solution at any required non-sampled point. Different surrogate models have been suggested to compute the expansion coefficients. For lower dimensional problems, direct interpolation or splines can be used [16]. Problems with higher dimensionality require more advanced techniques. The use of Radial Basis Function (RBF) is one of the common methods for such problems [2,17]. Hesthaven and Ubbiali (2018) [18] used neural networks to build a surrogate model for the coefficients. Gaussian regression can also be used to build the surrogate model [19].

For the ROM to produce an accurate representation of the original model, the snapshots need to capture the entire dynamics of the original model within the desired range. Consequently, the choice of the sampling scheme directly affects the accuracy of the POD method. Moreover, in the nonintrusive approach, the sampling points should be dense enough for the surrogate model to reproduce a reliable solution at non-sampled locations. Thus, the sampling strategy becomes even more relevant for nonintrusive methods. In addition, problems parametrized on high dimensional spaces are prone to the curse of dimensionality, that is the exponential increase of the computational time with the increase in the number of dimensions. In these cases, the efficient selection of the sampling points is crucial for any practical application. Latin Hypercube Sampling (LHS) can be an efficient way to address this issue [20]. However, the lack of adaptivity is a limitation in nonlinear cases. Guenot et al. (2013) [21] proposed an extension of LHS that improves the initial snapshot matrix by adaptively selecting new points based on the "influence" of the new point on the snapshot matrix.

A different sampling method based on sparse grids can also be applied. Sparse grids were first introduced by Smolyak (1963) [22] and, ever since, have been used to cope with high dimensional multivariate integration and interpolation problems. The method builds a hierarchical grid that preserves the properties for the unidimensional rule by a specific combination of the tensorized product [23]. In the context of reduced order models, Peherstorfer (2013) [24] suggested the use of sparse grids as a machine learning tool for reduced order models, which was tested on heat transfer problems. In addition, Xiao et al. (2015) [25] presented a method of propagating the expansion coefficients through time with the use of a sparse grid interpolant, which was demonstrated on the Navier-Stokes equations. Elman et al. (2011) [26] suggested the use of Reduced Basis method to further reduce the computational burden of stochastic collocation methods based on sparse grids.

Sparse grids were developed under the assumption that the function to be approximated is sufficiently smooth. In this case, the algorithm provides optimal selection of subspaces contributing to the interpolant function. However, in many applications, the smoothness of the function is unknown and cannot be established a priori. To that end, adaptive strategies can be employed to modify the classical sparse grids algorithm. In cases where the multivariate function is only sensitive to certain dimensions, the anisotropic sparse grid approach is suitable. In this strategy, which is also called (global) adaptive sparse grids, the grid is constructed by placing more points along certain dimensions that have higher importance. The importance of each dimension is identified during the construction stage by testing and comparing all dimensions [27]. Chen and Quarteroni (2015) [28] combined the anisotropic adaptive sparse grid with the reduced basis method for error estimation.

However, the dimension adaptive approach falls short of identifying regions with steep gradients or discontinuities. For these cases, an alternative local adaptive strategy can be more effective. In fact, one of the earliest work on sparse grids

by Zenger (1990) [29] suggested the use of local adaptivity for non-smooth functions. The objective of this strategy is to identify certain regions of higher importance and only refine the grid within these regions by benefiting from the hierarchical structure of the grids [30]. Griebel (1998) [31] showed how locally adaptive sparse grids can be used to adaptively discretize partial differential equation. In this implementation, Dirichlet boundary condition is assumed, and the unidimensional rule is chosen to not place any points along the boundaries. Pflueger (2010) [32] extended this idea by modifying the basis functions in a way that extrapolates towards the boundaries. The author used this algorithm for classification problems. This approach is suitable if the value at the boundaries is not important and only an estimate is required. Ma and Zabaras (2009) [33] used a unidimensional rule that places points at the boundaries for an adaptive collocation method. The authors then used the algorithm with an Anchored-ANOVA approach to model stochastic processes [34]. Applications of the locally adaptive sparse grids can also be seen in tracking function discontinuities in high dimensional spaces [35], high dimensional integrations [36], and economic modelling [37].

In this paper, our goal is to exploit the hierarchical nature of the adaptive sparse grids in order to efficiently build a POD-based reduced order model in a nonintrusive manner. We present an approach to utilize the local adaptivity in order to identify regions of high importance for the POD development. In our nonintrusive approach, no assumption is made on the value of the model at the boundaries of the parameter domain. Therefore, we follow the work by Ma and Zabaras (2009) [33] in defining the unidimensional rule for the adaptive sparse grids. We also introduce an approach to iteratively update the surpluses of the sparse grids as the POD modes develop. We suggest a criterion for the refinement strategy based on the physical space rather than the surpluses of the sparse grids. We also extend the locally adaptive algorithm by introducing a parameter that controls the greediness of the algorithm in generating the snapshots. Additionally, a strategy to validate and update the reduced model is proposed, which increases the robustness of the algorithm. Although the nonintrusive approach is considered in this paper, the proposed algorithm can equally be combined with a Galerkin-POD approach.

The remainder of this paper is organized as follows: in Section 2 the problem is formulated and the POD method is introduced. Section 3 presents the sparse grids as an interpolation technique by first introducing the classical sparse grids and subsequently the locally adaptive version. In this section, the refinement strategy and the proposed validation algorithm are also presented. The combined POD-Adaptive algorithm is presented in Section 4 along with the method of updating the surpluses. Three applications are presented in Section 5 that test the proposed algorithm numerically. The first is a neutron point kinetics problem in 5 dimensions presented in two cases: one is strongly nonlinear and the second case is weakly nonlinear. The second application is a general diffusion problem in 18 dimensions, and the last application is an analytical function of 20 dimensions. Finally, our conclusions are discussed in Section 6.

2. Proper Orthogonal Decomposition

Physical phenomena are modelled by capturing the dynamics of the system in a set of governing equations. These equations can then be solved numerically by some discretization technique. The solution of the discretized model results in the state (or field) vector describing the state of the system, which in turn is a function of some design parameters that characterise features of the system (geometry, materials,...,etc.). The discretized model can be written in the form

$$\mathbf{R}(\mathbf{y}(\mathbf{x}), \mathbf{x}) = \mathbf{0}, \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^n$ is a vector with n state variables, and $\mathbf{x} \in \mathbb{R}^d$ is a vector of the design parameters with dimension d. For high fidelity models, the dimension of the state vector (n) is usually large, which renders solving the model to be computationally expensive. ROM aims at recasting the high fidelity model into a simpler model with dimension $r < n$. The model can then be solved with reduced computational effort. The POD method approximates the state vector in terms of basis vectors (in a discrete analogy to Fourier expansion) as

$$\mathbf{y}(\mathbf{x}) \approx \sum_{h=1}^{r} c_h(\mathbf{x})\mathbf{u}_h, \tag{2}$$

where c_h is the amplitude of the basis vector $\mathbf{u}_h$. The POD basis vectors (also called POD modes) are data-driven, that is they are built based on data collected from the model to describe the state vector. The amplitude c_h is a function of the design parameter $\mathbf{x}$.

The POD method is based on sampling the model at different design parameter values. Each state solution is a snapshot of the model at a certain parameter value. The snapshots are collected in the snapshot matrix

$$\mathbf{M} = [\mathbf{y}(\mathbf{x}_1), \mathbf{y}(\mathbf{x}_2), \ldots, \mathbf{y}(\mathbf{x}_p)] \quad \in \mathbb{R}^{n \times p}, \tag{3}$$

where p is the number of sampling points. The goal of the POD method is to find the optimal basis vectors in some subspace $\mathbb{V}$ of dimension $r << n$ that minimizes the error of the approximation in the L_2 norm. Once the basis vectors are known, the amplitude $c_h(\mathbf{x})$ can be computed either intrusively, by projecting the governing equations, or non-intrusively using regression methods. Our approach is non-intrusive, and therefore the model can be considered as a black box, mapping a given input to the desired output. We can write a functional minimizing the approximation error in the L_2 norm as follows [38]:

$$\min_{\mathbf{u}_h} E = \sum_{j=1}^{p} \left\| \mathbf{y}(\mathbf{x}_j) - \sum_{h=1}^{r} c_h(\mathbf{x}_j)\mathbf{u}_h \right\|_{L2}. \tag{4}$$

The basis vectors are chosen such that they are orthonormal (i.e., $< \mathbf{u}_g, \mathbf{u}_h >= \delta_{gh}$). The POD basis solves the minimization problem in Equation (4). They can be obtained with the Singular Value Decomposition (SVD) as the left singular vectors of the snapshot matrix [2]. Using orthogonality of the modes, the value of the amplitude $c_h(\mathbf{x})$ at parameter value $\mathbf{x}_j$ can be computed as

$$c_h(\mathbf{x}_j) =< \mathbf{u}_h, \mathbf{y}(\mathbf{x}_j) > . \tag{5}$$

If the number of non-zero singular values is g, it can be shown that the rank of the snapshot matrix is also g. The POD basis vectors are formed by the first r left singular vectors (where $r \leq g$). If r is chosen to be strictly less than g, a POD truncation error can be quantified using the singular values (σ) as follows:

$$e_r = \frac{\sum_{k=r+1}^{n} \sigma_k^2}{\sum_{k=1}^{n} \sigma_k^2}. \tag{6}$$

To set a criterion for selecting the number of POD modes (r), a cut-off threshold (γ_{tr}) can be defined such that

$$e_r < \gamma_{\text{tr}} \qquad \forall r \in [1, \ldots n]. \tag{7}$$

Note that the truncation error only quantifies the error in representing the state solutions included in the snapshot matrix. However, with sufficient sampling points, it can be used as an indicator for the error in representing (new) solutions not included in the snapshot matrix.

3. Sparse grids for interpolation

In order to generate the snapshot matrix, we need to explore the parameter space by sampling the model at discrete points. Choosing an appropriate number of sampling points is a key challenge for any sampling strategy. Covering the entire range of dynamics of the unknown function is imperative for a successful construction of the ROM. In addition, extending the sampling strategy to high dimensional problems is another challenge that must be addressed. Different sampling schemes have been studied to determine an optimal set of sampling points for increased space coverage (e.g., LHS). However, such methods select the sampling points a priori without any insight into the function being sampled. This can lead to overlooking some localized nonlinearities or discontinuities. To reduce such risk, uniform sampling with small intervals can be used. However, this strategy is prohibitively expensive.

Sparse grids can be very effective for problems of high dimensionality. The construction of the sparse grids is a hierarchical approach that successively builds the new grid based on previous grid selection. Such construction is suitable for adaptive strategies since it can be used to build an algorithm that will terminate whenever a desired accuracy is reached. We will first introduce the classical sparse grid method then present our approach for implementing adaptivity.

3.1. Classical sparse grids

Sparse grids are constructed based on selecting a set of nodes separately for each dimension. These nodes are generated in a hierarchical manner by levels, where each level is assigned an integer index i. The unidimensional nodes are then tensorized to form the final sparse grids. We first consider one dimension, then generalize it to the multidimensional case.

Many choices are possible for selecting the unidimensional nodes. While the nodes can be disjoint (non-nested) as in [26] and [39], nested nodes are more convenient and efficient since function evaluations in this case are not repeated with increasing the sparse grid level. Therefore, we choose the nodes in a nested manner, that is $\mathcal{X}^i \subseteq \mathcal{X}^{i+1}$, where $\mathcal{X}^i$ is the set of nodes at the index level i. Since nodes are nested, we can also define the difference set as $\mathcal{X}_\Delta^{i+1} = \mathcal{X}^{i+1} \backslash \mathcal{X}^i$, where $\mathcal{X}_\Delta^{i+1}$ is a set that contains only the newly added nodes at level $i+1$. An overview of different possible sparse grid choices can be found in [40]. We generate the nodes in the range [0, 1] which is then scaled according to the physical range of the parameters in $\mathcal{X}^i$. In order to avoid confusion, we reserve the use of the term "node" for the unidimensional point while a multidimensional vector of coordinates formed by nodes along each dimension is given the term "point".

Ultimately, we aim to combine sparse grids with the POD method in order to build a nonintrusive ROM model. This imposes an additional constraint on the selection of the unidimensional nodes. This is because the nodes need to be separated sufficiently in parameter space to produce enriched POD modes covering the complete range of dynamics of the model. Nevertheless, such selection of nodes might not be ideal for interpolation. In many studies, Chebyshev nodes were found to perform better than uniform sampling [40]. However, Chebyshev nodes produce more nodes very close to each other at the boundaries and fewer nodes in the central region, increasing the risk of overlooking some dynamics in the inner region. Therefore, in order to achieve maximum separation of nodes over the entire parameter domain, we choose the equidistant rule to generate the nodes.

A Smolyak interpolant is built for the amplitudes $c(x)$ (from Equation (5), where the index h is dropped for notational convenience) by considering an operator $U^i(c)(x)$ that approximates the function $c(x)$ by an expansion as follows:

$$c(x) \approx U^i(c)(x) = \sum_{x_j^i \in \mathcal{X}^i} c(x_j^i) a_{x_j^i}^i(x), \tag{8}$$

where i is the level index, $\mathcal{X}^i$ is the set of nodes x_j^i at level i, $a_{x_j^i}^i(x)$ are basis functions, and $c(x_j^i)$ is the function value evaluated at the support nodes x_j^i. In principle, different choices for the basis functions are possible. However, due to our selection of equidistant nodes, any choice of (global) polynomial basis function is likely to yield poor approximation because of Runge's phenomenon [23]. Additionally, polynomial functions are not suitable for local adaptive strategies since their support covers the entire domain. Piecewise multilinear functions, on the other hand, are flexible because they have local support and thus can be used to refine specific regions of the domain. These basis functions, which are also called the hat functions because of their shape, satisfy $a_{x_j^i}^i(x) \in C([0,1])$, $a_{x_j^i}^i(x_j^i) = 1$, $a_{x_j^i}^i(y_j^i) = 0 \ \forall \ y_j^i \in \mathcal{X}^i$, $x_j^i \neq y_j^i$. For equidistant type nodes, we can define the basis functions as follows [40]:

$$\begin{aligned} a_{x^1}^1 &= 1 \quad \text{if } i = 1, \\ a_{x_j^i}^i(x) &= \begin{cases} 1 - (m^i - 1) \cdot |x - x_j^i|, & \text{if } |x - x_j^i| < \dfrac{1}{m^i - 1}, \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{9}$$

where m_i and the equidistant nodes x_j^i are defined as follows:

$$m^i = \begin{cases} 1 & \text{if } i = 1, \\ 2^{i-1} + 1 & \text{if } i > 1, \end{cases} \tag{10}$$

$$x_j^i = \begin{cases} 0.5 & \text{for } j = 1 \quad \text{if } m^i = 1, \\ \dfrac{j-1}{m^i - 1} & \text{for } j = 1, 2, \ldots, m^i \quad \text{if } m^i > 1. \end{cases} \tag{11}$$

m^i represents the number of nodes at level i (cardinality of $\mathcal{X}^i$).

Before generalizing to the multivariate case, we first define the difference formula

$$\Delta^i(c)(x) = U^i(c)(x) - U^{i-1}(c)(x), \tag{12}$$

with $U^0 = 0$. As a consequence of selecting nested nodes, the interpolant at level i can always recreate the interpolant at level $i-1$ (i.e., $U^{i-1}(c)(x) = U^i(U^{i-1}(c)(x))$). Therefore, Equation (12) can be rewritten in terms of the basis functions in Equation (8) as,

$$\Delta^i(c)(x) = \sum_{x_j^i \in \mathcal{X}^i} a_{x_j^i}^i(x) c(x_j^i) - \sum_{x_j^i \in \mathcal{X}^i} a_{x_j^i}^i(x) (U^{i-1}(c)(x_j^i)), \tag{13}$$

$$= \sum_{x_j^i \in \mathcal{X}^i} a_{x_j^i}^i(x) (c(x_j^i) - U^{i-1}(c)(x_j^i)). \tag{14}$$

Since the interpolant is completely represented at level $i-1$, $(c(x_j^i) - U^{i-1}(c)(x_j^i)) = 0, \forall x_j^i \in \mathcal{X}^{i-1}$. Thus,

$$\Delta^i(c)(x) = \sum_{x_j^i \in \mathcal{X}_\Delta^i} a_{x_j^i}^i(x) (c(x_j^i) - U^{i-1}(c)(x_j^i)). \tag{15}$$

This means that the interpolant needs to be evaluated only at the newly added nodes at each level increase. Thus, we can redefine x_j^i as the jth element in the set $\mathcal{X}_\Delta^i$. Knowing that the number of newly added nodes (cardinality of $\mathcal{X}_\Delta^i$) is $m_\Delta^i = m^i - m^{i-1}$, the difference formula can be written as,

$$\Delta^i(c)(x) = \sum_{j=1}^{m_\Delta^i} a_{x_j^i}^i(x) (c(x_j^i) - U^{i-1}(c)(x_j^i)). \tag{16}$$

Hence, the sum only runs over the newly added elements that are stored in $\mathcal{X}_\Delta^i$. The contributions of the previous level need not to be considered.

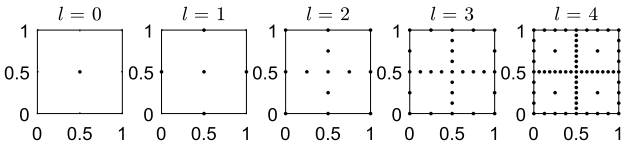


Fig. 1. Sparse grid points for $d = 2$ and different values of l.

Table 1
Number of sparse grid points by level and dimension.

Level (l)	$d=2$	$d=3$	$d=4$	$d=5$	$d=6$
0	1	1	1	1	1
1	5	7	9	11	13
2	13	25	41	61	85
3	29	69	137	241	389
4	65	177	401	801	1457
5	145	441	1105	2433	4865
6	321	1073	2929	6993	15121

The Smolyak algorithm can be applied to combine the unidimensional nodes into sparse grids by satisfying the following condition:

$$d \leq |\mathbf{i}| \leq l + d\,, \tag{17}$$

where d is the number of dimensions, $|\mathbf{i}| = i_1 + i_2 + \ldots + i_d$ with i_n being the index level along dimension n, and l defines the level of the sparse grids. Therefore, the points of the sparse grid at level l are formed by the set

$$B_{l,d} = \bigcup_{d \leq |\mathbf{i}| \leq l+d} (\mathcal{X}^{i_1} \otimes \mathcal{X}^{i_2} \otimes \ldots \otimes \mathcal{X}^{i_d}) \tag{18}$$

Fig. 1 shows the development of the sparse grids from level $l = 0$ to $l = 4$ for a two dimensional space ($d = 2$). Table 1 lists the number of sparse grid points generated per level and dimension.

The unidimensional formulation in Equation (8) can be extended to the multivariate case using the tensor product operation

$$(U^{i_1}(c)(x_1) \otimes \cdots \otimes U^{i_d}(c)(x_d)) = \sum_{j_1=1}^{m^{i_1}} \cdots \sum_{j_d=1}^{m^{i_d}} c(x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d}) \cdot (a^{i_1}_{x^{i_1}_{j_1}}(x_1) \otimes \cdots \otimes a^{i_d}_{x^{i_d}_{j_d}}(x_d)). \tag{19}$$

This shows that building the interpolant needs $\prod_{k=1}^{d} m^{i_k}$ function evaluations, which increases exponentially with the dimension. Smolyak combination technique can be used to reduce the number of function evaluations. The idea is based on the fact that not all points contribute equally to the interpolant, some have a minimal contribution which can be neglected. Therefore, a hierarchically structured algorithm can be built that includes points iteratively until a desired accuracy is reached.

The Smolyak combination technique forms the multivariate interpolant from the univariate difference formula (Equation (16)) as follows:

$$A_{l,d}(c)(\mathbf{x}) = \sum_{|\mathbf{i}| \leq l+d} \Delta^{i_1}(c)(x_1) \otimes \cdots \otimes \Delta^{i_d}(c)(x_d), \tag{20}$$

where l and $|\mathbf{i}|$ are defined as in Equation (17). This Equation can be split into two parts,

$$A_{l,d}(c)(\mathbf{x}) = \underbrace{\sum_{|i| < l+d} (\Delta^{i_1}(c)(x_1) \otimes \cdots \otimes \Delta^{i_d}(c)(x_d))}_{A_{l-1,d}(c)(\mathbf{x})} + \underbrace{\sum_{|i| = l+d} (\Delta^{i_1}(c)(x_1) \otimes \cdots \otimes \Delta^{i_d}(c)(x_d))}_{\Delta A_{l,d}(c)(\mathbf{x})}\,. \tag{21}$$

The first term ($A_{l-1,d}(c)(x)$) represents the interpolant value at the previous interpolation level and the second term ($\Delta A_{l,d}(c)(x)$) is the interpolation contributions from the newly added points. Since the points at each level are defined as a subset of the next level ($\mathcal{X}^i \subseteq \mathcal{X}^{i+1}$), the second term can be written in terms of the basis functions as

$$\Delta A_{l,d}(c)(\mathbf{x}) = \sum_{|\mathbf{i}|=l+d} \sum_{\mathbf{j}} (a^{i_1}_{x^{i_1}_{j_1}}(x_1) \otimes \cdots \otimes a^{i_d}_{x^{i_d}_{j_d}}(x_d)) \cdot \underbrace{(c(x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d}) - A_{l-1,d}(c)(x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d}))}_{w^{\mathbf{i}}_{\mathbf{j}}}, \tag{22}$$

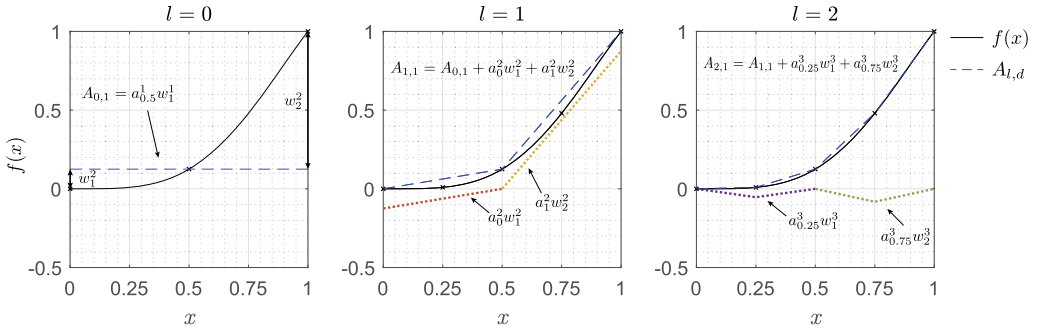


Fig. 2. The progression of the interpolant $A_{l,d}$ for a one dimensional function $f(x) = x^2 \sin(\frac{\pi}{2}x)^2$ from the first level $l = 0$ to level $l = 2$. The weights w^i_j and the basis functions $a^i_{x_j}$ are also shown. The generated nodes are $\mathcal{X}^1_\Delta = \{0.5\}$ at level $l = 0$, $\mathcal{X}^2_\Delta = \{0, 1\}$ at level $l = 1$, and $\mathcal{X}^3_\Delta = \{0.25, 0.75\}$ at level $l = 2$.

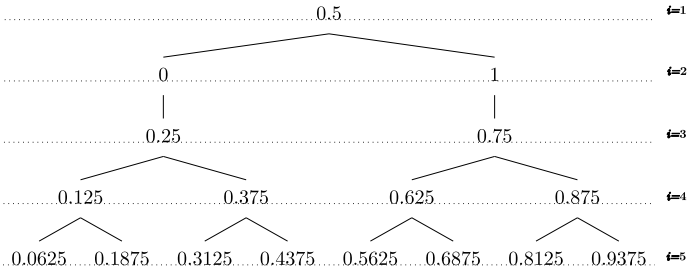


Fig. 3. Tree structure for the nodes in $\mathcal{X}^i_\Delta$ where the depth is assigned the level index i. Nodes are added at each level to half the distances between the nodes in the previous level.

where $\mathbf{j}$ is a multi-index $(j_1, \ldots, j_d)$, $j_k = 1, \ldots, m^{i_k}_\Delta$, $k = 1, \ldots, d$, and $m^{i_k}_\Delta$ is the number of newly added nodes along dimension k. Note that any point $\mathbf{x^i_j} = (x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d})$ at level l is included in the set $B_{l,d}$ (from Equation (18)). The term denoted by $w^\mathbf{i}_\mathbf{j}$ is called *hierarchical surplus* [40] which is the difference between the true function values at the newly added points and the corresponding approximation of the interpolant at the previous level. Therefore, these coefficients are simply a correction of the interpolant at level $l - 1$ to the actual values. Fig. 2 illustrates the progress of the interpolant for a simple one-dimensional function.

3.2. Locally adaptive sparse grids

The local adaptive method can be illustrated by showing the unidimensional nodes $\mathcal{X}^i_\Delta$ in a tree-like structure. Fig. 3 shows such a tree where the depth of the tree has been assigned the level index i. The root of the tree has a single node $\mathcal{X}^1 = \{0.5\}$. It is evident that nodes are added at each level to half the distances between the nodes in the previous levels. Therefore, each node has an ancestry as shown in the tree structure (Fig. 3). Each node has one parent and two children, except at index level $i = 2$ where each node has only one child. This ancestry dependence can be extended to multidimensional points by relating each point to a set of neighbouring points called forward points. Specifically, a forward point to $\mathbf{x}$ is a point on the grid that shares all nodes with $\mathbf{x}$ except along one dimension where the forward point node is a child of the node of $\mathbf{x}$. To that end, we define a forward neighbourhood operator $\Psi(\mathcal{S})$ that operates on a set of points $\mathcal{S} = \{\mathbf{x}_q | q = 1, \ldots, n\}$ and returns all forward points for all points in $\mathcal{S}$ as follows:

$$\Psi(\mathcal{S}) = \{(y_1, \ldots, y_d) | \ \exists i, q : b(y_i) = x_{q,i} \ \wedge \ y_j = x_{q,j} \ \forall j \neq i, \ q \in [1, \ldots, n], \ j, i \in [1, \ldots d]\}, \tag{23}$$

where $b(x)$ is a function that returns the parent of a node x from the tree. We also define a backward point for $\mathbf{x}$ as a point with a parent node along one of the dimensions of $\mathbf{x}$. A backward neighbourhood operator $\Psi^{-1}(\mathcal{S})$ that operates on the set $\mathcal{S}$ and returns the set of all backward points can be defined as

$$\Psi^{-1}(\mathcal{S}) = \{(y_1, \ldots, y_d) | \ \exists i, q : b(x_{q,i}) = y_i \ \wedge \ y_j = x_{q,j} \ \forall j \neq i, \ q \in [1, \ldots, n], \ j, i \in [1, \ldots d]\}. \tag{24}$$

Each point on the grid is surrounded by $2d$ forward points and d backward points. However, because of the exception at tree level 2 where nodes generate only one child, points that contain a node from level 2 have less than $2d$ forward points. Additionally, points that contain the root node 0.5 have less than d backward points because the parent function $b(x)$ returns the root itself for the root node (i.e., $b(0.5) = 0.5$). Note that the forward points are not unique since the same

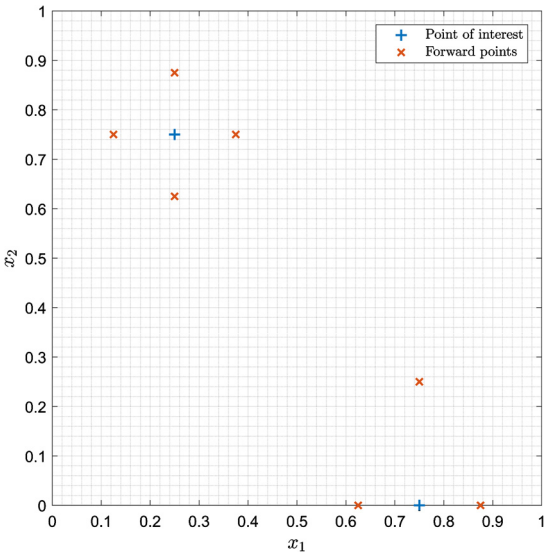


Fig. 4. A 2-dimensional example showing the four forward points of the points $\mathbf{x} = (0.25, 0.75)$ and the three forward points of the point $\mathbf{x} = (0.75, 0)$.

forward point can be generated from different points. Thus, it is important to keep track of the generated forward points to avoid duplication of points. By applying the backward neighbourhood operator successively, we can define the set of ancestor points $\Gamma(\mathcal{S})$ for all points $\mathbf{x}$ in $\mathcal{S}$

$$\Gamma(\mathcal{S}) = \bigcup_{q=1}^{L} \left(\Psi^{-1}\right)^q (\mathcal{S}), \tag{25}$$

where $\left(\Psi^{-1}\right)^L (\mathcal{S}) = (0.5, \ldots, 0.5)$. The set of ancestors for a point $\mathbf{x}$ represents all points with basis functions that contribute to the construction of the interpolant at point $\mathbf{x}$. Fig. 4 shows an example point $\mathbf{x} = (0.25, 0.75)$ with its forward points. The figure also shows a point on the boundary containing a node from level 2 $\mathbf{x} = (0.75, 0)$, which has less than $2d$ forward points.

3.2.1. Locally adaptive algorithm

The basic idea of the locally adaptive algorithm is to set a criterion for selecting important points then refining the grid iteratively by adding only the forward points of the selected important points.

Let $\mathcal{Z}^{k-1}$ be the set of important points, and I^{k-1} be a set of inactive points that were considered unimportant at iteration $k-1$. During the next iteration (k), the algorithm generates a testing set $\mathcal{T}^k$ from the forward points of $\mathcal{Z}^{k-1}$, and I^{k-1} as

$$\mathcal{T}^k = \Psi(\mathcal{Z}^{k-1}) \cup I^{k-1}, \tag{26}$$

and identifies a subset $\mathcal{Z}^k \subseteq \mathcal{T}^k$ that is considered important, which is then added to the grid $\mathcal{X}^k$ (i.e., $\mathcal{X}^k = \mathcal{Z}^k \cup \mathcal{X}^{k-1}$). This process is repeated until some global criterion is met.

In order to identify the important points $\mathcal{Z}^k$, we need to define a local error measure (ϵ_j^k). A point $\mathbf{x}_j$ in the testing set $\mathcal{T}^k$ is considered important and is admitted in $\mathcal{Z}^k$ if it has an error (ϵ_j^k) above a certain threshold (γ_{int})

$$\mathcal{Z}^k = \{\mathbf{x}_j \in \mathcal{T}^k | \epsilon_j^k > \gamma_{\text{int}}\}. \tag{27}$$

Points that do not meet this criterion are stored in the inactive set

$$\mathcal{I}^k = \{\{\mathcal{T}^k \backslash \mathcal{Z}^k\} \cup \mathcal{I}^{k-1}\}. \tag{28}$$

At each iteration, we need to evaluate the interpolant at the testing points $\mathcal{T}^k$ in order to compute ϵ_j^k. However, the interpolant in Equation (21) is written in terms of the (global) level l which is not relevant in the adaptive scheme because points are added based on their location and ancestry. Therefore, we can rewrite the interpolant in Equation (21) and Equation (22) for any point $\mathbf{x} = (x_1, \ldots, x_d)$ in terms of iteration k as

$$A_{k,d}(c)(\mathbf{x}) = A_{k-1,d}(c)(\mathbf{x}) + \Delta A_{k,d}(c)(\mathbf{x}) \,, \tag{29}$$

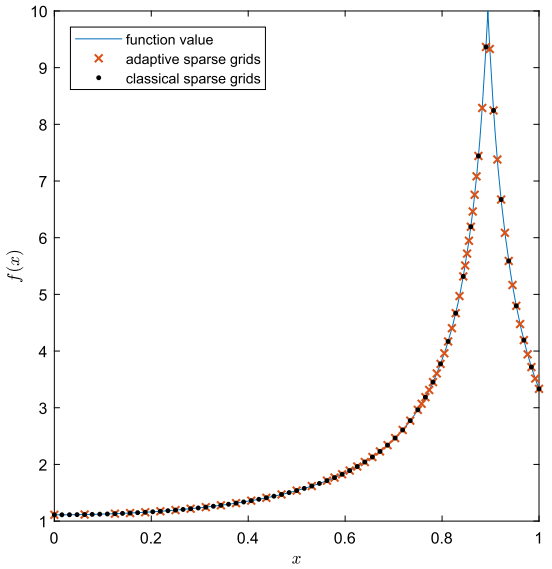


Fig. 5. A comparison showing the difference in sampling the function $f(x) = \frac{1}{|0.8-x^2|+0.1}$ between adaptive and classical sparse grids algorithms. The first 65 sampled points from both algorithms are marked, which shows the adaptive algorithm selecting more points around the peak and fewer at the smooth region.

$$\Delta A_{k,d}(c)(\mathbf{x}) = \sum_{n=1}^{m_k^\Delta} w_n^k \Theta_n(\mathbf{x}), \tag{30}$$

where $m_k^\Delta = card(Z^k)$, and Θ_n is the d-dimensional basis function for the point $\mathbf{x}_n \in \mathcal{Z}_k$,

$$\Theta_n(\mathbf{x}) = \prod_{p=1}^{d} a_{i_p}^{i_p}{}_{x_{n,p}^{i_p}}(x_p), \tag{31}$$

where $\mathbf{x}_n$ has support nodes $(x_{n,1}^{i_1}, \ldots, x_{n,d}^{i_d})$, and i_p is the level (tree depth) index for the support node $x_{n,p}^{i_p}$. The surplus corresponding to $\mathbf{x}_n$ is defined as

$$w_n^k = c(\mathbf{x}_n) - A_{k-1,d}(\mathbf{x}_n). \tag{32}$$

Once the important points are identified, they are added to the set $\mathcal{X}^k$ and their corresponding surpluses are stored in the set $\mathcal{W}^k$. The hierarchical surplus as defined in Equation (32) is a natural candidate for the local error measure ϵ_j^k. These surpluses are defined locally (for each point) and represent the deviation from the true value. This criterion was applied in [33,35]. However, Griebel (1998) [31] showed that taking the absolute value of the surpluses is too sharp of an indicator and can lead to a non-terminating algorithm in some cases. The author suggested weighing the surpluses with the integral of the corresponding basis functions in order to give more importance to points with basis functions that have wider support. This criterion was used in [36,34,32]. In our implementation, we combine the adaptivity with the POD to model a physical field. Therefore, we choose a local error measure based on physical space rather than the surpluses which are defined in parameter space. The local error measure in our approach is presented in Section 4 after presenting the method of integrating the POD with the adaptive sparse grids.

To highlight the difference between the classical and the locally adaptive sparse grids, Fig. 5 shows an analytical function that was sampled using both approaches. The first 65 sampled points from both algorithms are marked on the figure. The classical sparse grids algorithm resulted in a uniform sampling regardless of the function's behaviour. The adaptive algorithm, on the other hand, was more efficient by spending more points around the steep gradient and fewer points in the smooth region.

3.2.2. Including ancestor points in the adaptivity

A refinement criterion based entirely on the local error measure can lead to premature termination of the algorithm. This is observed when the true function value intersects (or closely intersects) the interpolant at the forward points. For example, the adaptive algorithm based only on a local error measure never converged when tested on the Rosenbrock function in 2D, defined as $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$ and $x, y \in (0, 1)$. The reason is that the value of this function at the root point $(0.5, 0.5)$ and at one of its forward points $(0.5, 0)$ are equal. Therefore, the surplus computed at the point $(0.5, 0)$

is zero which falsely implies that the interpolant is accurate around this point. As a consequence, the algorithm stops refining around the point $(0.5, 0)$ and assumes a constant interpolation between $(0.5, 0.5)$ and $(0.5, 0)$, which is incorrect. The missing behaviour around $(0.5, 0)$ cannot be recovered at subsequent iterations. This is significant because the basis function at $(0.5, 0)$ has support that spans half the domain.

This issue can be mitigated by redefining the selection of the important points to include the ancestors. Including missing ancestors has been discussed as part of building an accurate hierarchical interpolant (see [32]). Bungartz et al. (2003) [36] used the missing ancestors as support points for the d-polynomial basis functions where they contribute to the calculations of the surpluses of their descendants. In our approach, the goal is to use the adaptivity to explore the parameter space. Therefore, the ancestors are not only included for building the interpolant but also to search in the forward points of these ancestors for any possible missing behaviour. In this case, the local error indicator ϵ_j^k is still a measure for the importance of the region around the point but the algorithm prioritizes the search in the vicinity of ancestors before moving to the forward points. This is important because the basis functions for ancestors have wider support compared to the descendants. However, in order to keep the number of evaluations reduced, not all ancestors are included in the important set.

The important set is redefined to take into account the ancestors as follows:

$$\begin{aligned}
&\mathcal{C}^k = \{\mathbf{x}_j \in \mathcal{T}^k \mid \epsilon_j^k > \gamma_{\text{int}}\},\\
&\mathcal{Z}_a^k = \{\mathbf{x}_j \in \mathcal{C}^k \mid \ \Gamma(\mathbf{x}_j) \subseteq \mathcal{X}^{k-1}\},\\
&\mathcal{Z}_b^k = \{\mathbf{y}_i \in \Gamma(\mathbf{x}_j) \mid \mathbf{x}_j \in \mathcal{C}^k,\ \Gamma(\mathbf{x}_j) \cap \mathcal{C}^k = \emptyset \ \wedge\ \mathbf{y}_i \notin \mathcal{X}^{k-1} \wedge \Gamma(\mathbf{y}_i) \subseteq \mathcal{X}^{k-1}\},\\
&\mathcal{Z}^k = \mathcal{Z}_a^k \cup \mathcal{Z}_b^k.
\end{aligned} \tag{33}$$

In this definition, we first identify a set of candidate points $\mathcal{C}^k$ containing all points with an error above the defined threshold γ_{int}. Then, the important points set $\mathcal{Z}^k$ is formed by two parts. The first $\mathcal{Z}_a^k$ are points within the candidate points $\mathcal{C}^k$ that have all their ancestors already included in the sparse grid $\mathcal{X}^{k-1}$ from previous iterations. The second $\mathcal{Z}_b^k$ are the missing ancestors of candidate points with partial ancestry in $\mathcal{X}^{k-1}$. However, candidate points that have any ancestor point also as a candidate will not be considered because the error at these points is likely to be high due to the error at that ancestor. Such points will be added to the inactive set to be tested again in the next iteration after including the ancestor point first. Applying this strategy to the aforementioned Rosenbrock function resulted in the algorithm converging with 967 points to a relative error of 1%.

Such definition for the set of important points (Equation (33)) enhances the quality of exploring the parameter space because for every important point identified in iteration k, all $2d$ forward points will be tested in the next iteration (Equation (26)). However, for high dimensional problems where the model is linear (or almost linear) along certain dimensions, refining the grid in all dimensions unnecessarily increases the number of model evaluations. For such cases, we can control the number of model evaluations by introducing a parameter μ that tunes the greediness of the sampling scheme. This parameter reduces the number of points in the testing set $\mathcal{T}^{k+1}$ as follows:

$$\mathcal{T}^{k+1} = \left\{ \mathbf{x}_j \in \Psi(\mathcal{Z}^k) \;\middle|\; \frac{card(\Psi^{-1}(\mathbf{x}_j) \cap \mathcal{X}^k)}{card(\Psi^{-1}(\mathbf{x}_j))} \geq 1 - \mu \right\} \cup \mathcal{I}^k, \tag{34}$$

where the operator $card(\mathcal{Y})$ returns the cardinality of a set $\mathcal{Y}$, and μ is the greediness parameter that has a value $\in [0, 1]$. For every forward point in $\Psi(\mathcal{Z}^k)$, the fraction of its backward points that are included in the important set $\mathcal{X}^k$ is required to be greater than or equal to $1 - \mu$ in order for this point to be tested. Note that each point in a d-dimensional grid has up to d backward points. The algorithm is greedy for $\mu = 1$ because all forward points will be admitted and tested. For $\mu = 0$, a point will only be tested if all of its backward points were important, which directs the algorithm to avoid searching regions with no important points. The concept of only considering points whose backward points are important is inspired by the (anisotropic) dimension adaptive sparse grids, where indices of important grids are identified based on the importance of the indices of its backward neighbours (from previous iterations) [27]. By extending this concept to the local adaptivity, we have better control of the number of model evaluations, at the expense of exploring the parameter space less thoroughly.

3.2.3. Validation

Including the ancestors can only mitigate the premature termination issue but not resolve it completely. This can be observed, for example, in the one-dimensional sine function $\sin(2\pi x)$, which has the same value at the root point $\{0.5\}$ and at its forward points $\{0,1\}$. Both forward points, in this case, will have zero surpluses ($w_1^2 = w_2^2 = 0$). Therefore, the algorithm will still terminate without any further refinement even with including the ancestors rule. This issue can also arise in multidimensional functions. We can try to define a different error criterion to circumvent this case. However, as stated by Griebel [31], for any given error criteria, we can always find a function that will cause the algorithm to terminate prematurely. Therefore, we propose to include a validation step after convergence that will test the model at randomly generated points. If any of the random points results in an error greater than the tolerance, the algorithm enriches the model with more points around that point. This is achieved by considering all points from the inactive set with basis functions contributing to the interpolant at the random validation points

$$\mathcal{Q} = \{\mathbf{x}_n \in \mathcal{I}^k \mid |x_{r,p} - x_{n,p}^{i_p}| < \frac{1}{m_n^{i_p} - 1}, \forall i_p \neq 1, p = 1, \ldots, d\}, \tag{35}$$

where $\mathbf{x}_r = (x_{r,1}, \ldots, x_{r,d})$ is the tested random point, $x_{n,p}^{i_p}$ is the support node of the point $\mathbf{x}_n$ along dimension p with tree depth i_p, and $m_n^{i_p}$ is defined as in Equation (10). The points in $\mathcal{Q}$ are then considered candidate points (i.e., $\mathcal{C}^k = \mathcal{Q}$) and the adaptive algorithm is resumed. The basic principle here is that if the random point has an error above the tolerance, it implies that the algorithm incorrectly categorized one of the contributing basis functions as not important (added to the inactive set) and failed to build an accurate interpolant in that region. Therefore, reconsidering these points as candidate points will enrich the model around this region.

4. POD-Adaptive algorithm

The adaptive algorithm is used to guide the sampling scheme for the POD method. At each iteration, the high fidelity model $\mathbf{y}(\mathbf{x})$ is sampled at new grid points $\mathcal{T}^k$. Then based on the predefined error measure, an important subset $\mathcal{Z}^k$ is identified and added to $\mathcal{X}^k$. The snapshots corresponding to the points in $\mathcal{Z}^k$ are then added to the snapshots set $\mathcal{F}_k = \{\mathbf{y}(\mathbf{x}_j) | \mathbf{x}_j \in \mathcal{X}^k, j = 1, \ldots m_k\}$. We then perform a SVD on the snapshots and obtain new POD modes $\{\mathbf{u}_h \, | h = 1, \ldots r_k\}$. Each iteration will result in a new set of POD modes $\mathbf{u}_h$ and, consequently, a new set of amplitudes $c_h(\mathbf{x})$. Moreover, the number of POD modes might increase from iteration to the next because of the addition of new snapshots. As a consequence, the number of functions to be interpolated (amplitudes of the POD modes) also increases. In this section, we propose a scheme that is able to keep track of the changes in the amplitudes with minimized computational cost.

At iteration k, the high fidelity model is approximated as

$$\mathbf{y}(\mathbf{x}) \approx \sum_{h=1}^{r_k} c_h(\mathbf{x})\mathbf{u}_h. \tag{36}$$

Using the orthogonality of the POD modes, we can define the amplitudes at any point $\mathbf{x}_i$ as

$$c_h(\mathbf{x}_i) = < \mathbf{y}(\mathbf{x}_i), \mathbf{u}_h > . \tag{37}$$

We aim to approximate $c_h(\mathbf{x})$ with the interpolant $A_{k,d}(c_h)(\mathbf{x})$ (Equation (30)) and eventually build a ROM $\mathbf{y}_a(\mathbf{x})$ approximating $\mathbf{y}(\mathbf{x})$ such that

$$\mathbf{y}_a(\mathbf{x}) = \sum_{h=1}^{r_k} A_{k,d}(c_h)(\mathbf{x})\mathbf{u}_h. \tag{38}$$

The interpolant $A_{k,d}(c_h)(\mathbf{x})$ depends on the grid points $\mathcal{X}^k$ and the surpluses $\mathcal{W}_h^k$. For every amplitude (c_h), a specific interpolant is built with a corresponding set of surpluses $\mathcal{W}_h^k$.

At iteration $k+1$, the adaptive algorithm selects a new set of grid points $\mathcal{Z}^{k+1}$. Then, the set of snapshots are updated, $\mathcal{F}^{k+1} = \{\mathbf{y}(\mathbf{x}_j) | \mathbf{x}_j \in \mathcal{X}^{k+1}, j = 1, \ldots m_{k+1}\}$, where $\mathcal{X}^k \subset \mathcal{X}^{k+1}$ and $m_{k+1} > m_k$. As a consequence, we obtain a new set of POD modes $\{\hat{\mathbf{u}}_h \, | h = 1, \ldots r_{k+1}\}$ and corresponding amplitudes $\{\hat{c}_h(\mathbf{x}) \, | h = 1, \ldots r_{k+1}\}$. The POD model at iteration $k+1$ can be written as

$$\mathbf{y}_a(\mathbf{x}) = \sum_{g=1}^{r_{k+1}} \hat{c}_h(\mathbf{x})\hat{\mathbf{u}}_h. \tag{39}$$

In principle, $\hat{c}_h(\mathbf{x})$ is a new function of $\mathbf{x}$ and is not related to $c_h(\mathbf{x})$ because the POD modes are different (i.e., $\mathbf{u}_h \neq \hat{\mathbf{u}}_h$). Therefore, in order to construct the ROM as in Equation (38), a new interpolant for $\hat{c}_h$ $(A_{k+1,d}(\hat{c}_h)(\mathbf{x}))$ needs to be rebuilt hierarchically starting from the first iteration. From Equation (29), $A_{k+1,d}(\hat{c}_h)(\mathbf{x})$ is formed by two parts

$$A_{k+1,d}(\hat{c}_h)(\mathbf{x}) = A_{k,d}(\hat{c}_h)(\mathbf{x}) + \Delta A_{k+1,d}(\hat{c}_h)(\mathbf{x}). \tag{40}$$

The first term $A_{k,d}(\hat{c}_h)(\mathbf{x})$ is the interpolant from the previous iterations, which needs to be computed first for all $\mathbf{x}_j \in \mathcal{X}^k$ in order to obtain the unknown surpluses $\hat{\mathcal{W}}_h^k$. Recomputing the entire interpolant for all previous points at each iteration is inefficient and counter-productive to the hierarchical structure of the Smolyak algorithm. However, since the grid points are selected in a nested manner, we can find an efficient way to update the surpluses from the previous iterations without having to recompute the interpolant hierarchically. Once these surpluses are updated, the surpluses for the new points in $\mathcal{Z}^{k+1}$ can be computed for the second term $\Delta A_{k+1,d}(\hat{c}_h)(\mathbf{x})$ as in Equation (32).

To update the surpluses $\hat{\mathcal{W}}_h^k$, we first notice that the amplitudes from the two consecutive iterations k and $k+1$ are not equal, that is

$$\hat{c}_h(\mathbf{x}) \neq c_h(\mathbf{x}). \tag{41}$$

In physical space, however, assuming negligible SVD truncation error, both POD models from Equation (36) and Equation (39) are defined to reproduce the exact snapshots at the points in $\mathcal{X}^k$ because the points are nested ($\mathcal{X}^k \subset \mathcal{X}^{k+1}$). Therefore,

$$\sum_{g=1}^{r_{k+1}} \hat{c}_g(\mathbf{x}_j)\hat{\mathbf{u}}_g = \sum_{h=1}^{r_k} c_h(\mathbf{x}_j)\mathbf{u}_h \qquad \forall \mathbf{x}_j \in \mathcal{X}^k. \tag{42}$$

We can use the orthogonality property and project the equation onto $\hat{\mathbf{u}}_g$ to obtain the amplitudes

$$\hat{c}_g(\mathbf{x}_j) = \sum_{h=1}^{r_k} c_h(\mathbf{x}_j) < \mathbf{u}_h, \hat{\mathbf{u}}_g > \qquad \forall \mathbf{x}_j \in \mathcal{X}^k,\ g = 1, \ldots, r_{k+1}. \tag{43}$$

Since the interpolant $A_{k,d}(\hat{c}_g)(\mathbf{x})$ is a function of the surpluses rather than the function $\hat{c}_g$, it is more convenient to find a relation between $\hat{w}_j^k$ and w_j^k. We first note that the set $\mathcal{X}^k$ is formed by the union of the important points from all previous iterations, that is

$$\mathcal{X}^k = \bigcup_{l=1}^{k} \mathcal{Z}^l. \tag{44}$$

The definition of the surpluses in Equation (32) can be written for amplitude $c_h(\mathbf{x}_j)$ at any iteration l as

$$w_{j,h}^l = c_h(\mathbf{x}_j) - A_{l-1,d}(c_h)(\mathbf{x}_j), \qquad \forall \mathbf{x}_j \in \mathcal{Z}^l. \tag{45}$$

Substituting Equation (45) in Equation (43)

$$\hat{w}_{j,g}^l + A_{l-1,d}(\hat{c}_g)(\mathbf{x}_j) = \sum_{h=1}^{r_k} \left[w_{j,h}^l + A_{l-1,d}(c_h)(\mathbf{x}_j) \right] < \mathbf{u}_h, \hat{\mathbf{u}}_g > \quad \forall \mathbf{x}_j \in \mathcal{Z}^l, l = 1, \ldots, k,\ g = 1, \ldots, r_{k+1}. \tag{46}$$

We can further reduce Equation (46) by noticing that $A_{0,d}(\hat{c}_g)(\mathbf{x}) = A_{0,d}(c_h)(\mathbf{x}) = 0$. Therefore, for $l = 1$,

$$\hat{w}_{j,g}^1 = \sum_{h=1}^{r_k} \left[w_{j,h}^1 \right] < \mathbf{u}_h, \hat{\mathbf{u}}_g > \quad \forall \mathbf{x}_j \in \mathcal{Z}^1,\ g = 1, \ldots, r_{k+1}. \tag{47}$$

For $l = 2$,

$$\hat{w}_{j,g}^2 + A_{1,d}(\hat{c}_g)(\mathbf{x}_j) = \sum_{h=1}^{r_k} \left[w_{j,h}^2 + A_{1,d}(c_h)(\mathbf{x}_j) \right] < \mathbf{u}_h, \hat{\mathbf{u}}_g > \quad \forall \mathbf{x}_j \in \mathcal{Z}^2,\ g = 1, \ldots, r_{k+1}. \tag{48}$$

Note that both interpolants $A_{k,d}(\hat{c}_g)(\mathbf{x})$ and $A_{k,d}(c_h)(\mathbf{x})$ share the same support nodes.

From the definition of the interpolant in Equation (29) and Equation (30), it follows that

$$A_{1,d}(c)(\mathbf{x}_j) = \Delta A_{1,d} = \sum_{n=1}^{m_1^\Delta} w_n^1 \Theta_n(\mathbf{x}_j). \tag{49}$$

Thus, Equation (48) becomes

$$\hat{w}_{j,g}^2 + \sum_{n=1}^{m_1^\Delta} \hat{w}_{n,g}^1 \Theta_n(\mathbf{x}_j) = \sum_{h=1}^{r_k} \left[w_{j,h}^2 + \sum_{n=1}^{m_1^\Delta} w_{n,h}^1 \Theta_n(\mathbf{x}_j) \right] < \mathbf{u}_h, \hat{\mathbf{u}}_g > \quad \forall \mathbf{x}_j \in \mathcal{Z}^2,\ g = 1, \ldots, r_{k+1}. \tag{50}$$

Using Equation (47) to replace $\hat{w}_{n,g}^1$ in Equation (50), we get

$$\hat{w}_{j,g}^2 + \sum_{n=1}^{m_1^\Delta} \sum_{h=1}^{r_k} w_{n,h}^1 < \mathbf{u}_h, \hat{\mathbf{u}}_g > \Theta_n(\mathbf{x}_j) = \sum_{h=1}^{r_k} \left[w_{j,h}^2 + \sum_{n=1}^{m_1^\Delta} w_{n,h}^1 \Theta_n(\mathbf{x}_j) \right] < \mathbf{u}_h, \hat{\mathbf{u}}_g > \quad \forall \mathbf{x}_j \in \mathcal{Z}^2,\ g = 1, \ldots, r_{k+1}, \tag{51}$$

which simplifies to

$$\hat{w}_{j,g}^2 = \sum_{h=1}^{r_2} w_{j,h}^2 < \mathbf{u}_h, \hat{\mathbf{u}}_g > \qquad \forall \mathbf{x}_j \in \mathcal{Z}^2,\ g = 1, \ldots, r_{k+1}. \tag{52}$$

Recursively, we find a general expression for $\hat{w}^k_{j,g}$ as

$$\hat{w}^k_{j,g} = \sum_{h=1}^{r_k} w^k_{j,h} \; < \mathbf{u}_h, \hat{\mathbf{u}}_g > \qquad \forall \mathbf{x}_j \in \mathcal{X}^k, \; g = 1, \ldots, r_{k+1}. \tag{53}$$

Since w^k_j is simply a measure of the deviation of the interpolant from the true value, this result means that we can simply obtain the surpluses at any iteration $k+1$ by taking this difference from the previous iterations to the physical space first, then projecting onto the new space formed by $\hat{\mathbf{u}}_g$. Equation (53) can be used first to update the surpluses for all previous points $\mathcal{X}^k$ before computing the surpluses at the new points $\mathcal{Z}^{k+1}$ (Equation (32)).

To define the local error measure ϵ^k_j, we choose the L_2-norm error in physical space as follows:

$$\epsilon^k_j = \left\| \mathbf{y}(\mathbf{x}_j) - \sum_{h=1}^{r_k} A_{k,d}(c_h)(\mathbf{x}_j)\mathbf{u}_h \right\|_{L_2} \qquad \forall \mathbf{x}_j \in \mathcal{X}^{\Delta}_k, \tag{54}$$

and the candidate points $\mathcal{C}^k$ are selected as

$$\mathcal{C}^k = \{\mathbf{x}_j \in \mathcal{T}^k | \epsilon^k_j > (\gamma_{\text{int}} \left\| \mathbf{y}(\mathbf{x}_j) \right\|_{L_2} + \zeta_{\text{abs}})\}, \tag{55}$$

where γ_{int} is the interpolation threshold and ζ_{abs} is the absolute tolerance, which is introduced to deal with functions of small magnitudes. The important points $\mathcal{Z}^k$ are then selected by considering the ancestors as in Equation (33).

Additionally, we can define a global error estimate ($\epsilon^k_{\max}$) in the L_∞ norm

$$\epsilon^k_{\max} = \max_j \epsilon^k_j. \tag{56}$$

The algorithm is terminated once the global error is below a given relative tolerance ζ_{rel}. A trivial choice for the tolerance is to be equal to the selected threshold (i.e., $\zeta_{\text{rel}} = \gamma_{\text{int}}$). However, setting a different global tolerance is useful to avoid non-terminating algorithm or oversampling the high fidelity model. The algorithm terminating criterion that takes into account both the relative and absolute error is introduced as

$$\epsilon^k_{\max} < (\zeta_{\text{rel}} \left\| \mathbf{y}(\mathbf{x}_j) \right\| + \zeta_{\text{abs}}). \tag{57}$$

The algorithm is summarized in Algorithm 1 and the validation algorithm is summarized in Algorithm 2. Note that not all sampled points are included in the snapshots. The POD modes are formed only by the snapshots corresponding to the points in $\mathcal{X}^k$. Points that are not considered important (not included in $\mathcal{Z}^k$) are not included in the snapshots. This strategy reduces the computational cost of performing the SVD.

5. Applications

In this section, we present three different numerical tests. The first is a reactor physics problem with 5 dimensions. This problem is presented in two cases in order to test the POD-Adaptive algorithm in two different settings, the first is a strongly nonlinear setting and the second is weakly nonlinear. The second numerical test is a general diffusion problem with 18 dimensions, which demonstrates the performance of the algorithm in higher dimensionality. The validation algorithm is also tested in this problem. The final test case is an oscillatory analytical function in 20 dimensions, which tests the algorithm in identifying regions of strong nonlinearity.

5.1. Test Case 1: point kinetics

Models of nuclear reactors are complex because they aim to capture the dynamics of multi-physics phenomena occurring at various scales. The Point Kinetics model is a simple approximation that models the temporal evolution of the reactor power as a function of perturbations in the reactor. In this model, the spatial behaviour is not considered, and the reactor is condensed into a single point, hence the name. The model is described by a set of ordinary differential equations as follows [41]:

$$\frac{\mathrm{d}P(t)}{\mathrm{d}t} = \frac{\rho(t) - \beta}{\Lambda} P(t) + \lambda C(t), \tag{58}$$

$$\frac{\mathrm{d}C(t)}{\mathrm{d}t} = \frac{\beta}{\Lambda} P(t) - \lambda C(t), \tag{59}$$

where $P(t)$ is the reactor power at time t, ρ is the reactivity of the reactor, β is the effective fraction of delayed neutrons, Λ is the neutron generation time, λ is the one-group decay constant, and $C(t)$ is the one-group precursors concentration.

Algorithm 1 POD-Adaptive.

Require: an interpolation threshold γ_{int}, a relative tolerance ζ_{rel}, an absolute tolerance ζ_{abs}, a POD truncation threshold γ_{tr}, a greediness parameter μ, and a target model $y(\mathbf{x})$.
Ensure: grid points $\mathcal{X}^k$ and their surpluses $\mathcal{W}_h^k$, inactive set $\mathcal{I}^k$, and POD modes $\mathbf{u}_h$.
1: **Initialization**:
- set $k=1$, $\mathcal{X}^1=\mathcal{Z}^1=\{(0.5,\ldots,0.5)\}$, $\mathcal{I}^1=\{\emptyset\}$, $\epsilon_1^1=\inf$
- evaluate the model at $\mathbf{y}(0.5,\ldots,0.5)$ and add the resulting snapshot to $\mathcal{F}$
- perform SVD on the snapshots in $\mathcal{F}$ to obtain $\mathbf{u}_1$
- compute the amplitude $c(0.5,\ldots,0.5)=<\mathbf{y}(0.5,\ldots,0.5),\mathbf{u}_1>$
- set $\mathcal{W}_1^1=\{c_1(0.5,\ldots,0.5)\}$
2: **while** any $\epsilon_j^k > (\zeta_{\text{rel}} \left\|\mathbf{y}(\mathbf{x}_j)\right\| + \zeta_{\text{abs}})$ **do**
3: $k=k+1$
4: compute $\mathcal{T}^k=\left\{\mathbf{x}_j\in\Psi(\mathcal{Z}^{k-1}) \;\middle|\; \frac{card(\Psi^{-1}(\mathbf{x}_j)\cap\mathcal{X}^{k-1})}{card(\Psi^{-1}(\mathbf{x}_j))}\geq 1-\mu\right\}\cup\mathcal{I}^{k-1}$
5: **for all** $\mathbf{x}_j\in\mathcal{T}^K$ **do**
6: evaluate $\mathbf{y}(\mathbf{x}_j)$
7: compute $A_{k-1,d}(c)(\mathbf{x}_j)$ as in Equation (29)
8: compute $\epsilon_j^k=\left\|\mathbf{y}(\mathbf{x}_j)-\sum_{h=1}^{r_k}A_{k-1,d}^h(c_h)(\mathbf{x}_j)\mathbf{u}_h\right\|_{L_2}$
9: **end for**
10: find the candidate points $\mathcal{C}^k=\{\mathbf{x}_j\in\mathcal{T}^k|\epsilon_j^k>(\gamma_{\text{int}}\left\|\mathbf{y}(\mathbf{x}_j)\right\|_{L_2}+\zeta_{\text{abs}})\}$.
11: find $\mathcal{Z}_a^k$, $\mathcal{Z}_b^k$ and $\mathcal{Z}^k$ as in Equation (33)
12: compute $\mathcal{X}^k=\mathcal{Z}^k\cup\mathcal{X}^{k-1}$
13: update the inactive set $\mathcal{I}^k=\{\mathcal{T}^k\backslash\mathcal{Z}^k\}\cup\mathcal{I}^{k-1}$
14: add the snapshots corresponding to the points in $\mathcal{Z}^k$ to $\mathcal{F}$
15: perform SVD on the snapshots in $\mathcal{F}$ to obtain new POD modes $\hat{u}_g$
16: truncate the POD modes such that $e_r<\gamma_{\text{tr}}$
17: use Equation (53) to update the surpluses for points in $\mathcal{X}^{k-1}$
18: compute the surpluses at the selected points in $\mathcal{Z}^k$ as in Equation (32)
19: **end while**

Algorithm 2 POD-Adaptive Validation.

Require: an interpolation threshold γ_{int}, a relative tolerance ζ_{rel}, an absolute tolerance ζ_{abs}, a POD truncation threshold γ_{tr}, a greediness parameter μ, a target model $y(\mathbf{x})$, grid points $\mathcal{X}^k$ and their surpluses $\mathcal{W}_h^k$, inactive set $\mathcal{I}^k$, POD modes u_h, and a number of random points to be tested v.
Ensure: grid points $\mathcal{X}^k$ and their surpluses $\mathcal{W}_h^k$, inactive set $\mathcal{I}^k$, and POD modes u_h.
1: generate a set of v random points $\mathcal{V}$
2: **for all** $\mathbf{x}_r\in\mathcal{V}$ **do**
3: evaluate $\mathbf{y}(\mathbf{x}_r)$
4: compute $A_{k,d}(c)(\mathbf{x}_r)$ as in Equation (29)
5: compute $\epsilon_r=\left\|\mathbf{y}(\mathbf{x}_r)-\sum_{h=1}^{r_k}A_{k,d}^h(c_h)(\mathbf{x}_r)\mathbf{u}_h\right\|_{L_2}$
6: **end for**
7: **while** any $\epsilon_r>(\zeta_{\text{rel}}\left\|\mathbf{y}(\mathbf{x}_r)\right\|+\zeta_{\text{abs}})$ **do**
8: find $\mathcal{Q}$ as in Equation (35)
9: $k=k+1$
10: set $\mathcal{C}^k=\mathcal{Q}$
11: perform *Algorithm 1* starting from step 11.
12: recompute ϵ_r for all $x_r\in\mathcal{V}$
13: **end while**

A reactor is usually controlled by control rods which mostly influence the reactivity of the reactor. A positive reactivity $\rho>\beta$ causes the reactor power to be unstable in a very short period of time, a state called supercritical. On the other hand, $0<\rho<\beta$ also causes a surge of power but at a much-reduced rate which allows for enough time to compensate and control the reactor. Therefore, the time evolution of the power is strongly nonlinear when the reactivity insertion is close to β. This problem is similar to the one presented by Perkó et al. (2014) [42], where the uncertainty in the maximum power was studied using adaptive polynomial chaos.

In this example, we consider a transient problem by assuming an insertion of a large positive reactivity that triggers the emergency safety system of the reactor. Thus, starting from an initial stable reactor at $t<0$ ($\rho=0$), the reactor is perturbed with a step reactivity insertion $\rho(t)=\rho_1>0$ at $t=0$, causing the power to increase exponentially. At $t=t_s$, a strong negative step reactivity is inserted $\rho(t)=\rho_2<0$, which simulates the insertion of shutdown rods to stop the reactor. Under these assumptions, we can solve Equation (58) and Equation (59). For $0\leq t\leq t_s$, we get

$$P(t)=P_0\left[\frac{w^- - \frac{\rho_1}{\Lambda}}{w^- - w^+}e^{w^+t}-\frac{w^+ - \frac{\rho_1}{\Lambda}}{w^- - w^+}e^{w^-t}\right], \tag{60}$$

$$C(t)=\frac{\beta P_0}{\Lambda\lambda}\left[\frac{w^-}{w^- - w^+}e^{w^+t}-\frac{w^+}{w^- - w^+}e^{w^-t}\right], \tag{61}$$

Table 2
Nominal values and the studied range of variations for Setting 1 and Setting 2 of the Point Kinetics model.

Parameter	Setting 1	Setting 2	Change
ρ_1	0.9β	0.7β	±5%
λ	0.09441 s^{-1}	0.09441 s^{-1}	±5%
Λ	0.478×10^{-6} s	0.478×10^{-6} s	±5%
P_0	1 W	1 W	±5%
β	0.00403	0.00403	±5%
ρ_2	-5β	-5β	0
t_s	10^{-2} s	10^{-2} s	0

where

$$w^{\pm} = \frac{\rho_1 - \beta - \lambda\Lambda \pm \sqrt{(\beta - \rho_1 + \lambda\Lambda)^2 + 4\lambda\Lambda\rho_1}}{2\Lambda}, \tag{62}$$

and P_0 is the initial power level.

The solution of the power for $t > t_s$ is

$$P(t) = \left[\frac{P_s w_s^- - P_s \frac{\rho_2 - \beta}{\Lambda} - \lambda C_s}{w_s^- - w_s^+}\right] e^{w_s^+ (t - t_s)} - \left[\frac{P_s w_s^+ - P_s \frac{\rho_2 - \beta}{\Lambda} - \lambda C_s}{w_s^- - w_s^+}\right] e^{w_s^- (t - t_s)}, \tag{63}$$

where P_s and C_s are respectively the power and precursors concentration at $t = t_s$, and

$$w_s^{\pm} = \frac{\rho_2 - \beta - \lambda\Lambda \pm \sqrt{(\beta - \rho_2 + \lambda\Lambda)^2 + 4\lambda\Lambda\rho_2}}{2\Lambda}. \tag{64}$$

We study this model under variations of 5 parameters P_0, λ, Λ, ρ_1, and β. We assume a uniform distribution with ±5% change from the nominal values. We consider two settings. The first is a case where the reactivity insertion is close to the supercritical state ($\rho_1 = 0.9\beta$), which describes strong nonlinear behaviour. The second setting is selected such that the reactor is slightly further from the strong nonlinear behaviour, $\rho_1 = 0.7\beta$. Nominal values of both settings are shown in Table 2.

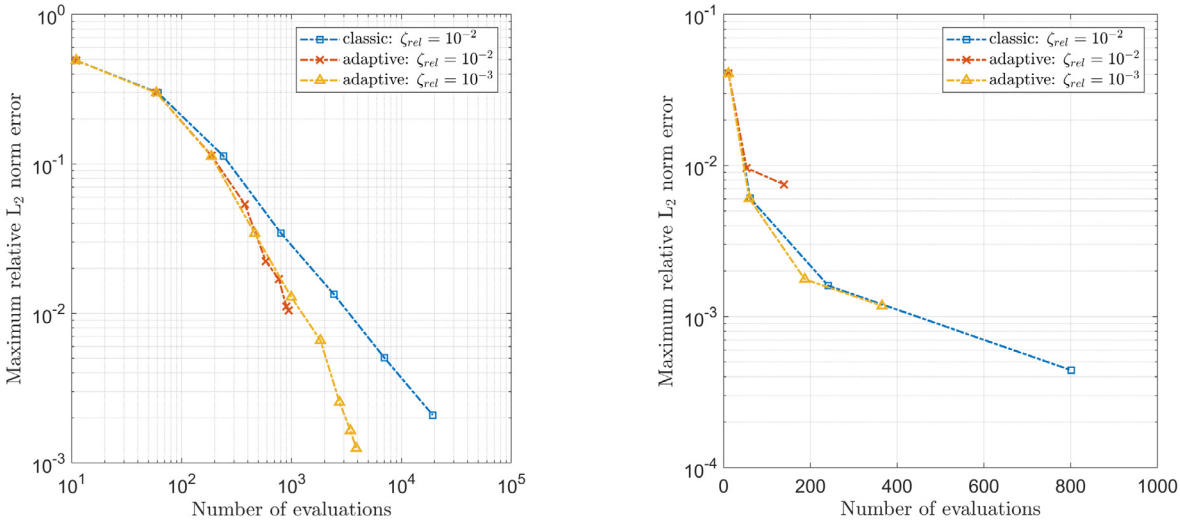
For both settings the relative tolerance (ζ_{rel}) was set to 10^{-2}, absolute tolerance (ζ_{abs}) was 10^{-4}, interpolation threshold (γ_{int}) was 5×10^{-3}, truncation threshold (γ_{tr}) was 10^{-12}, and the greediness parameter μ was 1. A ROM model was built for each setting separately. In Setting 1, the algorithm stopped after sampling 944 points while for the second setting the algorithm required only 139 points, which is expected since Setting 2 is further from the unstable region. However, in both settings, only about 16% of the sampled points were included in the ROM. We then tested the models on 10000 randomly generated points that were not part of the training set. In order to show the progress of the ROM per iteration, Fig. 6 shows the maximum relative L_2 error as a function of the number of model evaluations by testing the model at the random points after each iteration. On the same figure, we also compare the error with a different ROM model built using the classical sparse grid method (no adaptivity). The classical model is equivalent to setting the interpolation threshold γ_{int} and the absolute tolerance ζ_{abs} to be zero.

The advantage of the adaptive strategy compared to the classical is clear for Setting 1 (Fig. 6a) as the tolerance was achieved with a much-reduced number of evaluations. The reason is that the classical algorithm adds points along all dimensions equally in every iteration whereas the adaptive algorithm only adds points in regions of higher error. As a consequence, the adaptive algorithm samples the most sensitive dimensions more densely. This is evident from the number of unique nodes in every dimension selected by the algorithm, which is shown in Table 3. For the initial power level, the algorithm sampled 5 nodes only, which are the first five nodes: the (centre) root node 0.5, the children of the root node (the boundary nodes) 0 and 1, and their children 0.25 and 0.75. This means that the algorithm built a linear interpolant with the nodes 0.5, 0, and 1, then when it tested the interpolant at the nodes 0.25, and 0.75 the error was sufficiently low and no further refinement was needed. This is expected because the initial power level is a linear scaling factor, as can be seen from the solution in Equation (60). For the decay constant (λ), the algorithm assumes a constant nominal value because only 3 nodes were sampled. The root node results in the nominal value but when testing on its children (boundary nodes), the error was sufficiently low to stop any further refinement along this dimension. The neutron generation time (Λ) also has minimal effect on the power evolution, which is evident in the low number of selected unique nodes. As expected, the model is most sensitive to the reactivity (ρ_1) and the delayed neutron fraction (β), which the adaptive algorithm could recognize by adding more points along these dimensions.

Fig. 7 shows a projection of the evaluated points in Setting 1 on the $\rho_1 - \beta$ plane. This figure illustrates the region where the algorithm selected the most points. The region corresponding to higher reactivity and lower delayed neutron fraction was considered the most important (mathematically $\rho_1 \approx \beta$). This was expected because a lower fraction of the delayed neutrons means the reactor responds stronger to perturbations with instantaneous prompt neutrons. Reactors with lower β are closer to the instability and harder to control. Higher reactivity insertion ρ_1 also causes the neutron population inside

Table 3
Number of unique nodes in each dimension selected by the POD-Adaptive algorithm in the Point Kinetics problem.

	P_0	λ	Λ	ρ_1	β
Number of nodes in Setting 1	5	3	5	31	29
Number of nodes in Setting 2	5	3	4	7	7



(a) Maximum relative L_2 error for the strongly nonlinear Setting 1.

(b) Maximum relative L_2 error for the weakly nonlinear Setting 2.

Fig. 6. Point Kinetics: Maximum relative L_2 error as a function of the number of evaluations. Classical and adaptive algorithms are compared. The effect of reducing all tolerances of the adaptive algorithm by a factor of 10 is also shown (marked with $\zeta_{\text{rel}} = 10^{-3}$).

the reactor to multiply much faster. Additionally, Fig. 6 shows the effect of reducing the tolerances by a factor of 10 (i.e., $\zeta_{\text{rel}} = 10^{-3}$, $\zeta_{\text{abs}} = 10^{-5}$, $\gamma_{\text{int}} = 5 \times 10^4$). The algorithm produced a ROM that achieved the required tolerance with a higher number of model evaluations. Nevertheless, the adaptive algorithm in this case reduced the number of model evaluations compared to the classical approach by a factor of 6.

For Setting 2, both the adaptive and the classical models reached the required tolerance at similar rates, as shown in Fig. 6b. The adaptivity had no advantage here since the problem is almost linear. Reducing the threshold brings the adaptive method closer to the classical performance because more points are admitted in the ROM per iteration. For example, Fig. 6b shows that both the adaptive and the classical approaches sampled 61 points by the second iteration. However, when tested on the random points, the classical approach had a lower error compared to the adaptive because the adaptive marked some of these points as inactive and did not include them in the ROM. Although Fig. 6b shows the error to be lower than the tolerance by the second iteration, the adaptive algorithm converged after three iterations because the error estimate ($\epsilon^k_{\max}$) was still slightly above the tolerance at the second iteration. Reducing the threshold caused the adaptive to include all 61 points in the ROM, which also resulted in an error almost following the classical model for all iterations. Nevertheless, the adaptive algorithm is able to stop the sampling and converge to the required relative tolerance of 10^{-3} after 367 points while the classical approach can only stop after a predefined number of points, which in this case was 801 points.

Fig. 8a and Fig. 9a show the distribution of the relative L_2 error resulting from the tests for Setting 1 and Setting 2, respectively. In Setting 1, 99.98% of the points resulted in an error less than the tolerance of 1% while in Setting 2 all tested points resulted in errors less than the set tolerance. The maximum error found in Setting 1 was 1.05%. This point is simulated and compared to the reference solution in Fig. 8b. The figure shows the power increase at $t = 0$ due to the positive reactivity ($\rho_1 = 0.00375$), which is then sharply reduced at $t = 10^{-2}$s due to the insertion of the shutdown rods ($\rho_2 = -5\beta$). In Setting 2, the maximum error was 0.75% which is shown in Fig. 9b along with the corresponding reference solution. The initial power increase is due to a perturbation of $\rho_1 = 0.00293$ and the decrease is again due to the shutdown rods. The POD-Adaptive model produced the simulations for the 10000 points in 3 seconds while the reference model required 15 seconds.

In order to test the effect of the greediness parameter, we rerun the algorithm with reduced values of μ and tested the resultant ROM models on the same 10000 random points. We compare the number of model evaluations, the percentage of these evaluations utilized by the ROM, the maximum error found and the percentage of tested points with errors below the set tolerance. The results are summarized in Table 4. As expected, reducing the greediness parameter decreased the number of model evaluations and improved the utilization of these points. It can also be seen that this improvement compromised

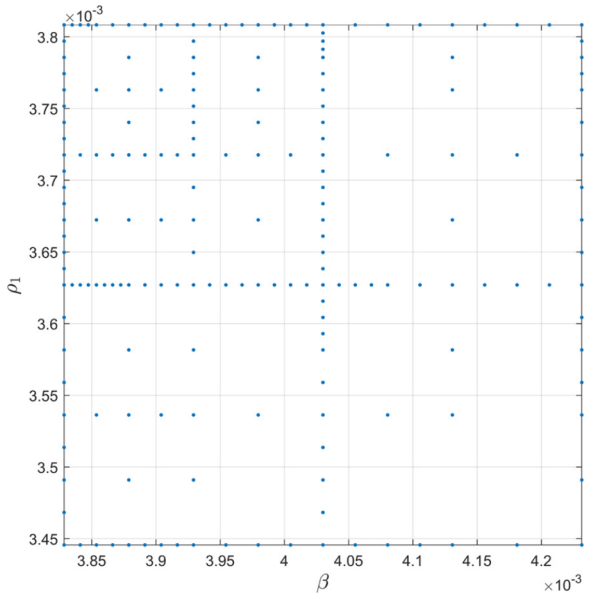
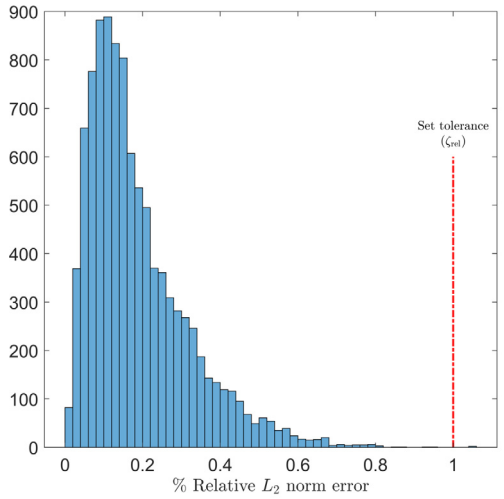
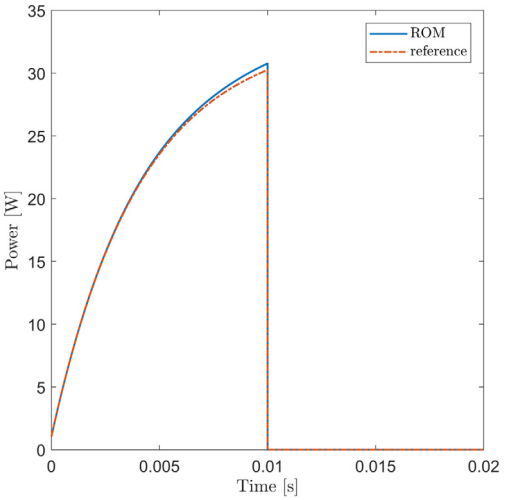


Fig. 7. Point Kinetics Setting 1: Projection of the sampled points on the $\rho_1 - \beta$ plane with $\zeta_{\text{rel}} = 10^{-2}$.



(a) Histogram of the relative L_2 error. 99.98% of the points resulted in an error less than the set tolerance.



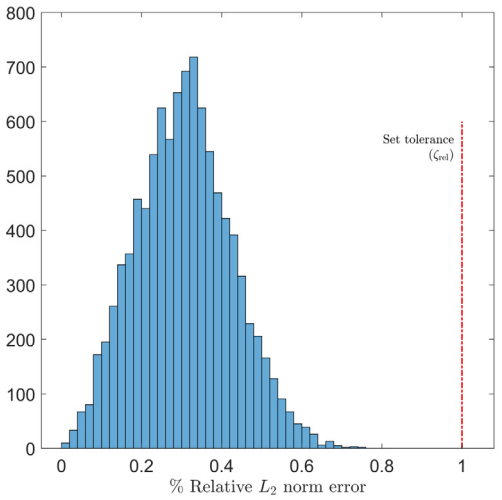
(b) Simulation comparisons between the reference model and POD-Adaptive ROM at the point of maximum error (1.05%).

Fig. 8. Point Kinetics Setting 1: Histogram and simulation of the point of maximum error, which resulted from testing the ROM model on 10000 randomly generated points. The POD-Adaptive model produced the simulations faster than the reference model by a factor of 5.

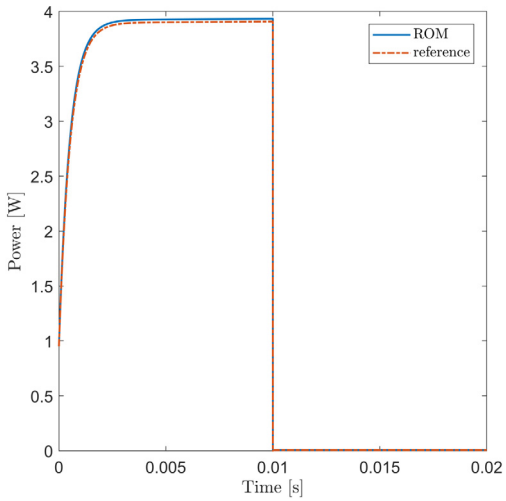
the accuracy of the model to some extent. For example, selecting $\mu = 0$ increased the maximum error for setting 1 to 4.7% and decreased the number of points below the tolerance to 89%. However, this model was built with only about 17% of the points needed by the default (greedy) model. The table also shows a case of reducing the interpolation threshold γ_{int} to 10^{-3} while $\mu = 0$. The model, in this case, matched the default model in terms of accuracy with 58% less points. The significant reduction in the number of points with decreasing μ can also be seen in Setting 2 and similar conclusions can be drawn.

5.2. Test Case 2: diffusion

The diffusion equation has application in many scientific disciplines. We consider the time independent diffusion equation with a space dependent diffusion coefficient and a removal term



(a) Histogram of the relative L_2 error. 100% of the points resulted in an error less than the set tolerance.



(b) Simulation comparisons between the reference model and POD-Adaptive ROM at the point of maximum error (0.75%).

Fig. 9. Point Kinetics Setting 2: Histogram and simulation of the point of maximum error, which resulted from testing the ROM model on 10000 randomly generated points. The POD-Adaptive model produced the simulations faster than the reference model by a factor of 5.

Table 4
Point Kinetics: Results of comparing different ROM models by varying the greediness parameter μ.

	μ	Number of model evaluations	% of utilized points	Maximum relative L_2 error	% of points $< \zeta_{rel}$
Setting 1	1	944	15.9%	1.05%	99.98%
	0.5	529	26.6%	1.11%	99.98%
	0	163	60.12%	4.78%	89.38%
	0[a]	401	66.83%	1.06%	99.98%
Setting 2	1	139	15.8%	0.75%	100%
	0.5	91	24.1%	0.75%	100%
	0	43	46.5%	0.96%	100%
	0[a]	77	28.6%	0.27%	100%

[a] $\gamma_{int} = 10^{-3}$.

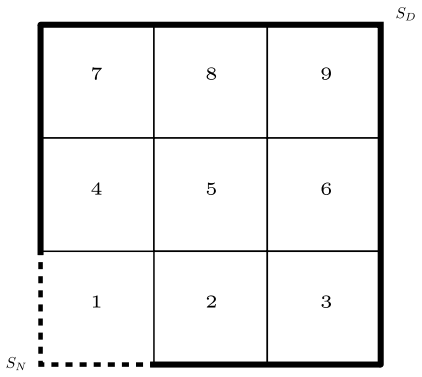
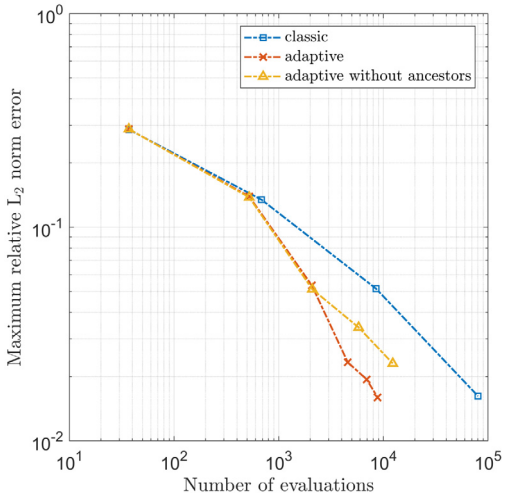


Fig. 10. Domain of the diffusion problem showing the numbered 9 regions and the boundary segments S_D and S_N.

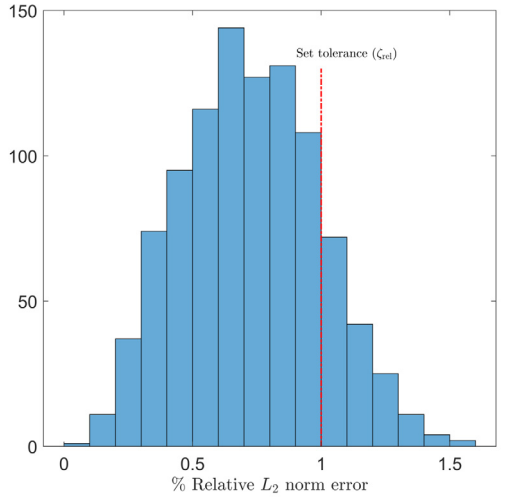
$$\nabla \cdot D(\mathbf{r}) \nabla \phi(\mathbf{r}) + \alpha(\mathbf{r}) \phi(\mathbf{r}) = Q(\mathbf{r}), \tag{65}$$

where $\phi(\mathbf{r})$ is the diffusing material at location $\mathbf{r}$, $D(\mathbf{r})$ is the diffusion coefficient at $\mathbf{r}$, $\alpha(\mathbf{r})$ is the removal coefficient, and $Q(\mathbf{r})$ is the source function. We consider a checkerboard domain with 9 regions as shown in Fig. 10. Each region is considered to be homogeneous with constant properties across the region. Only the lower left corner (Region 1) has a uniform unit source, i.e.,

$$Q(\mathbf{r}) = 1 \qquad \forall\, \mathbf{r} \in \text{Region } 1, \tag{66}$$



(a) Maximum relative L_2 error as a function of the number of evaluations for the diffusion problem.



(b) Histogram of the relative L_2 error. 84.4% of the points resulted in an error less than the set tolerance of 1%.

Fig. 11. Diffusion: Maximum relative error per iteration after the test on 1000 random points and histogram of the relative error at the same points for the POD-Adaptive ROM after convergence.

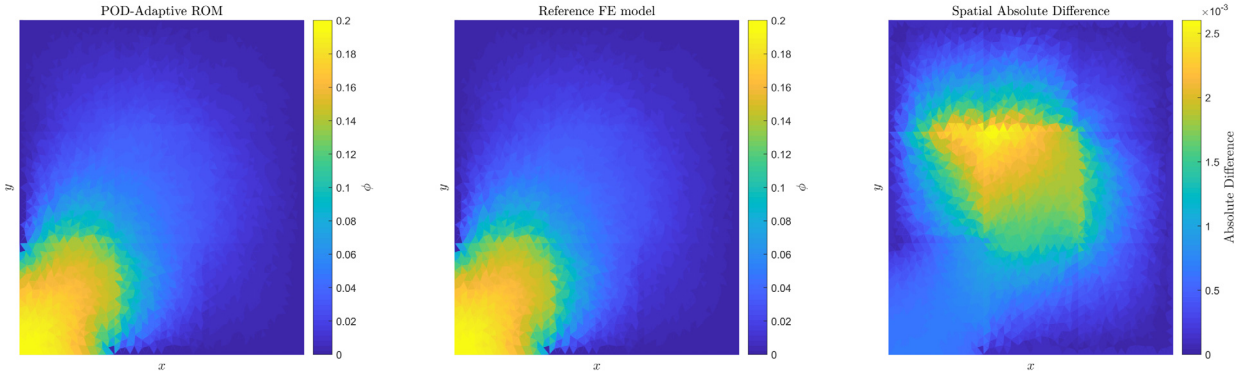
while the source is zero for all other regions. The boundary conditions were taken to be

$$\begin{aligned} \phi(\mathbf{r}) &= 0 \qquad \text{on } S_D, \\ \hat{n} \cdot \nabla\phi(\mathbf{r}) &= 0 \qquad \text{on } S_N, \end{aligned} \tag{67}$$

where $\hat{n}$ is the outgoing normal vector, S_N is the outer boundary segment for the source region and S_D is the outer boundary segment for all other regions as shown in Fig. 10. The diffusion equation was solved on a two dimensional plane (i.e., $\mathbf{r} = (x, y)$). We consider the model to be parametrized with respect to the diffusion and removal coefficients. Therefore, we have a total of 18 dimensions corresponding to the diffusion coefficients (D_n) and removal coefficients (α_n) within every region n. The diffusion coefficients were taken to be in the range [0.3 1.7], and the removal coefficient in [0.0075 0.0425], where the nominal value of the solution is the centre of the range. The equation was solved using a Finite Element (FE) code, which was considered to be the reference model. We then build a ROM for the FE model using the POD-Adaptive algorithm with relative tolerance (ζ_{rel}) set to 10^{-2}, absolute tolerance (ζ_{abs}) set to 10^{-4}, interpolation threshold (γ_{int}) set to 5×10^{-3}, truncation threshold (γ_{tr}) set to 10^{-12} and a value of 1 for the greediness parameter μ.

The algorithm converged after 8815 sampling points. As a benchmark showing the progress of the error during the construction stage, the model was tested on 1000 randomly generated points after each iteration. Fig. 11a shows the maximum relative error per iteration as a function of the number of evaluations. The classical sparse grids error is also shown in Fig. 11a for comparison. Both the adaptive and the classical sparse grids needed a relatively large number of model evaluations due to the high dimensionality of the problem. However, the number of model evaluations used by the adaptive algorithm was less by a factor of 10 compared to the classical approach. The adaptive algorithm recognized that the model was more sensitive to property changes in the source region and sampled more points there compared to other regions. Fig. 13 shows the total number of unique nodes selected in each region (i.e., the number of unique diffusion coefficient nodes and the number of unique removal coefficient nodes). The same figure shows that the algorithm placed more importance on the diffusion coefficient than the removal coefficient. In fact, the algorithm considered the removal coefficient to be constant in all regions at the nominal value except in the source region where it was considered to be a linear factor. Moreover, the projection of the points on the plane of diffusion and removal coefficients in the source region (Fig. 14) shows that more importance was given to lower values of the diffusion coefficient because the solution is smoother for higher diffusivity.

The POD-Adaptive algorithm sampled 8815 points but only 336 points were included in the ROM. The small fraction of important points indicates that the algorithm was oversampling the model, which can be attributed to the higher dimensionally of this problem. Despite that the algorithm eventually recognized the important dimensions, all dimensions needed to be searched at the initial steps, which increased the number of evaluations. As a test for the refinement criterion, we run the algorithm with the same tolerances again but by not including the ancestor points. Not including the ancestor points is equivalent to marking all candidate points C^k as important without considering their ancestry first (i.e., Equation 33 becomes $Z^k = C^k$). The algorithm without the ancestors converged after 12313 sampling points. The same set of 1000 random points used to test the default ROM was used to test the ROM resulting from this strategy. The resulting error per iteration is shown in Fig. 11a. From this figure, it can be seen that both strategies were equivalent for the first three



(a) Solution of the POD-Adaptive ROM at the point with maximum error (1.6%).

(b) Solution of the reference FE code at the point with maximum error.

(c) The spatial absolute difference between the ROM and the FE code at the point with maximum error.

Fig. 12. Diffusion: Comparison between the POD-Adaptive ROM and the reference model at the point with maximum error resulting from the test on 1000 random points. The ROM produced the simulation faster than the reference model by a factor of 40.

iterations. However, the strategy of including the ancestors reduced the error further and eventually converged with less model evaluations. This indicates that by not including the ancestor points in this problem, some behaviour of the reference model is overlooked, which results in increased model evaluations to compensate for the missing dynamics.

The solution resulting from the ROM at the point with the maximum error is shown in Fig. 12a and the reference solution at the same point is shown in Fig. 12b and the absolute difference is in Fig. 12c. The POD-Adaptive model outperformed the reference model in the time required to simulate 1000 points by a factor of 40. This factor represents the computational time required to simulate 1000 points with the reference model over the computational time required to simulate the same points with the ROM model. However, the histogram of the error in Fig. 11b shows that only 15.6% of the tested points resulted in an error above the required tolerance. At this stage, the validation algorithm was started in order to reduce this percentage. We initiated the validation stage with the top 20 random points corresponding to the highest relative error. The validation algorithm required an additional 14532 points before convergence. However, only 21% of these points were included in the ROM. Then, we tested this model with a new set of 1000 random points. With this model, all tested points were less than the tolerance with the maximum relative error found to be 0.6%. With the additional validation points, the total number of model evaluations was 23,347.

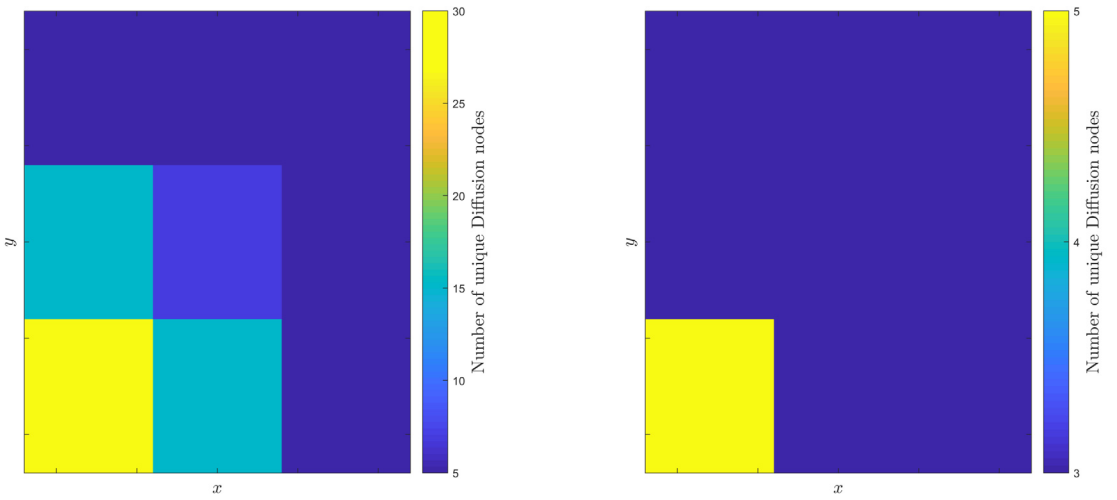
In Table 5, we compare models built with different values of the greediness parameter μ. Evidently, a greedy algorithm is an overkill for this problem because the percentage of utilized points was very small. By choosing a μ value of 0, the number of model evaluations was massively reduced. This can be explained by the fact that most dimensions of this problem were linear or only mildly non-linear. From Fig. 13, we can deduce that, out of the 18 dimensions, 8 were considered constant (only 3 nodes were sampled along each of them), and 6 were considered linear (with 5 nodes). The greedy setting ($\mu = 1$), however, caused the algorithm to constantly search along these irrelevant dimensions as well in every iteration. By setting $\mu = 0$, the algorithm disregarded every point that had a backward point that was seen to be unimportant in previous iterations. Naturally, such non-greedy strategy resulted in overlooking some of the dynamics and the accuracy was deteriorated somewhat (e.g. 3.1% maximum error vs 1.6%). Nevertheless, reducing the value for the interpolation threshold compensated for the lower accuracy and resulted in a ROM model that outperformed the greedy model both in terms of the accuracy and the number of evaluations.

5.3. Test Case 3: modified Morris function

The Morris function is a single valued function in 20 dimensions that was developed to test optimization algorithms [43]. In order to test the POD-Adaptive algorithm, we propose a modified version by having the output as a field defined over a 2-dimensional plane. We propose the modified Morris function as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{20} \beta_i w_i \sin(i\pi x_1) + \sum_{i<j}^{20} \beta_{i,j} w_i w_j \sin(i\pi x_1)\sin(j\pi x_2) + \sum_{i<j<l}^{20} \beta_{i,j,l} w_i w_j w_l \sin(i\pi x_1)\sin(j\pi x_2)\cos(l\pi x_1) + \sum_{i<j<l<s}^{20} \beta_{i,j,l,s} w_i w_j w_l w_s \sin(i\pi x_1)\sin(j\pi x_2)\cos(l\pi x_1)\cos(s\pi x_2), \quad (68)$$

where $x_1, x_2 \in [0, 1]$, and



(a) Number of Diffusion coefficient nodes in each region.

(b) Number of Removal coefficient nodes in each region.

Fig. 13. Diffusion: number of nodes in each region selected by the adaptive algorithm showing the importance of the regions as discovered by the algorithm.

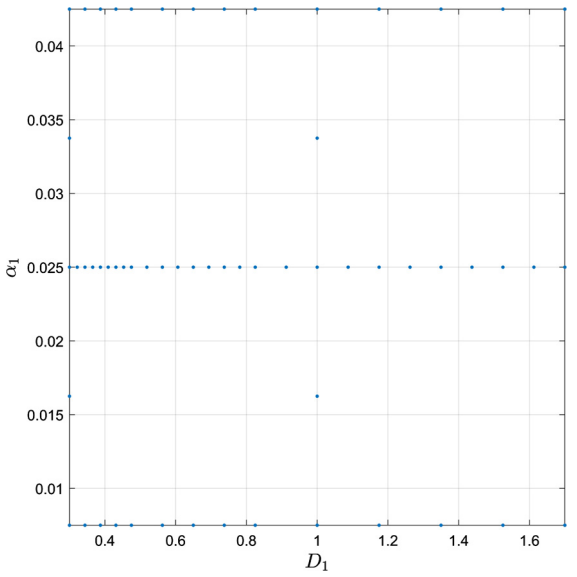


Fig. 14. Diffusion: Projection of the sampled points on the plane of the diffusion and removal coefficients ($D_1 - \alpha_1$ plane) in the source region.

$$w_i = \begin{cases} 2\left(\frac{1.1\lambda_i}{\lambda_i + 0.1} - 0.5\right), & \text{for } i = 3, 5, 7 \\ 2\left(\lambda_i - 0.5\right), & \text{otherwise,} \end{cases} \tag{69}$$

where λ_i is the input parameter defined uniformly on the interval $[0, 1]$ and the constants β defined as

$$\beta_i = \begin{cases} 20, & \text{for } i = 1, \ldots, 10 \\ (-1)^i, & \text{otherwise,} \end{cases} \qquad \beta_{i,j,l} = \begin{cases} -10, & \text{for } i, j, l = 1, \ldots, 5 \\ 0, & \text{otherwise,} \end{cases}$$

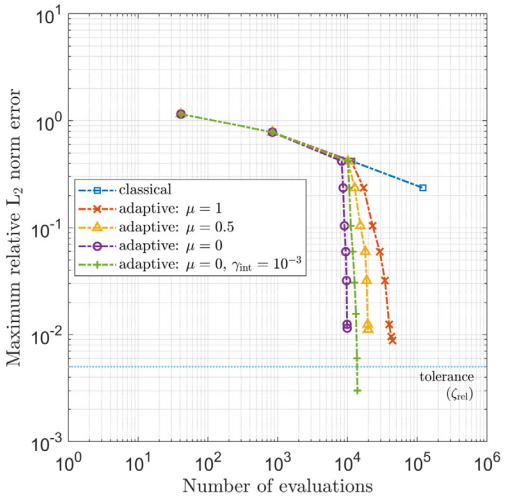
$$\beta_{i,j} = \begin{cases} -15, & \text{for } i, j = 1, \ldots, 6 \\ (-1)^{i+j}, & \text{otherwise,} \end{cases} \qquad \beta_{i,j,l,s} = \begin{cases} 5, & \text{for } i, j, l, s = 1, \ldots, 4 \\ 0, & \text{otherwise.} \end{cases} \tag{70}$$

The modified Morris function is parametrized on 20 dimensions corresponding to λ_i. The function is linear in all dimensions except three: λ_3, λ_5, and λ_7. A ROM was built using the POD-Adaptive algorithm with relative tolerance (ζ_{rel}) equal to 5×10^{-3}, absolute tolerance (ζ_{abs}) set to 5×10^{-4}, interpolation threshold (γ_{int}) equal to 5×10^{-3}, truncation threshold (γ_{tr}) equal to 10^{-12}, and the greediness parameter $\mu = 1$.

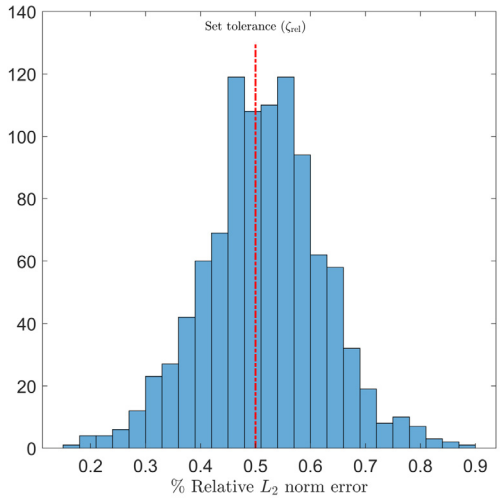
Table 5
Diffusion: Results of comparing different ROM models by varying the greediness parameter μ.

μ	Number of model evaluations	% of utilized points	Maximum relative L_2 error	% of points $< \zeta_{\text{rel}}$
1	8815	3.8%	1.6%	84.4%
0.5	1720	9.8%	2.3%	77.4%
0	385	41.6%	3.1%	41.6%
0[a]	1577	40.0%	1.0%	99.9%

[a] $\gamma_{\text{int}} = 10^{-3}$.



(a) Maximum relative L_2 error as a function of the number of evaluations for the modified Morris problem.



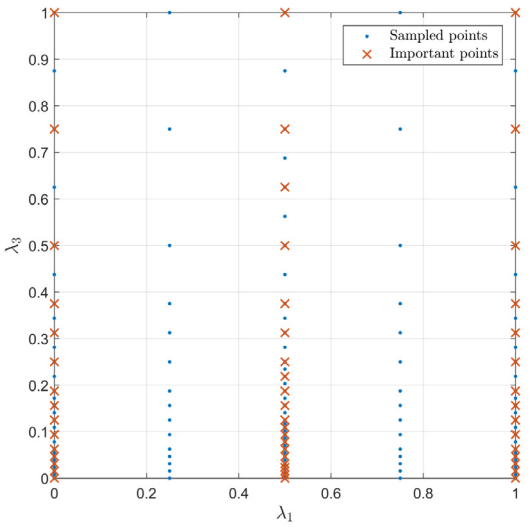
(b) Histogram of the relative L_2 error. 44.3% of the points resulted in an error less than the set tolerance of 0.5%.

Fig. 15. Modified Morris: Maximum relative error per iteration after the test on 1000 random points and histogram of the relative error at the same points for the POD-Adaptive ROM after convergence.

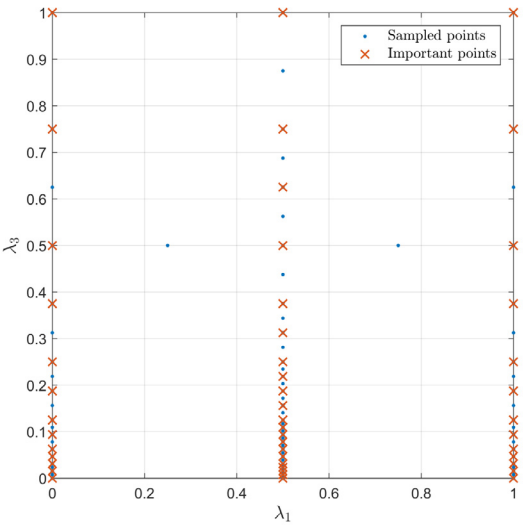
This problem was found to be challenging to the algorithm due to the higher dimensionality and the oscillatory nature of the solution. The algorithm converged after sampling 44,297 points, but only 1874 points were admitted to the ROM, which implied that the algorithm was oversampling. The produced ROM model was tested on 1000 random points after each iteration. Fig. 15a shows the maximum relative error resulting from the tests as a function of the number of evaluations. For comparison, the error from a ROM with classical sparse grid sampling is also shown on the same figure. The classical approach requires significantly more points. The main reason for the difference in performance is that the POD-Adaptive algorithm correctly recognized that λ_3, λ_5, λ_7 are nonlinear and selected 31 unique nodes for λ_3, 31 nodes for λ_5, and 29 nodes for λ_7. For all other dimensions, 5 nodes were sampled, which is an accurate linear assumption.

Moreover, lower values of λ_i were found to be more important than higher values. This is illustrated in Fig. 16, which shows the projection of the sampled points on the $\lambda_1 - \lambda_3$ plane. The selected important points are also marked on the figure. For λ_1, the algorithm selected 5 nodes and marked only 3 as important (the centre root node and the boundaries). On the other hand, λ_3 is nonlinear and was more refined in the lower region. Identifying this important region enabled the algorithm to reduce the number of evaluations considerably compared to the classical approach. Comparison between the ROM and the reference solution at the point of maximum error is shown in Fig. 17a and Fig. 17b while Fig. 17c shows the absolute difference between the two solutions. Histogram of the error (Fig. 15b) shows that the error almost followed a normal distribution around the set tolerance but only 44.3% of the points were strictly below the tolerance. The reference model for the modified Morris function was implemented with nested loops which required about 3 seconds to complete a single evaluation. The time to evaluate 1000 points with the reference model was about 3241 seconds while the ROM model needed only 33 seconds to evaluate the same points, which is a reduction by about a factor of 100 in computational time.

On Fig. 15a, we also show the effect of reducing the greediness of the algorithm (decreasing μ). Table 6 summarizes the differences in performance between these models. As expected, lower values of the μ converged faster but with reduced accuracy. We notice again that the non-greedy ROM model with $\mu = 0$ and reduced interpolation threshold ($\gamma_{\text{int}} = 10^{-3}$) was the best compromise between accuracy and efficiency. In order to highlight the difference in the sampling scheme with respect to varying μ, Fig. 16 shows the projection of the sampled points on the $\lambda_1 - \lambda_3$ plane for the two cases of greedy $\mu = 1$ and non-greedy $\mu = 0$. At the boundaries of λ_1, we notice that, for every refinement along λ_3, the algorithm also searches the children of these boundaries (nodes 0.25, and 0.75). However, λ_1 is linear and these points never reveal any significant dynamics to be marked important. This search creates an unnecessary line of points at the inner nodes $\lambda_1 = 0.25$,

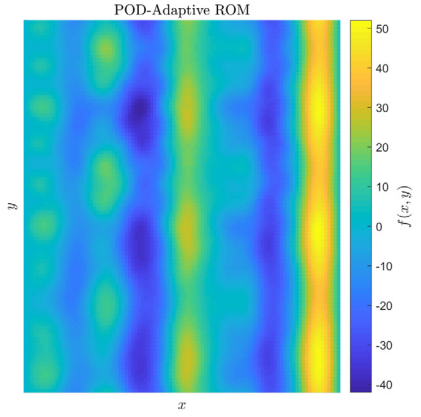


(a) Projection of the sampled points for the greedy algorithm ($\mu = 1$).

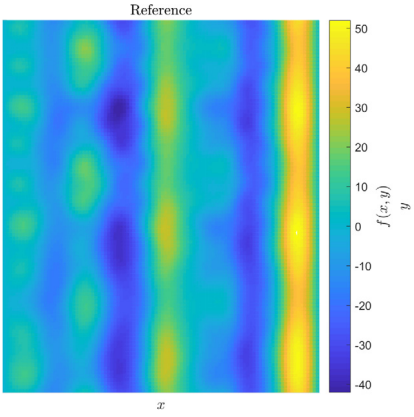


(b) Projection of the sampled points for the non-greedy algorithm ($\mu = 0$).

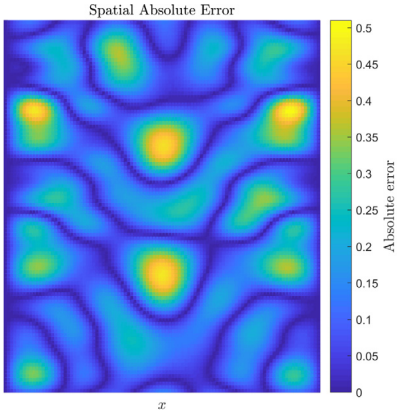
Fig. 16. Modified Morris: Projection of the sampled points on the plane of $\lambda_1 - \lambda_3$ comparing the selected points between the greedy ($\mu = 1$) and the non-greedy ($\mu = 0$) algorithms. The selected important points ($\mathcal{X}^k$) are marked with crosses.



(a) Simulation of the POD-Adaptive ROM at the point with maximum error (0.88%).



(b) Solution of the reference model at the point with maximum error.



(c) The spatial absolute difference between the ROM and the reference model at the point with maximum error.

Fig. 17. Modified Morris: Comparison between the POD-Adaptive ROM and the reference model at the point with maximum error resulting from the test on 1000 random points. The ROM produced the simulation faster than the reference model by a factor of 100.

Table 6
Modified Morris: Results of comparing different ROM models by varying the greediness parameter μ.

μ	Number of model evaluations	% of utilized points	Maximum relative L_2 error	% of points $< \zeta_{\text{rel}}$
1	44,296	3.8%	0.88%	44.3%
0.5	19,852	8.9%	1.11%	14.8 %
0	9928	17.2%	1.15%	13.3%
0[a]	13,872	22.4%	0.3%	100%

[a] $\gamma_{\text{int}} = 10^{-3}$.

Table 7
Summary of results for all test cases using $\mu = 0$ and $\gamma_{\text{int}} = 10^{-3}$.

		Dimensionality	Required number of model evaluations	Speed-up factor*	Maximum relative L_2 error	% of points $< \zeta_{\text{rel}}$
Point Kinetics	Setting 1	5	401	5	1.06%	99.9%
	Setting 2	5	77	5	0.27%	100%
Diffusion		18	1577	40	1.0%	99.9%
Modified Morris		20	13,872	100	0.3%	100%

* The speed-up factor is the ratio of the computational time required to simulate 1000 points with the reference model to the computational time required to simulate the same points with the ROM model.

and 0.75. Of course, this line is repeated for all linear dimensions, which is one of the causes for the additional cost in model evaluations compared to the non-greedy setting. For the case of $\mu = 0$, this line is eliminated because the algorithm stopped searching dimension λ_1 after this dimension was found to be linear along the line $\lambda_3 = 0.5$. This successfully reduced the number of model evaluations without loss in accuracy.

6. Conclusions

We have presented a practical approach for integrating the locally adaptive sparse grid technique with the POD method to develop a nonintrusive ROM algorithm. The local adaptivity provides an effective sampling scheme for the POD snapshots while the hierarchical interpolant builds a surrogate model for the POD amplitudes. For increased robustness of the refinement strategy, the ancestor points are prioritized when selecting the important points. We also introduced a greediness parameter that provides additional control of the number of model evaluations during the construction phase. Additionally, a validation algorithm was presented with the purpose of not only to certify the model after convergence but also enhancing it once a discrepancy is detected. In addition, an efficient way of updating the surpluses with every POD modes update was presented, which reduces the computational burden of recalculating the interpolant after each iteration.

The developed POD-Adaptive algorithm was tested numerically on three applications. In all tests, the algorithm considered the system as a black box without any knowledge of the equations being solved. Table 7 summarizes the results of all test cases for values of $\mu = 0$ and $\gamma_{\text{int}} = 10^{-3}$. Every ROM model was then tested on random points that were not part of the training set. In all test cases, the built ROM model was able to reduce the computational time and provide solutions in good agreements with the reference model (errors were within the set tolerances). The computational time to perform these simulations was reduced by a factor of 5 in the Point kinetics case, by a factor of 40 in the Diffusion case, and by a factor of 100 in the modified Morris case. We compared the POD-adaptive with the classical (non-adaptive) sparse grids technique. The POD-Adaptive reduced the number of model evaluations compared to the classical approach significantly. This was most evident in the nonlinear cases (i.e., Point Kinetics Setting 1, Diffusion, and the modified Morris function). For linear and smooth problems, as in the Point Kinetics Setting 2, both the adaptive and the classical approaches provided similar performance. For all applications not only did the algorithm recognize the important dimensions but it also recognized the important range within that dimension. Therefore, although the method is locally adaptive, it is also implicitly (globally) dimension adaptive.

The greediness parameter reduced the total number of model evaluations and improved the utilization of the sampled points significantly. As expected, problems with higher dimensions tend to have a smaller percentage of utilized points. The models for the Diffusion and the modified Morris problems utilized around 4% of the points whereas the lower dimensional Point Kinetics model included 15% of the points. This indicates that the algorithm explored the parameter space for these problems by oversampling the reference model. By reducing the greediness parameter, the quantity and utilization of the sampled point were improved. It was shown that this improvement compromised the accuracy of the model to some extent. Nevertheless, the lost accuracy was recovered with a reduced interpolation threshold (γ_{int}), and a compromise between accuracy and efficiency was achieved. While the risk of overlooking localized dynamics is increased for a reduced μ, the developed validation stage can reveal any missing dynamics and update the ROM model in such cases.

The algorithm can still be improved with the regards to the sampling and utilization of points. For large scale, very high-dimensional problems (e.g. in the hundreds), the main limitation of the algorithm is the large number of model evaluations potentially required. Knowledge of the physics of the system can be helpful to screen and reduce the dimensionality a priori. Additionally, limiting the range of the input parameters will restrict the space in which the algorithm searches the domain. We can also use a different unidimensional rule that limits the scope of the search to specific important regions. For example, we can choose a rule with no boundary points if we know that the sensitivity of the model to variations at the boundaries are small. Moreover, using higher order basis functions is an interesting area to investigate for reducing the number of model evaluation and improving the utilization of the points. These areas are subjects of future research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors acknowledge the support of King Abdulaziz City for Science and Technology (KACST) for this work.

References

[1] P. Benner, S. Gugercin, K. Willcox, A survey of projection-based model reduction methods for parametric dynamical systems, SIAM Rev. 57 (2015) 483–531.
[2] V. Buljak, Inverse Analyses with Model Reduction: Proper Orthogonal Decomposition in Structural Mechanics, Springer, Berlin, 2011.
[3] J.L. Lumley, The structure of inhomogeneous turbulent flows, in: Atmospheric Turbulence and Radio Wave Propagation, vol. 790, 1967, pp. 166–178.
[4] R. Bourguet, M. Braza, A. Dervieux, Reduced-order modeling of transonic flows around an airfoil submitted to small deformations, J. Comput. Phys. 230 (2011) 159–184.
[5] A. Placzek, D.-M. Tran, R. Ohayon, A nonlinear POD-Galerkin reduced-order model for compressible flows taking into account rigid body motions, Comput. Methods Appl. Mech. Eng. 200 (2011) 3497–3514.
[6] C.W. Rowley, T. Colonius, R.M. Murray, Model reduction for compressible flows using POD and Galerkin projection, Physica D 189 (2004) 115–129.
[7] M. Bergmann, L. Cordier, Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models, J. Comput. Phys. 227 (2008) 7813–7840.
[8] Z. Wang, I. Akhtar, J. Borggaard, T. Iliescu, Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison, Comput. Methods Appl. Mech. Eng. 237–240 (2012) 10–26.
[9] J. Baiges, R. Codina, S.R. Idelsohn, Reduced-order modelling strategies for the finite element approximation of the incompressible Navier-Stokes equations, in: Computational Methods in Applied Sciences, Springer International Publishing, 2014, pp. 189–216.
[10] A.G. Buchan, C.C. Pain, F. Fang, I.M. Navon, A POD reduced-order model for eigenvalue problems with application to reactor physics, Int. J. Numer. Methods Eng. 95 (2013) 1011–1032.
[11] A. Sartori, D. Baroli, A. Cammi, D. Chiesa, L. Luzzi, R. Ponciroli, E. Previtali, M.E. Ricotti, G. Rozza, M. Sisti, Comparison of a modal method and a proper orthogonal decomposition approach for multi-group time-dependent reactor spatial kinetics, Ann. Nucl. Energy 71 (2014) 217–229.
[12] S. Lorenzi, A. Cammi, L. Luzzi, G. Rozza, A reduced order model for investigating the dynamics of the gen-IV LFR coolant pool, Appl. Math. Model. 46 (2017) 263–284.
[13] A. Quarteroni, G. Rozza, A. Manzoni, Certified reduced basis approximation for parametrized partial differential equations and applications, J. Math. Ind. 1 (2011) 3.
[14] G. Rozza, D.B.P. Huynh, A.T. Patera, Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations, Arch. Comput. Methods Eng. 15 (2008) 229–275.
[15] M.A. Grepl, A.T. Patera, A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations, Modél. Math. Anal. Numér. 39 (2005) 157–181.
[16] H.V. Ly, H.T. Tran, Modeling and control of physical processes using proper orthogonal decomposition, Math. Comput. Model. 33 (2001) 223–236.
[17] C. Audouze, F.D. Vuyst, P.B. Nair, Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations, Numer. Methods Partial Differ. Equ. 29 (2013) 1587–1628.
[18] J. Hesthaven, S. Ubbiali, Non-intrusive reduced order modeling of nonlinear problems using neural networks, J. Comput. Phys. 363 (2018) 55–78.
[19] N. Nguyen, J. Peraire, Gaussian functional regression for output prediction: model assimilation and experimental design, J. Comput. Phys. 309 (2016) 52–68.
[20] A. Giunta, S. Wojtkiewicz, M. Eldred, Overview of modern design of experiments methods for computational simulations (invited), in: 41st Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, 2003.
[21] M. Guénot, I. Lepot, C. Sainvitu, J. Goblet, R. Filomeno Coelho, Adaptive sampling strategies for non-intrusive pod-based surrogates, Eng. Comput. 30 (2013) 521–547.
[22] S.A. Smolyak, Quadrature and interpolation formulas for tensor products of certain classes of functions, in Dokl. Akad. Nauk SSSR 4 (1963) 240–243.
[23] V. Barthelmann, E. Novak, K. Ritter, High dimensional polynomial interpolation on sparse grids, Adv. Comput. Math. 12 (2000) 273–288.
[24] B. Peherstorfer, Model Order Reduction of Parametrized Systems with Sparse Grid Learning Techniques, Ph.D. thesis, Technische Universität München, München, 2013.
[25] D. Xiao, F. Fang, A. Buchan, C. Pain, I. Navon, A. Muggeridge, Non-intrusive reduced order modelling of the Navier–Stokes equations, Comput. Methods Appl. Mech. Eng. 293 (2015) 522–541.
[26] H.C. Elman, C.W. Miller, E.T. Phipps, R.S. Tuminaro, Assessment of collocation and Galerkin approaches to linear diffusion equations with random data, Int. J. Uncertain. Quantificat. 1 (2011).
[27] T. Gerstner, M. Griebel, Dimension-adaptive tensor-product quadrature, Computing 71 (2003) 65–87.
[28] P. Chen, A. Quarteroni, A new algorithm for high-dimensional uncertainty quantification based on dimension-adaptive sparse grid approximation and reduced basis methods, J. Comput. Phys. 298 (2015) 176–193.
[29] C. Zenger, Sparse grids, in: W. Hackbusch (Ed.), Parallel Algorithms for Partial Differential Equations, in: Notes on Numerical Fluid Mechanics, vol. 31, Vieweg, 1990, pp. 241–251.
[30] H.-J. Bungartz, M. Griebel, Sparse grids, Acta Numer. 13 (2004) 147–269.
[31] M. Griebel, Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences, Computing 61 (1998) 151–179.
[32] D. Pflüger, B. Peherstorfer, H.-J. Bungartz, Spatially adaptive sparse grids for high-dimensional data-driven problems, J. Complex. 26 (2010) 508–522.
[33] 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.
[34] X. Ma, N. Zabaras, An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations, J. Comput. Phys. 229 (2010) 3884–3915.
[35] J.D. Jakeman, R. Archibald, D. Xiu, Characterization of discontinuities in high-dimensional stochastic problems on adaptive sparse grids, J. Comput. Phys. 230 (2011) 3977–3997.
[36] H.-J. Bungartz, S. Dirnstorfer, Multivariate quadrature on adaptive sparse grids, Computing 71 (2003) 89–114.
[37] J. Brumm, S. Scheidegger, Using adaptive sparse grids to solve high-dimensional dynamic models, Econometrica 85 (2017) 1575–1612.
[38] P. Holmes, J.L. Lumley, G. Berkooz, Turbulence, Coherent Structures, Dynamical Systems and Symmetry, Cambridge University Press, Cambridge, 1996.
[39] F. Nobile, R. Tempone, C.G. Webster, A sparse grid stochastic collocation method for partial differential equations with random input data, SIAM J. Numer. Anal. 46 (2008) 2309–2345.
[40] A. Klimke, Uncertainty Modeling using Fuzzy Arithmetic and Sparse Grids, Ph.D. thesis, Universität Stuttgart, Stuttgart, 2006.
[41] J.J. Duderstadt, L.J. Hamilton, Nuclear Reactor Analysis, John Wiley & Sons, Inc., 1976.
[42] Z. Perkó, L. Gilli, D. Lathouwers, J.L. Kloosterman, Grid and basis adaptive polynomial chaos techniques for sensitivity and uncertainty analysis, J. Comput. Phys. 260 (2014) 54–84.
[43] M.D. Morris, Factorial sampling plans for preliminary computational experiments, Technometrics 33 (1991) 161–174.