LECTURE NOTES IN COMPUTATIONAL SCIENCE AND ENGINEERING 88

Jochen Garcke · Michael Griebel *Editors*

Sparse Grids and Applications

Springer

Lecture Notes
in Computational Science
and Engineering 88

Editors:

Timothy J. Barth
Michael Griebel
David E. Keyes
Risto M. Nieminen
Dirk Roose
Tamar Schlick

For further volumes:
http://www.springer.com/series/3527

Jochen Garcke • Michael Griebel
Editors

Sparse Grids and Applications

Springer

Editors
Jochen Garcke
Michael Griebel
Institut für Numerische Simulation
Universität Bonn
Bonn
Germany

ISSN 1439-7358
ISBN 978-3-642-31702-6 ISBN 978-3-642-31703-3 (eBook)
DOI 10.1007/978-3-642-31703-3
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012950005

Mathematics Subject Classification (2010): 65D99, 65M12, 65N99, 65Y20, 65N12, 62H99

Cover illustration: By courtesy of Dirk Pflüger

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

In the recent decade, there has been growing interest in the numerical treatment of high-dimensional problems. It is well known that classical numerical discretization schemes fail in more than three or four dimensions due to the curse of dimensionality. The technique of sparse grids allows to overcome this problem to some extent under suitable regularity assumptions. This discretization approach is obtained from a multi-scale basis by a tensor product construction and subsequent truncation of the resulting multiresolution series expansion.

Hans-Joachim Bungartz, Jochen Garcke, Michael Griebel, and Markus Hegland organized a workshop specifically to strengthen the research on the mathematical understanding and analysis of sparse grid discretization. Particular focus was given to aspects arising from applications. More than 40 researchers from four different continents attended the workshop in Bonn, Germany, from May 16–20, 2011.

This volume of LNCSE now comprises selected contributions from attendees of the workshop. The contents range from numerical analysis and stochastic partial differential equations to applications in data analysis, finance, and physics.

The workshop was hosted by the Institut für Numerische Simulation and the Hausdorff Research Institute for Mathematics (HIM) of the Rheinische Friedrich-Wilhelms-Universität Bonn as part of the Trimester Program *Analysis and Numerics for High Dimensional Problems*. Financial support of the HIM is kindly acknowledged. We especially thank Christian Rieger for his efforts and enthusiasm in the local organization of the workshop and the staff of the HIM for their assistance.

Bonn, Germany

Jochen Garcke
Michael Griebel

Contents

Contributors

Bastian Bohn Institute for Numerical Simulation, University of Bonn, Bonn, Germany

Hans-Joachim Bungartz Technische Universität München, Garching, Germany

Mike Espig Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany

Jochen Garcke Institut für Numerische Simulation, Universität Bonn, Bonn, Germany

Fraunhofer SCAI, Schloss Birlinghoven, Sankt Augustin, Germany

Thomas Gerstner Institut für Mathematik, Johann Wolfgang Goethe-Universität, Frankfurt am Main, Germany

Michael Griebel Institute for Numerical Simulation, University of Bonn, Bonn, Germany

Wolfgang Hackbusch Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany

Helmut Harbrecht Mathematisches Institut, Basel, Switzerland

Markus Hegland Australian National University, Canberra, Australia

Stefan Heinz Institut für Mathematik, Johann Wolfgang Goethe-Universität, Frankfurt am Main, Germany

Alexander Hullmann Institute for Numerical Simulation, University of Bonn, Bonn, Germany

John D. Jakeman Department of Mathematics, Purdue University, West Lafayette, IN, USA

Frank Jenko Max-Planck-Institut für Plasmaphysik, Garching, Germany

Christoph Kowitz Institute for Advanced Study, Technische Universität München, Munich, Germany

Alexander Litvinenko Technische Universität Braunschweig, Braunschweig, Germany

Hermann G. Matthies Technische Universität Braunschweig, Braunschweig, Germany

Benjamin Peherstorfer Technische Universität München, Garching, Germany

Michael Peters Mathematisches Institut, Basel, Switzerland

Dirk Pflüger Institute for Parallel and Distributed Systems, University of Stuttgart, 70569 Stuttgart, Germany

Christoph Reisinger Mathematical Institute, University of Oxford, Oxford, UK

Stephen G. Roberts Australian National University, Canberra, Australia

Markus Siebenmorgen Mathematisches Institut, Basel, Switzerland

Elmar Zander Technische Universität Braunschweig, Braunschweig, Germany

Stefan Zimmer Universität Stuttgart, Stuttgart, Germany

An Adaptive Sparse Grid Approach for Time Series Prediction

Bastian Bohn and Michael Griebel

Abstract A real valued, deterministic and stationary time series can be embedded in a—sometimes high-dimensional—real vector space. This leads to a one-to-one relationship between the embedded, time dependent vectors in $\mathbb{R}^d$ and the states of the underlying, unknown dynamical system that determines the time series. The embedded data points are located on an m-dimensional manifold (or even fractal) called attractor of the time series. Takens' theorem then states that an upper bound for the embedding dimension d can be given by $d \leq 2m + 1$.

The task of predicting future values thus becomes, together with an estimate on the manifold dimension m, a scattered data regression problem in d dimensions. In contrast to most of the common regression algorithms like support vector machines (SVMs) or neural networks, which follow a data-based approach, we employ in this paper a sparse grid-based discretization technique. This allows us to efficiently handle huge amounts of training data in moderate dimensions. Extensions of the basic method lead to space- and dimension-adaptive sparse grid algorithms. They become useful if the attractor is only located in a small part of the embedding space or if its dimension was chosen too large.

We discuss the basic features of our sparse grid prediction method and give the results of numerical experiments for time series with both, synthetic data and real life data.

B. Bohn (✉) · M. Griebel
Institute for Numerical Simulation, University of Bonn, 53115, Bonn, Germany
e-mail: bohn@ins.uni-bonn.de; griebel@ins.uni-bonn.de

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_1,

1 Introduction and Problem Formulation

One of the most important tasks in the field of data analysis is the prediction of future values from a given time series of data. In our setting, a time series $(s_j)_{j=1}^{\infty}$ is an ordered set of real values. The task of forecasting can now be formulated as:

Given the values $s_1, \ldots, s_N$, predict s_{N+1}!

To tackle the forecasting problem, we assume that the values s_j stem from an underlying stationary process which evolves in time. The aim is then to reconstruct the domain of this process as good as possible from the data $s_1, \ldots, s_N$ and to use this reconstruction for the prediction of the value s_{N+1}. To this end, let M_0 represent the phase space of the underlying system, let $\phi : M_0 \to M_0$ denote the corresponding equations of motion and let $o : M_0 \to \mathbb{R}$ be an observable which defines a time series by

$$(s_j)_{j=1}^{\infty} = \left(o\left(\phi^j(\mathbf{x}_0)\right)\right)_{j=1}^{\infty}, \tag{1}$$

where $\mathbf{x}_0 \in M_0$ is an arbitrary initial condition of the process and

$$\phi^j = \underbrace{\phi \circ \phi \circ \ldots \circ \phi}_{j \text{ times}}.$$

In practice M_0, ϕ and o are of course not known, but only the values s_j of the time series are given. To tackle the forecasting problem, we need to find a connection between the past values of the time series and the next one.

Takens' theorem [2, 25] provides the theoretical background to construct algorithms for this purpose. Assuming that a given equidistant time series consists of measurements, i.e. evaluations of the observable o, of an m-dimensional process which follows some deterministic equation of motion ϕ, there is the possibility to find a regular m-dimensional submanifold U of $\mathbb{R}^{2m+1}$ which is diffeomorphic to the phase space M_0 of the underlying system. The most common construction of such a submanifold works via delay embedding. In this case the time-dependent observations s_j are themselves used as coordinates to represent $U \subset \mathbb{R}^{2m+1}$. Here, since an element $\mathbf{x} \in U$ corresponds to one specific point in phase space, the dynamics of the process and thus the evolution of the time series itself are completely determined by $\mathbf{x}$ and the forecasting problem translates into an ordinary regression-like task in $\mathbb{R}^{2m+1}$. For an overview of the delay embedding scheme and different approaches to time series analysis see [21].

In this paper we use a regularized least squares approach to find an adequate approximation to the solution of the prediction problem. In combination with a FEM-like grid discretization this leads to a non-data-based approach whose computational costs grow only linearly in the number of elements of the given time series. Then, in contrast to most data-based techniques like support vector machines or standard neural networks using radial basis functions, this approach

is able to handle huge time series. But, if $d = 2m + 1$ denotes the dimension of the ambient space and 2^t is the number of grid points in one direction, the number of points in a conventionally discretized ambient space would grow like $O\left(2^{td}\right)$. Thus, this naive approach suffers from the curse of dimensionality which restricts the application of a conventional discretization to low-dimensional, i.e. to one-, two- or three-dimensional spaces.

To circumvent this problem the sparse grid discretization [1] is used in this paper. A first approach for the prediction of financial time series with sparse grids has been presented in [8]. For *regular sparse grids*, the number of grid points increases only like $O(2^t \cdot t^{d-1})$, i.e. the curse of dimensionality is now just present with respect to the term t. This way, we are able to efficiently deal with huge amounts of data in moderate dimensions up to about $d = 10$. Moreover, for most time series the high-dimensional data does not fill the whole space. The process obtained by using the delay embedding method then visits only a small fraction of the whole discretized area. This observation justifies a *space-adaptive sparse grid* [13] discretization which resolves the trajectory of the process. Finally, an ANOVA-like approach leads to *dimension-adaptive sparse grids* [10, 17] that are useful if our a priori choice of d is too large.

Thus, we will introduce two different adaptive algorithms in this paper: The space-adaptive algorithm locally adapts to features of the prediction function whereas the dimension-adaptive algorithm refines by employing subspaces which are relevant for an efficient representation of the prediction function in its ANOVA-decomposition.

In summary, each of our algorithms processes the following steps:

1. Estimation of the dimension m of the underlying process
2. Rewriting the forecasting problem as a regression problem in $\mathbb{R}^d$ with $d = 2m+1$
3. Approximating the solution of the regression problem in a discretized (regular, space-adaptive or dimension-adaptive) sparse grid space
4. Predicting the value s_{N+1} by point evaluation of the computed sparse grid function at $(s_{N-2m}, \ldots, s_N)^T$

Altogether, we obtain a new class of algorithms for the prediction of time series data which scale only linearly with the length of the given time series, i.e. the amount of data points, but still allow us to use reasonably large window sizes for the delay embedding due to our sparse grid approach. The new methods give excellent prediction results with manageable computational costs.

The remainder of this paper is organized as follows: In Sect. 1, we describe the delay embedding scheme and review some crucial issues concerning the application of Takens' theorem. In Sect. 2, we show how the forecasting problem can be rewritten as a regression problem. We also derive the regularized least squares functional which determines our predictor function. In Sect. 3, we deal with the regular sparse grid approximation. We deduce the associated linear system and solve it using a preconditioned CG-algorithm. Then, we introduce and discuss space- and

dimension-adaptive sparse grid algorithms. In Sect. 4, we give the results of numerical experiments which illustrate the favorable properties of our new methods.

2 Takens' Theorem and the Delay Embedding Scheme

We now provide the essential theory concerning Takens' theorem [25] and give a hint to some modifications from [2].

For an arbitrary $d \in \mathbb{N}$ we can create vectors

$$\mathbf{t}_j := \left(s_{j-d+1}, s_{j-d+2}, \ldots, s_{j-1}, s_j\right)^T \in \mathbb{R}^d, \ j \geq d$$

following the so-called delay embedding scheme. Each vector consists of d consecutive past time series values. A connection between these delay vectors and the unknown evolution of the process in the phase space is established by the following theorem:

Theorem 1. *Let M_0 be a compact m-dimensional C^2-manifold, let $\phi : M_0 \to M_0$ denote a C^2-diffeomorphism and let $o \in C^2(M_0, \mathbb{R})$. Then, $\boldsymbol{\rho}_{(\phi,o)} : M_0 \hookrightarrow \mathbb{R}^{2m+1}$ defined by*

$$\boldsymbol{\rho}_{(\phi,o)}(\mathbf{x}) := \left(o(\mathbf{x}), o(\phi(\mathbf{x})), o\left(\phi^2(\mathbf{x})\right), \ldots, o\left(\phi^{2m}(\mathbf{x})\right)\right) \tag{2}$$

is generically[1] *an embedding.*

This is Takens' theorem for discrete time series, see [19, 25]. The embedding $\boldsymbol{\rho}_{(\phi,o)}$ establishes a one-to-one connection between a state in the phase space M_0 and a $(2m+1)$-dimensional delay vector constructed by (2). It can formally be inverted and we obtain

$$\begin{aligned}
\phi^{j-2m}(\mathbf{x}_0) &= \boldsymbol{\rho}^{-1}_{(\phi,o)}\left(\left(o\left(\phi^{j-2m}(\mathbf{x}_0)\right), o\left(\phi^{j-2m+1}(\mathbf{x}_0)\right), \ldots, o\left(\phi^{j}(\mathbf{x}_0)\right)\right)\right) \\
&= \boldsymbol{\rho}^{-1}_{(\phi,o)}\left(\left(s_{j-2m}, s_{j-2m+1}, \ldots, s_j\right)\right) \\
&= \boldsymbol{\rho}^{-1}_{(\phi,o)}\left(\mathbf{t}_j\right)
\end{aligned}$$

[1] Here, "generically" means the following:
If $X_l := \left\{\mathbf{x} \in M_0 \mid \phi^l(\mathbf{x}) = \mathbf{x}\right\}$ fulfills $|X_l| < \infty$ for all $l \leq 2m$ and if the Jacobian matrix $\left(\mathrm{D}\phi^l\right)_{\mathbf{x}}$ of ϕ^l at $\mathbf{x}$ has pairwise distinct eigenvalues for all $l \leq 2m, \mathbf{x} \in X_l$, then the set of all $o \in C^2(M_0, \mathbb{R})$ for which the embedding property of Theorem 1 does not hold is a null set. As $C^2(M_0, \mathbb{R})$ is an infinite dimensional vector space, the term "null set" may not be straightforward. It should be understood in the way that every set $Y \supset \left\{o \in C^2(M_0, \mathbb{R}) \mid \boldsymbol{\rho}_{(\phi,o)} \text{ is an embedding}\right\}$ is prevalent.

for all $j \geq d$ with $\mathbf{t}_j \in \mathbb{R}^{2m+1}$ and thus $d = 2m + 1$. Applying $o \circ \phi^{2m+1}$ on both sides we obtain

$$o\left(\phi^{j+1}(\mathbf{x}_0)\right) = o\left(\phi^{2m+1}\left(\boldsymbol{\rho}^{-1}_{(\phi,o)}\left(\mathbf{t}_j\right)\right)\right). \tag{3}$$

This means that the value $s_{j+1} = o\left(\phi^{j+1}(\mathbf{x}_0)\right)$ is completely determined by the previous $2m+1$ values $s_{j-2m}, \ldots, s_j$.[2] Note that not necessarily all of the preceding $2m + 1$ values are essential to specify the current one, but Theorem 1 states that $2m + 1$ values are always sufficient to do so. Note furthermore that not only the next but all following values are determined by $2m + 1$ consecutive time series values. To see this one can just recursively follow the scheme in (3). Thus, if we have for example an equidistant time series with a 1 min gap between successive values but are interested in a 15 min forecast, we can still use $2m + 1$ consecutive values as input in our regression algorithm later on.[3]

Often, the equations of motion are described by a system of time-continuous differential equations instead of a time-discrete mapping ϕ as in Theorem 1. To this end, let V denote a vector field in $C^2(M_0, TM_0)$, let $o \in C^2(M_0, \mathbb{R})$ and let $\mathbf{z} : \mathbb{R}^+ \to M_0$ fulfill the differential equation

$$\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t} = \mathbf{V}(\mathbf{z}),\ \mathbf{z}(0) = \mathbf{z}_0 \tag{4}$$

for given $\mathbf{z}_0 \in M_0$. We define $\phi_t(\mathbf{z}_0) := \mathbf{z}(t)$ as the flow of the vector field $\mathbf{V}$. Now ϕ_τ can be used in Theorem 1 instead of ϕ for an arbitrary $\tau \in \mathbb{R}^+$ and the time-continuous setting is covered as well. For a thorough treatment of this case we refer to [2, 25].

A main requirement for Takens' theorem is the compactness of the manifold M_0, i.e. the domain of the process which contains all possible states. Sometimes the dynamics tends to form a so-called "strange attractor", which means that the trajectories of the system do not form a manifold anymore but just a point set A of non-integer dimension. In [2] it was shown that it is possible to generalize Theorem 1 also to this case:

Theorem 2. *Let $A \subset M_0 \subset \mathbb{R}^k$ where M_0 is an open subset of $\mathbb{R}^k$ and A is a compact subset of M_0 which possesses box-counting dimension $\widehat{\dim}(A) = m$. Furthermore, let $\phi : M_0 \to M_0$ be a C^2-diffeomorphism and let $o \in C^2(M_0, \mathbb{R})$. Then, for $\boldsymbol{\rho}_{(\phi,o)} : M_0 \hookrightarrow \mathbb{R}^{\lfloor 2m+1 \rfloor}$ defined as in (2), the properties*

[2] All functions on the right hand side of (3) are at least twice differentiable. As M_0 is compact, the concatenation of these functions lies in the standard Sobolev space $H_2(\boldsymbol{\rho}_{(\phi,o)}(M_0))$, where $\boldsymbol{\rho}_{(\phi,o)}(M_0) \subset \mathbb{R}^{2m+1}$ denotes the image of M_0 under $\boldsymbol{\rho}_{(\phi,o)}$.

[3] An alternative would be to simulate a time series with 15 min gaps by omitting intermediate values which would lead to a considerable reduction of the number of points. This is however not advantageous, as more points usually lead to better prediction results for the numerical algorithm.

1. $\boldsymbol{\rho}_{(\phi,o)}$ *is one-to-one on* A *and*
2. $\boldsymbol{\rho}_{(\phi,o)}$ *is an immersion on each compact subset* C *of a smooth manifold contained in* A

generically[4] *hold.*

Here, $\lfloor a \rfloor$ denotes the largest integer which is smaller or equal to $a \in \mathbb{R}^+$.

In real world applications, the set A is not a priori known. But for the delay embedding scheme to work we only need to know the box-counting dimension $\widehat{\dim}(A)$ of the set A. Its estimation is an elaborate task by its own. To this end, various approaches exist in the literature [22, 23, 26]. Here, we recommend using the Grassberger-Procaccia algorithm [12] to estimate the correlation dimension $\tilde{m}$ as an approximation of the box-counting dimension m since this worked best in our experiments. The delay length is then set to $d = \lfloor 2\tilde{m} + 1 \rfloor$.

In summary we have a theory which provides us with a justification to use delayed vectors like in (2) as input for a learning tool.

3 The Regression Problem and the Regularized Least Squares Approach

In this section, we describe how the task of predicting a time series can be recast into a higher-dimensional regression problem by means of delay embedding. Furthermore, we motivate a specific regularized least squares approach.

We assume that we have an infinite time series $\left(s_j\right)_{j=1}^{\infty}$ which is just an observation of a deterministic process on an m-dimensional attractor, compare Sect. 2. From now on, let $d := \lfloor 2m + 1 \rfloor$ denote the embedding dimension used for the delay scheme. We define

$$\mathbf{t}_j := \left(s_{j-d+1}, s_{j-d+2}, \ldots, s_{j-1}, s_j\right)^T \in \mathbb{R}^d, \; j \geq d, \tag{5}$$

to be the j-th delay vector. Due to Takens' theorem, there exists a $\hat{g} : \mathbb{R}^d \to \mathbb{R}$, with $\hat{g} := o \circ \phi^d \circ \boldsymbol{\rho}_{(\phi,o)}^{-1}$, cf. (3), such that

$$\hat{g}\left(\mathbf{t}_j\right) = s_{j+1} \text{ for all } j \geq d. \tag{6}$$

If we assume that $\left(s_j\right)_{j=1}^{N}$ is known a priori then our goal is to find a good approximation g to $\hat{g}$ with the help of $N - d + 1$ training patterns

[4] Here "generically" means the following:
If $\tilde{X}_l := \left\{\mathbf{x} \in A \mid \phi^l(\mathbf{x}) = \mathbf{x}\right\}$ fulfills $\widehat{\dim}\left(\tilde{X}_l\right) \leq \frac{l}{2}$ for all $l \leq \lfloor 2m + 1 \rfloor$ and if $\left(\mathrm{D}\phi^l\right)_{\mathbf{x}}$ has pairwise distinct eigenvalues for all $l \leq \lfloor 2m + 1 \rfloor, \mathbf{x} \in \tilde{X}_l$, then the set of all $o \in C^2\left(M_0, \mathbb{R}\right)$ for which the properties in Theorem 2 do not hold is a null set.

$$\left(\mathbf{t}_j, s_{j+1}\right) \in \mathbb{R}^d \times \mathbb{R},\ j = d, \ldots, N-1. \tag{7}$$

Thus, we now have to deal with a regression problem instead of the forecasting problem. Our approach is to choose $g \in X \subset \left\{f : \mathbb{R}^d \to \mathbb{R}\right\}$ as

$$g = \arg\min_{f \in X} \mathcal{F}(f)$$

where $\mathcal{F} : X \to \mathbb{R}^+ \cup \{\infty\}$ is a functional that expresses how good functions from X approximate $\hat{g}$. The function space X still has to be specified.

To this end, as we do not know any embedded points on which we want to evaluate g afterwards, it is common to minimize the expectation of some Lebesgue measurable cost function $c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+ \cup \{\infty\}$ with respect to the density $p : \mathbb{R}^d \times \mathbb{R} \to [0, 1]$ of all possible input patterns. This leads to

$$\mathcal{F}(f) = \mathbb{E}_p\left[c\left(y, f(\mathbf{x})\right)\right]$$

and thus gives

$$g = \arg\min_{f \in X} \mathbb{E}_p\left[c\left(y, f(\mathbf{x})\right)\right] = \arg\min_{f \in X} \int_{\mathbb{R}^d \times \mathbb{R}} c\left(y, f(\mathbf{x})\right) p(\mathbf{x}, y)\left(\mathrm{d}\mathbf{x} \otimes \mathrm{d}y\right).$$

Note here that we have to restrict X to contain only Lebesgue measurable functions to make this term well-defined.

Since we have to cope with training patterns and do not know the exact density p, we use the empirical density

$$\hat{p}(\mathbf{x}, y) = \frac{1}{N-d} \sum_{j=d}^{N-1} \delta_{\mathbf{t}_j}(\mathbf{x}) \delta_{s_{j+1}}(y)$$

instead of p. This results in the problem of finding the argument of the minimum of

$$\mathcal{F}(f) = \int_{\mathbb{R}^d \times \mathbb{R}} c\left(y, f(\mathbf{x})\right) \hat{p}(\mathbf{x}, y)\left(\mathrm{d}\mathbf{x} \otimes \mathrm{d}y\right) = \frac{1}{N-d} \sum_{j=d}^{N-1} c\left(s_{j+1}, f(\mathbf{t}_j)\right). \tag{8}$$

Note that if we want to calculate the point evaluations $f\left(\mathbf{t}_j\right)$, then the set of admissible functions X has here to be restricted further to contain only functions for which point evaluations are well defined.

We decided to use $c(a, b) := (a - b)^2$. One can easily show that for this specific cost function a minimizer of $\mathcal{F}$ maximizes the likelihood of the given input data under the assumption of a Gaussian noise term being added to each exact time series value, see e.g. Sect. 3.3 in [24].[5]

[5] Other cost functions can be used as well but these might lead to non-quadratic or even non-convex minimization problems.

The minimization of (8) for $f \in X$ still leads to an ill-posed problem and a further restriction of the space of admissible functions is therefore needed. To this end, Tikhonov proposed to add a constraint of the form $\Psi(f) \leq c$ with an arbitrary positive constant c and a nonnegative functional $\Psi : X \to \mathbb{R}^+$ which is strictly convex on a certain subspace depending on the problem itself, see [27]. Using the method of Lagrange multipliers we then obtain the new minimization problem

$$g = \arg\min_{f \in X} \mathcal{F}(f) := \arg\min_{f \in X} \left(\frac{1}{N-d} \sum_{j=d}^{N-1} c\left(s_{j+1}, f(\mathbf{t}_j)\right) + \lambda \Psi(f) \right) \tag{9}$$

which is well-posed if λ is positive. We will employ the Sobolev semi-norm

$$\Psi(f) := |f|_{H^1_{\text{mix}}} = \sum_{|\mathbf{a}|_\infty = 1} \left\| \frac{\mathrm{d}^{a_1}}{\mathrm{d}x_1^{a_1}} \cdots \frac{\mathrm{d}^{a_d}}{\mathrm{d}x_d^{a_d}} f \right\|^2_{L_2(\mathbb{R}^d)} \tag{10}$$

since this perfectly fits after discretization to our basis functions as we will see later. Here $\mathbf{a} = (a_1, \ldots, a_d)$ denotes a multi index and $|\mathbf{a}|_\infty := \max_{i=1,\ldots,d} |a_i|$. We will use the function $g \in X$ defined in (9) as continuous approximation to $\hat{g}$ from now on.

Instead of our H^1_{mix}-semi-norm, a method using gradient penalties—which corresponds to the H^1 semi-norm—was presented in [9] and error bounds were provided for a discrete solution achieved by the so-called combination technique. Note that some of these results rely on the assumption of independent and uniformly distributed samples. Nevertheless, similar results can be given for our case under the assumption of independently drawn samples according to the probability distribution on the reconstructed attractor. The resulting errors then refer to the attractor measure instead of the Lebesgue measure.

3.1 Minimization for an Arbitrary Basis

Now let $\{\gamma_i\}_{i=1}^\infty$ be a basis of $\Gamma := \{f \in X \mid \Psi(f) \leq c\}$. Our task is to find a

$$\mathbf{w} := (w_1, w_2, \ldots)$$

with $w_i \in \mathbb{R}$ for each $i \in \mathbb{N} \setminus \{0\}$, which minimizes

$$\frac{1}{N-d} \sum_{j=d}^{N-1} \left(s_{j+1} - \sum_{i=1}^{\infty} w_i \gamma_i\left(\mathbf{t}_j\right) \right)^2 + \lambda \sum_{i=1}^{\infty} \sum_{k=1}^{\infty} \left(w_i w_k \sum_{|\mathbf{a}|_\infty = 1} \langle \mathrm{D}^{\mathbf{a}} \gamma_i, \mathrm{D}^{\mathbf{a}} \gamma_k \rangle_{L_2(\mathbb{R}^d)} \right) \tag{11}$$

where $\mathrm{D}^{\mathbf{a}} = \frac{\mathrm{d}^{a_1}}{\mathrm{d}x_1^{a_1}} \ldots \frac{\mathrm{d}^{a_d}}{\mathrm{d}x_d^{a_d}}$ denotes a multivariate derivative. The corresponding function

$$g(\mathbf{x}) = \sum_{i=1}^{\infty} w_i \gamma_i(\mathbf{x})$$

then would give us the approximate prediction $s_{N+1} \approx g(\mathbf{t}_N)$ by point evaluation at $\mathbf{t}_N$. As (11) is a sum of strictly convex functions, the argument g of the minimum can be found by identifying the zeroes of $\frac{\mathrm{d}}{\mathrm{d}w_l} \mathcal{F}(f)$ for all $l \in \mathbb{N} \setminus \{0\}$. For $\eta := (N-d)\lambda$ this leads to the *infinite* system

$$\sum_{j=d}^{N-1} s_{j+1} \gamma_l(\mathbf{t}_j) = \sum_{i=1}^{\infty} w_i \left(\sum_{j=d}^{N-1} \gamma_l(\mathbf{t}_j) \gamma_i(\mathbf{t}_j) + \eta h(\gamma_i, \gamma_l) \right) \tag{12}$$

for all $l \in \mathbb{N} \setminus \{0\}$, where $h : \Gamma \times \Gamma \to \mathbb{R}$ denotes the semi-definite bilinear form

$$h(s,t) = \sum_{|\mathbf{a}|_\infty = 1} \langle \mathrm{D}^{\mathbf{a}} s, \mathrm{D}^{\mathbf{a}} t \rangle_{L_2(\mathbb{R}^d)}.$$

3.2 Minimization for a Kernel Basis in a Reproducing Kernel Hilbert Space

To derive a finite solution procedure, the following approach is standard in the mathematical learning community. For the case $\Psi(f) = ||f||_{\mathcal{H}}^2$, with $\mathcal{H}$ being a reproducing kernel Hilbert space, we can write g from (9) as a finite linear combination of evaluations of the reproducing kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ in the points corresponding to the training patterns

$$g(\mathbf{x}) = \sum_{j=d}^{N-1} g_j k(\mathbf{t}_j, \mathbf{x})$$

with some real-valued weights g_j. This is known as the representer theorem for reproducing kernel Hilbert spaces, see e.g. [24]. Analogous observations as above result with the property $\langle k(\mathbf{t}_i, \cdot), k(\mathbf{t}_j, \cdot) \rangle_{\mathcal{H}} = k(\mathbf{t}_i, \mathbf{t}_j)$ in the finite system

$$\sum_{j=d}^{N-1} s_{j+1} k(\mathbf{t}_j, \mathbf{t}_l) = \sum_{j=d}^{N-1} g_j \left(\sum_{i=d}^{N-1} k(\mathbf{t}_i, \mathbf{t}_j) k(\mathbf{t}_i, \mathbf{t}_l) + \eta k(\mathbf{t}_j, \mathbf{t}_l) \right)$$

for all $l \in \{d, \ldots, N-1\}$. If the $\left(k\left(\mathbf{t}_j, \mathbf{x}\right)\right)_{j=d}^{N-1}$ are linearly independent[6] this leads to the linear system

$$\mathbf{s} = (\mathbf{K} + \eta \mathbf{I})\, \mathbf{g} \tag{13}$$

where $\mathbf{K} \in \mathbb{R}^{(N-d)\times(N-d)}$ is the kernel matrix with entries $\mathbf{K}_{i,j} = k\left(\mathbf{t}_i, \mathbf{t}_j\right)$, $\mathbf{I} \in \mathbb{R}^{(N-d)\times(N-d)}$ is the identity matrix and $\mathbf{g} = (g_d, \ldots, g_{N-1})^T \in \mathbb{R}^{N-d}$, $\mathbf{s} = (s_{d+1}, \ldots, s_N)^T \in \mathbb{R}^{N-d}$.

Note that for the case (10) we only regularized with a semi-norm of a reproducing kernel Hilbert space but still get the representation

$$g(\mathbf{x}) = \sum_{j=d}^{N-1} g_j k\left(\mathbf{t}_j, \mathbf{x}\right) + g_0(\mathbf{x})$$

with a $g_0 : \mathbb{R}^d \to \mathbb{R}$ from the null space of Ψ and a certain kernel function k, see [24].

One could now try to solve the linear system (13). The major problem of this approach—besides the knowledge of an explicit formulation of the reproducing kernel[7]—is the complexity with respect to the number of input patterns. The direct solution of (13) would involve a number of operations of the order $O\left(N^3\right)$ since we have to deal with a full system matrix here. But even if one does not compute the inverse of $\mathbf{K} + \eta\mathbf{I}$ directly and uses an appropriate iterative scheme instead, the complexity for solving this system is at least $O\left(N^2\right)$ because of the dense kernel matrix $\mathbf{K}$. Therefore, in the next section, we will consider the infinite system (12) in the first place and resort to a further approximation of our prediction problem by discretization.

4 Discretization via Sparse Grids

To find an approximate solution to (12) we restrict ourselves to a finite dimensional subspace $\Gamma_M := \operatorname{span}\{\gamma_i\}_{i=1}^M \subset \Gamma := \{f \in X \mid \Psi(f) \leq c\}$ for some $M \in \mathbb{N}$. For the naive full grid approach the curse of dimensionality then shows up in the number of necessary grid points which grows exponentially with d. To deal with this issue, we will employ the sparse grid discretization technique and its adaptive enhancements here. To this end, we will assume that the domain of $\hat{g}$ (and thus g) is the d-dimensional hypercube

$$\mathbf{H}_d := [0,1]^d.$$

[6] If this is not the case we can choose a linearly independent subsystem and continue analogously.

[7] See [28] for several reproducing kernels and their corresponding Hilbert spaces.

Note that this is not a restriction since the domain of the underlying original process is compact (cf. Theorems 1 and 2). By rescaling the resulting domain of the reconstructed process we always can obtain the domain $[0, 1]^d$.

4.1 Multilevel Hierarchical Bases and Regular Sparse Grids

First, we recall the construction of a full grid space using a piecewise linear hierarchical basis and discuss its relation to a sparse grid space. Let the one-dimensional hat function $\phi : \mathbb{R} \to [0, 1]$ be defined by

$$\phi(x) := \begin{cases} 1 - |x|, & \text{if } x \in [-1, 1] \\ 0 & \text{else} \end{cases}$$

and let

$$\phi_{l,i}(x) := \phi(2^l \cdot x - i)|_{[0,1]}$$

for any $l, i \in \mathbb{N}$ be a dilated and rescaled version of ϕ restricted to the interval $[0, 1]$. One can easily see that $\operatorname{supp}(\phi_{l,i}) = \left((i-1)2^{-l}, (i+1)2^{-l}\right) \cap [0, 1]$. The construction of a d-dimensional hat function is straightforward via the tensor product

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) := \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j),$$

where $\mathbf{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$ is the multivariate level and $\mathbf{i} = (i_1, \ldots, i_d) \in \mathbb{N}^d$ denotes the multivariate position index. Furthermore, we define $\mathbf{x}_{\mathbf{l},\mathbf{i}} := \mathbf{i} \cdot 2^{-\mathbf{l}}$, where the multiplication has to be understood componentwise, i.e. $\mathbf{x}_{\mathbf{l},\mathbf{i}} = \left(x_{l_1,i_1}, \ldots, x_{l_d,i_d}\right)^T$ with $x_{l_j,i_j} := i_j \cdot 2^{-l_j}$. For a fixed $\mathbf{l} \in \mathbb{N}^d$, we then have with

$$\Omega_{\mathbf{l}} := \left\{\mathbf{x}_{\mathbf{l},\mathbf{i}} \mid \mathbf{0} \leq \mathbf{i} \leq 2^{\mathbf{l}}\right\}$$

the full grid of level $\mathbf{l}$. Here, the inequalities are to be understood componentwise and $\mathbf{0} = (0, \ldots, 0)$ is the null index. The space of piecewise d-linear functions on the grid $\Omega_{\mathbf{l}}$ is

$$V_{\mathbf{l}} := \operatorname{span}\{B_{\mathbf{l}}\} \quad \text{with } B_{\mathbf{l}} = \left\{\phi_{\mathbf{l},\mathbf{i}} \mid \mathbf{0} \leq \mathbf{i} \leq 2^{\mathbf{l}}\right\}.$$

$B_{\mathbf{l}}$ is called nodal basis since the value of a function $f_{\mathbf{l}}(\mathbf{x}) = \sum_{\mathbf{0} \leq \mathbf{i} \leq 2^{\mathbf{l}}} f_{\mathbf{l},\mathbf{i}} \cdot \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) \in V_{\mathbf{l}}$ on one of the grid points $\mathbf{x}_{\mathbf{l},\mathbf{j}}$ of $\Omega_{\mathbf{l}}$ is given by the coefficient $f_{\mathbf{l},\mathbf{j}} \in \mathbb{R}$ that corresponds to $\phi_{\mathbf{l},\mathbf{j}}$.

Now, let

$$\mathbf{I}_{\mathbf{l}} := \left\{ \mathbf{i} \in \mathbb{N}^d \;\middle|\; \begin{array}{ll} 0 \leq i_j \leq 1, & \text{if } l_j = 0 \\ 1 \leq i_j \leq 2^{l_j} - 1,\ i_j \text{ odd} & \text{if } l_j > 0 \end{array} \text{ for all } 1 \leq j \leq d \right\}. \quad (14)$$

Then, $W_{\mathbf{l}} := \operatorname{span}\left\{\phi_{\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in \mathbf{I}_{\mathbf{l}}\right\}$ is a hierarchical increment space (or detail space) because of the property

$$W_{\mathbf{l}} = \operatorname{span}\left\{ B_{\mathbf{l}} \setminus \bigcup_{j=1}^{d} B_{\mathbf{l}-\mathbf{e}_j} \right\}$$

where $\mathbf{e}_j$ denotes the j-th unit vector and $B_{\mathbf{k}} := \emptyset$ for each $\mathbf{k} = (k_1, \ldots, k_d)$ with $k_j < 0$ for some $j = 1, \ldots, d$. Thus we get

$$V_{\mathbf{l}} = \bigoplus_{\mathbf{k} \leq \mathbf{l}} W_{\mathbf{k}} = \operatorname{span}\left\{\tilde{B}_{\mathbf{l}}\right\}$$

with the hierarchical basis

$$\tilde{B}_{\mathbf{l}} := \left\{\phi_{\mathbf{k},\mathbf{i}} \mid \mathbf{i} \in \mathbf{I}_{\mathbf{k}}, \mathbf{k} \leq \mathbf{l}\right\}.$$

Now, we can define the space of piecewise d-linear functions on the regular (isotropic) full grid

$$\Omega_t := \Omega_{(t,\ldots,t)} = \left\{\mathbf{x}_{\mathbf{k},\mathbf{i}} \mid |\mathbf{k}|_\infty \leq t, \mathbf{i} \in \mathbf{I}_{\mathbf{k}}\right\}$$

of level $t \in \mathbb{N}$ by

$$V_t := V_{(t,\ldots,t)} = \bigoplus_{|\mathbf{k}|_\infty \leq t} W_{\mathbf{k}}.$$

If

$$f_t = \sum_{|\mathbf{k}|_\infty \leq t} \sum_{\mathbf{i} \in \mathbf{I}_{\mathbf{k}}} f_{\mathbf{k},\mathbf{i}} \phi_{\mathbf{k},\mathbf{i}}$$

is the interpolant of $f \in H^2(\mathbf{H}_d)$ in V_t it holds that

$$||f - f_t||_{L_2(\mathbf{H}_d)} = O\left(2^{-2t}\right). \tag{15}$$

Next, we define the regular sparse grid of level t by

$$\Omega_t^s := \left\{\mathbf{x}_{\mathbf{k},\mathbf{i}} \mid n_d(\mathbf{k}) \leq t, \mathbf{i} \in \mathbf{I}_{\mathbf{k}}\right\} \tag{16}$$

and the corresponding function space by

$$V_t^s := \bigoplus_{\substack{\mathbf{k} \in \mathbb{N}^d \\ n_d(\mathbf{k}) \leq t}} W_{\mathbf{k}},$$

where $n_d(\mathbf{0}) := 0$ and

$$n_d(\mathbf{k}) := |\mathbf{k}|_1 - d + |\{m \mid \mathbf{k}_m = 0\}| + 1$$

for every other $\mathbf{k} \in \mathbb{N}^d$. Here, $|\mathbf{k}|_1 := \sum_{j=1}^{d} |\mathbf{k}_j|$ denotes the ℓ^1 norm. This specific definition of n_d guarantees that the resolution of grids on the boundary is the same as the resolution of grids in the interior of the domain.

If

$$f_t^s(\mathbf{x}) = \sum_{\substack{\mathbf{k}\in\mathbb{N}^d \\ n_d(\mathbf{k})\le t}} \sum_{\mathbf{i}\in\mathbf{I}_\mathbf{k}} \alpha_{\mathbf{k},\mathbf{i}} \phi_{\mathbf{k},\mathbf{i}}(\mathbf{x}) \in V_t^s$$

is the interpolant of $f \in H^2_{\text{mix}}(\mathbf{H}_d)$ in V_t^s, it holds that

$$||f - f_t^s||_{L_2(\mathbf{H}_d)} = O\left(2^{-2t} t^{d-1}\right).$$

Thus, compared to (15), the accuracy is only slightly worse by a factor t^{d-1}. However, the number of points in the full grid is $|\Omega_t| = O\left(2^{td}\right)$ and suffers from the curse of dimensionality for large d whereas, in the sparse grid case, $M := |\Omega_t^s| = O\left(2^t \cdot t^{d-1}\right)$ holds and the exponential dependence of d now only affects the level t instead of 2^t. For a thorough treatment of sparse grids, approximation results and complexity issues we refer to [1] and the references therein.

By solving (9) in the discrete space $V_t^s \subset \Gamma$ we get (analogously to (12))

$$\sum_{j=d}^{N-1} s_{j+1} \phi_{\mathbf{l},\mathbf{i}}\left(\mathbf{t}_j\right) = \sum_{\substack{\mathbf{k}\in\mathbb{N}^d : n_d(\mathbf{k})\le t, \\ \mathbf{m}\in\mathbf{I}_\mathbf{k}}} \alpha_{\mathbf{k},\mathbf{m}} \left(\sum_{j=d}^{N-1} \phi_{\mathbf{l},\mathbf{i}}\left(\mathbf{t}_j\right) \phi_{\mathbf{k},\mathbf{m}}\left(\mathbf{t}_j\right) + \eta h\left(\phi_{\mathbf{l},\mathbf{i}}, \phi_{\mathbf{k},\mathbf{m}}\right) \right) \tag{17}$$

for all $\mathbf{l} \in \mathbb{N}^d : n_d(\mathbf{l}) \le t$ and $\mathbf{i} \in \mathbf{I}_\mathbf{l}$.

A preconditioned multilevel conjugate gradient (pCG) algorithm is used to solve the linear system (17) iteratively. Here, for reasons of simplicity, we employ as preconditioner the inverse of the diagonal of the system matrix of (17) after its transformation to a prewavelet representation, see [15]. As we only need to implement matrix-vector-multiplications for the pCG algorithm, the system matrices are not assembled explicitly. The hierarchical structure and the compact support of our basis functions allow a fast application[8] of the first term in the brackets on the right hand side of (17) in $O\left(N \cdot t^d\right)$ operations. Because of the product structure of H^1_{mix} an efficient implementation of the unidirectional principle can be employed for the on-the-fly multiplication of the term corresponding to the bilinear form h, see e.g. [4]. This needs $O(M)$ operations. Thus, the costs of a single iteration of the pCG algorithm are only $O\left(N \cdot t^d + M\right) = O\left((N \cdot t + 2^t) \cdot t^{d-1}\right)$ operations. For a detailed review of computational issues on the implementation of sparse grid methods, grid traversal strategies and linear system solvers, we refer to [4]. See Fig. 1 for two sparse grid examples.

[8] Note that the use of the combination technique [16] even allows here for a slight improvement to $O\left(N \cdot t^{d-1}\right)$. In both cases, however, the constant in the O-notation grows exponentially with d.

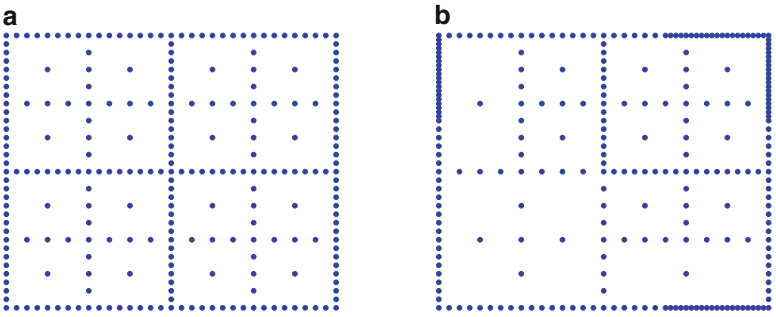


Fig. 1 Different sparse grid examples in two dimensions. (**a**) Regular sparse grid of level 5. (**b**) Space-adaptive sparse grid

4.2 Space-Adaptive Sparse Grids

Since most attractors only fill a sparse pattern of $\mathbf{H}_d$, it is obvious that a regular grid is not necessarily the best structure to approximate a function on such an attractor. On the one hand, there might not be enough grid points in relevant regions to fit the function which leads to bad approximations. On the other hand, there might be too many grid points in irrelevant areas which causes overfitting and results in an unnecessary high cost complexity, see [9] for a thorough treatment of this issue. One would prefer a grid which rather matches the shape of the trajectory than the ambient space $\mathbf{H}_d$. Such a grid (and of course the corresponding function space) can be derived using an iterative algorithm which adaptively creates finer grid resolutions where needed. The main component of such a procedure is an appropriate error indicator which decides if the grid has to be locally refined in a certain region. We here simply use

$$\epsilon_{\mathbf{l},\mathbf{i}} := \left|\left|\alpha_{\mathbf{l},\mathbf{i}}\phi_{\mathbf{l},\mathbf{i}}\right|\right|_{L_\infty(\mathbf{H}_d)} = \left|\alpha_{\mathbf{l},\mathbf{i}}\right|$$

as such an indicator. For more elaborate techniques and details on how to choose a reliable *and* efficient indicator $\epsilon_{\mathbf{l},\mathbf{i}}$ for the case of specific norms of the error, we refer to [13].

Our overall algorithm proceeds as follows: First, it starts with a regular sparse grid for some low level $\Omega^s_{\text{adp}} = \tilde{\Omega}^s_{\text{adp}} := \Omega^s_t$ and solves (17) on this grid. Then, it checks for each $\left\{(\mathbf{l},\mathbf{i}) \mid \mathbf{x}_{\mathbf{l},\mathbf{i}} \in \tilde{\Omega}^s_{\text{adp}}\right\}$ if $\epsilon_{\mathbf{l},\mathbf{i}} > \varepsilon$, where $\varepsilon \in \mathbb{R}^+$ is some fix threshold. If this is the case for the pair $(\mathbf{l},\mathbf{i})$ with odd i_j or $i_j = 0$ for each $j \in \{1,\ldots,d\}$, all of its child nodes are inserted into the grid Ω^s_{adp} if they are not already contained.[9] In the one-dimensional case the child nodes are defined as

[9]Note here that it is not enough to check the surplus of points which have been inserted in the last iteration. The hierarchical surplus of all other points can change as well when calculating the solution on the refined grid.

$$\text{child}\,(x_{l,i}) := \begin{cases} \{x_{l+1,2i\pm1}\} & \text{if } l > 0, \\ \{x_{1,1}\} & \text{if } l = 0, i = 1, \\ \{x_{0,1}\} & \text{if } l = 0, i = 0. \end{cases} \tag{18}$$

In the multivariate case we define $\text{child}\left(\mathbf{x_{l,i}}\right)$ as

$$\left\{ \mathbf{x_{k,m}} \in \Omega_{\mathbf{k}} \;\middle|\; \begin{array}{l} \text{There exists } j \in \{1,\ldots,d\}, \text{ s.t. } x_{k_j,m_j} \in \text{ child}\left(x_{l_j,i_j}\right) \\ \text{and } k_h = l_h, m_h = i_h \text{ for all } h \in \{1,\ldots,d\} \setminus \{j\} \end{array} \right\}. \tag{19}$$

After the insertion it has to be guaranteed—by e.g. inserting further nodes where needed—that all hierarchical ancestors of every inserted point are contained in the resulting grid. Otherwise, an incorrect hierarchical basis representation for the corresponding function space would result and common grid traversal algorithms would run into problems. To achieve this we simply insert each missing direct ancestor and proceed recursively with the inserted points until each direct ancestor to every grid point has been inserted into Ω^s_{adp}. The direct ancestors of points $\mathbf{x_{l,i}}$ with odd i_j or $i_j = 0$ for each $j = \{1,\ldots,d\}$ are defined by

$$\text{directAnc}\left(\mathbf{x_{l,i}}\right) := \left\{ \mathbf{x_{k,m}} \in \Omega_{\mathbf{l}} \mid \mathbf{x_{l,i}} \in \text{ child}\left(\mathbf{x_{k,m}}\right) \right\}. \tag{20}$$

Algorithm 1 The space-adaptive sparse grid algorithm

Input: starting level t, threshold ε, #iterations L, error indicators $\epsilon_{\mathbf{l,i}}$, time series $\left(s_j\right)_{j=1}^{N}$, embedding dimension d, regularization parameter λ
Output: space-adaptive sparse grid Ω^s_{adp}

initialize: $\Omega^s_{\text{adp}} \leftarrow \Omega^s_t$, $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_t$, It $\leftarrow 0$
while It $< L$ **do**
  solve (17) on $\tilde{\Omega}^s_{\text{adp}}$
  for all $(\mathbf{k},\mathbf{m})$ with odd m_j or $m_j = 0$ for each $j \in \{1,\ldots,d\}$ and $\mathbf{x_{k,m}} \in \tilde{\Omega}^s_{\text{adp}}$ **do**
    if $\epsilon_{\mathbf{k,m}} > \varepsilon$ **then**
      $\Omega^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}} \cup \text{ child}\left(\mathbf{x_{k,m}}\right)$
    end if
  end for
  if $\tilde{\Omega}^s_{\text{adp}} = \Omega^s_{\text{adp}}$ **then**
    return Ω^s_{adp}
  end if
  $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}}$
  for all $\mathbf{x_{k,m}}$ with odd m_j or $m_j = 0$ for each $j \in \{1,\ldots,d\}$ and $\mathbf{x_{k,m}} \in \tilde{\Omega}^s_{\text{adp}}$ **do**
    $\Omega^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}} \cup \text{AllAncestors}(\mathbf{k}, \mathbf{m}, d)$
  end for
  $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}}$
  It $\leftarrow$ It+1
end while
return Ω^s_{adp}

Algorithm 2 AllAncestors($\mathbf{l}$, $\mathbf{i}$, d)

Input: multivariate level $\mathbf{l}$, multivariate index $\mathbf{i}$, embedding dimension d
Output: set X of all ancestors of $\mathbf{x}_{\mathbf{l},\mathbf{i}}$

initialize: $X \leftarrow \emptyset$
$X \leftarrow X \cup \text{directAnc}(\mathbf{x}_{\mathbf{l},\mathbf{i}})$
for all $\mathbf{x}_{\mathbf{k},\mathbf{m}} \in \text{directAnc}(\mathbf{x}_{\mathbf{l},\mathbf{i}})$ with odd m_j or $m_j = 0$ for each $j \in \{1, \ldots, d\}$ **do**
    $X \leftarrow X \cup \text{AllAncestors}(\mathbf{k}, \mathbf{m}, d)$
end for
return X

When every relevant grid point of $\tilde{\Omega}^s_{\text{adp}}$ has been visited and treated accordingly, we set $\tilde{\Omega}^s_{\text{adp}} = \Omega^s_{\text{adp}}$ and start anew. This iteration runs until either no point needs to be refined or the number of iterations reaches some fixed limit $L \in \mathbb{N}$. A summary of the procedure can be found in Algorithm 1. For details on runtime and technical issues we refer to [4].

4.3 Dimension-Adaptive Sparse Grids

In the case of attractors which fill a highly anisotropic part of the ambient space $\mathbf{H}_d$ or in case the ambient space dimension was overestimated, it is desirable to employ dimension-adaptive refinement instead of pure space-adaptive refinement. There, refinement takes place globally but only in directions which are relevant for the construction of a good forecasting function. Dimension-adaptivity for sparse grids has been introduced in [17]. The application of dimension-adaptive algorithms has been studied for integration in [10] and for approximation in [6, 7]. The approach which we use in the following is a little bit different though, it can be found in [4].

To motivate the idea of dimension-adaptive grids we will shortly review the concept of the ANOVA (Analysis of Variance) decomposition. We introduce a splitting

$$V = \mathbf{1} \oplus \mathcal{C} \tag{21}$$

of a space V of univariate functions with domain $[0, 1]$ into the space of constant functions $\mathbf{1}$ and the remainder $\mathcal{C}$. This is done using the identity

$$f = P(f) + (f - P(f))$$

for some projector $P : V \to \mathbf{1}$ with $P|_{\mathbf{1}} = \text{id}$.

For multivariate tensor product function spaces V we apply the splitting in every direction, i.e.

$$
\begin{aligned}
V &= \bigotimes_{i=1}^{d} V_i = \bigotimes_{i=1}^{d} (\mathbf{1}_i \oplus \mathcal{C}_i) \\
&= \mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_d \\
&\oplus \bigoplus_{i=1}^{d} (\mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_{i-1} \otimes \mathcal{C}_i \otimes \mathbf{1}_{i+1} \otimes \ldots \otimes \mathbf{1}_d) \\
&\oplus \bigoplus_{i=1}^{d} \bigoplus_{j=i+1}^{d} \left(\mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_{i-1} \otimes \mathcal{C}_i \otimes \mathbf{1}_{i+1} \otimes \ldots \otimes \mathbf{1}_{j-1} \otimes \mathcal{C}_j \otimes \mathbf{1}_{j+1} \otimes \ldots \otimes \mathbf{1}_d\right) \\
&\vdots \\
&\oplus \mathcal{C}_1 \otimes \ldots \otimes \mathcal{C}_d,
\end{aligned} \tag{22}
$$

and receive a unique splitting of a function $f \in V$ into the sum of a constant function, d univariate functions, $\frac{d(d-1)}{2}$ bivariate functions, and so on, i.e.

$$
f(x_1, \ldots, x_d) = f_0 + \sum_{i=1}^{d} f_i(x_i) + \sum_{i=1}^{d} \sum_{j=i+1}^{d} f_{ij}(x_i, x_j) + \ldots + f_{1,\ldots,d}(x_1, \ldots, x_d). \tag{23}
$$

We call f_0 the ANOVA component of order 0, the f_i are ANOVA components of order 1, and so on.
The most common choice for P is

$$
P(f) := \int_{[0,1]} f(x) \mathrm{d}x
$$

for $V \subset L_2([0,1])$, which just gives the classical L_2-ANOVA decomposition. Another choice is

$$
P(f) := f(a)
$$

which leads to a well-defined decomposition if the point evaluation in a is well-defined for all functions in V. This results in the so-called anchored ANOVA decomposition with anchor a. It is well suited to our piecewise linear basis functions.

Here, to transfer the concept of the multivariate anchored ANOVA decomposition to the piecewise linear hierarchical basis discretization, we have to change the index set introduced in (14) as in [6]. We define

$$
\tilde{\mathbf{I}}_{\mathbf{l}} := \left\{ \mathbf{i} \in \mathbb{N}^d \;\middle|\; \begin{array}{ll} i_j = 0, & \text{if } l_j = -1 \\ i_j = 1, & \text{if } l_j = 0 \\ 1 \le i_j \le 2^{l_j} - 1,\ i_j \text{ odd} & \text{if } l_j > 0 \end{array} \quad \text{for all } 1 \le j \le d \right\} \tag{24}
$$

and allow the negative level -1. Furthermore, we define the one-dimensional basis function $\phi_{-1,0} := \chi_{[0,1]}$ to be the indicator function of the interval $[0, 1]$. With this and the definition

$$\tilde{W}_{\mathbf{l}} := \operatorname{span}\{\phi_{\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in \tilde{\mathbf{I}}_{\mathbf{l}}\}$$

we see[10] that

$$\begin{aligned}\tilde{V}_{\mathbf{l}} := \bigoplus_{-\mathbf{1}\le\mathbf{k}\le\mathbf{l}} \tilde{W}_{\mathbf{k}} &= \bigoplus_{-\mathbf{1}\le\mathbf{k}\le\mathbf{l}} \operatorname{span}\{\phi_{\mathbf{k},\mathbf{m}} \mid \mathbf{m} \in \tilde{\mathbf{I}}_{\mathbf{k}}\} \\ &= \bigoplus_{\mathbf{0}\le\mathbf{k}\le\mathbf{l}} \operatorname{span}\{\phi_{\mathbf{k},\mathbf{m}} \mid \mathbf{m} \in \mathbf{I}_{\mathbf{k}}\} = \bigoplus_{\mathbf{0}\le\mathbf{k}\le\mathbf{l}} W_{\mathbf{k}} = V_{\mathbf{l}}\end{aligned}$$

for all $\mathbf{l}$ with $l_j \ge 0$ for all $j = 1, \ldots, d$. This way, we just have split the space of linear functions on $[0, 1]$, which was previously spanned by the two linear basis functions associated to the two boundary points, further into the sum of one constant (level -1) and one linear function (level 0). If we define the norm of a multivariate level index with possibly negative coordinates as

$$|\mathbf{l}| := |(\max(l_1, 0), \ldots, \max(l_d, 0))|$$

we can maintain our previous definition for sparse grids (16) using

$$\tilde{n}_d(\mathbf{k}) := \begin{cases} 0 & \text{if } k_j \le 0 \text{ for all } 1 \le j \le d \\ |\mathbf{k}|_1 - d + |\{m \mid \mathbf{k}_m \le 0\}| + 1 & \text{else} \end{cases}$$

instead of $n_d(\mathbf{k})$. But we now are able to identify functions which are constant in direction j as they are elements of $\tilde{V}_{(l_1,\ldots,l_{j-1},-1,l_{j+1},\ldots,l_d)}$. This approach fits to a discretized anchored ANOVA decomposition with $a = 0$. To this end, we now define an infinite-dimensional univariate function space

$$V = \tilde{V}_{-1} \oplus \bigoplus_{i=0}^{\infty} \tilde{W}_i \tag{25}$$

and, with the choice $\mathbf{1}_i = \left(\tilde{V}_{-1}\right)_i$ and $\mathcal{C}_i = \left(\bigoplus_{j=0}^{\infty} \tilde{W}_j\right)_i$ in (21), we again obtain the splitting (22) which is now conform to the infinite-dimensional tensor product-hierarchical basis. In other words, if we use the alternative basis that is defined by the index set $\tilde{\mathbf{I}}_l$, the only univariate basis function ψ for which $P(\psi) \neq 0$ is $\psi = \phi_{-1,0}$ for $P(f) := f(0)$ and the anchored ANOVA decomposition completely fits to the hierarchical tensor product basis.

So far, the subspaces of the ANOVA decomposition are (up to the very first one) still infinite-dimensional and need to be further discretized. To this end, for a

[10] Note that $W_{\mathbf{l}}$ and $\tilde{W}_{\mathbf{l}}$ are the same for a multilevel index $\mathbf{l}$ with $l_j \ge 1$ for all $j = 1, \ldots, d$.

regular sparse grid of level t, we truncate each term of the ANOVA-decomposition as follows:

$$
\begin{aligned}
V_t^s = {} & \mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_d \\
& \oplus \bigoplus_{i=1}^{d} \bigoplus_{\substack{\tilde{n}_1(k_i) \le t \\ k_i \in \mathbb{N}}} \left(\mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_{i-1} \otimes \left(\tilde{W}_{k_i}\right)_i \otimes \mathbf{1}_{i+1} \otimes \ldots \otimes \mathbf{1}_d \right) \\
& \oplus \bigoplus_{i=1}^{d} \bigoplus_{j=i+1}^{d} \bigoplus_{\substack{\tilde{n}_2(k_i,k_j) \le t \\ k_i,k_j \in \mathbb{N}}} \left(\mathbf{1}_1 \otimes \ldots \otimes \left(\tilde{W}_{k_i}\right)_i \otimes \ldots \otimes \left(\tilde{W}_{k_j}\right)_j \otimes \ldots \otimes \mathbf{1}_d \right) \\
& \vdots \\
& \oplus \bigoplus_{\substack{\tilde{n}_d(k_1,\ldots,k_d) \le t \\ k_1,\ldots,k_d \in \mathbb{N}}} \left(\tilde{W}_{k_1}\right)_1 \otimes \ldots \otimes \left(\tilde{W}_{k_d}\right)_d .
\end{aligned}
$$

Thus, we discretize every k-variate component function of the ANOVA decomposition (23) with a regular k-dimensional sparse grid, where $k \in \{1, \ldots, d\}$.

A dimension-adaptive procedure can now be defined analogously to the space-adaptive algorithm. To this end, we employ the error indicator

$$\epsilon_{\mathbf{l}} := \max_{\mathbf{i} \in \tilde{\mathbf{I}}_{\mathbf{l}}} \epsilon_{\mathbf{l},\mathbf{i}}$$

which is just defined on the detail spaces $\tilde{W}_{\mathbf{l}}$ and does not rely on a single point anymore. For refinement we now simply insert *all* the points belonging to the basis functions of $\tilde{W}_{\mathbf{l}+\mathbf{e}_j}$ for each direction j with $l_j \neq -1$. Thus, we only insert grid nodes that lie in the same ANOVA component as nodes in $\tilde{W}_{\mathbf{l}}$. By doing this, refinement of the grid affects relevant ANOVA terms of the function but neglects higher-order terms. As in the case of spatial adaptivity, ancestors of new points have to be inserted into the grid.[11] The whole refinement procedure is iterated in the same way as previously.

Additionally, we compress the grid in an initial preprocessing step before starting the dimension-wise refinement procedure. To this end, for a given ε, starting with a regular sparse grid of level t, every detail space $\tilde{W}_{\mathbf{k}} \subset V_t^s$ for which $\epsilon_{\mathbf{k}} \le \varepsilon$ holds, is marked first. Then, if the points in a marked subspace $\tilde{W}_{\mathbf{k}}$ are not needed as an ancestor to a point in a non-marked subspace, all points belonging to $\tilde{W}_{\mathbf{k}}$ are removed from the grid. This compression is done, since a regular sparse grid of level 0 contains already part of each ANOVA component. Thus, it is not possible to

[11] For the one-dimensional case one simply defines $x_{0,1}$ to be the single child node of $x_{-1,0}$. The generalization to the multi-dimensional case is straightforward.

identify relevant ANOVA components of a function just by looking at a current adaptive grid. One would rather want to completely neglect components which do not contribute to the representation of a function and then start the adaptive procedure on a dimensionally reduced grid. The overall dimension-adaptive process is given in detail in Algorithm 3. Note that we start with a grid consisting of at least 2^d points. Since lower order ANOVA terms do not contain any information about the relevance of higher order terms, this is the only way to assure a correct treatment of every ANOVA component. An alternative strategy which starts with one grid point and iteratively adds ANOVA components can be found in [7].

Algorithm 3 The dimension-adaptive sparse grid algorithm using the basis defined by (24)

Input: starting level t, threshold ε, #iterations L, error indicators $\epsilon_{\mathbf{l}}$, time series $\left(s_j\right)_{j=1}^{N}$, embedding dimension d, regularization parameter λ
Output: dimension-adaptive sparse grid Ω^s_{adp}

initialize for compression: $Y \leftarrow \emptyset$
for all $\tilde{W}_{\mathbf{k}} \subset V_t^s$ **do**
  if $\epsilon_{\mathbf{k}} > \varepsilon$ **then**
    $Y \leftarrow Y \cup \left\{\mathbf{x}_{\mathbf{k},\mathbf{m}} \mid \mathbf{m} \in \tilde{\mathbf{I}}_{\mathbf{k}}\right\}$
  end if
end for
$Z \leftarrow Y$
for all $\mathbf{x}_{\mathbf{k},\mathbf{m}} \in Y$ **do**
  $Z \leftarrow Z \cup \text{AllAncestors}(\mathbf{k}, \mathbf{m}, d)$
end for

initialize for adaption: $\Omega^s_{\text{adp}} \leftarrow Z$, $\tilde{\Omega}^s_{\text{adp}} \leftarrow Z$, It $\leftarrow 0$
while It $< L$ **do**
  solve (17) on $\tilde{\Omega}^s_{\text{adp}}$
  for all $\tilde{W}_{\mathbf{k}}$ with $\mathbf{x}_{\mathbf{k},\mathbf{m}} \in \tilde{\Omega}^s_{\text{adp}}$ for each $\mathbf{m} \in \tilde{\mathbf{I}}_{\mathbf{k}}$ **do**
    if $\epsilon_{\mathbf{k}} > \varepsilon$ **then**
      for all $j \in \{1, \ldots, d\}$ **do**
        if $k_j > -1$ **then**
          $\Omega^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}} \cup \left\{\mathbf{x}_{\mathbf{k}+\mathbf{e}_j,\mathbf{m}} \mid \mathbf{m} \in \tilde{\mathbf{I}}_{\mathbf{k}+\mathbf{e}_j}\right\}$
        end if
      end for
    end if
  end for
  if $\tilde{\Omega}^s_{\text{adp}} = \Omega^s_{\text{adp}}$ **then**
    return Ω^s_{adp}
  end if
  $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}}$
  for all $\mathbf{x}_{\mathbf{k},\mathbf{m}}$ with odd m_j or $m_j = 0$ for each $j \in \{1, \ldots, d\}$ and $\mathbf{x}_{\mathbf{k},\mathbf{m}} \in \tilde{\Omega}^s_{\text{adp}}$ **do**
    $\Omega^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}} \cup \text{AllAncestors}(\mathbf{k}, \mathbf{m}, d)$
  end for
  $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}}$
  It $\leftarrow$ It$+1$
end while
return Ω^s_{adp}

5 Numerical Results

We will now present numerical results for our sparse grid algorithms when applied to both, synthetically constructed time series and series which stem from real world applications. All data has been properly scaled, such that the embedded points are situated in $\mathbf{H}_d$. The preconditioned conjugate gradient algorithm, which is used to solve the linear system (17), is always iterated until the quotient $||r_k||_{D^{-1}}/||r_0||_{D^{-1}}$ is smaller than 10^{-13} where D is the diagonal preconditioning matrix we used,[12] r_k denotes the residual of (17) after the k-th iteration and $||r_k||_{D^{-1}} := \sqrt{r_k^T D^{-1} r_k}$.

5.1 *Hénon Map in 2d*

First, we show results concerning the famous Hénon map

$$z_{n+1} := a - z_n^2 + bz_{n-1}, \tag{26}$$

see also [18]. Using the notation of Sect. 1 we have

$$\phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} a - x_1^2 + bx_2 \\ x_1 \end{pmatrix}$$

and

$$o\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = x_2,$$

where ϕ and o are defined on $\mathbb{R}^2$. It is easy to see that

$$\det \mathrm{D}\phi = -b,$$

where $\mathrm{D}\phi$ denotes the Jacobian matrix of ϕ. We will restrict ourselves to the most popular case $a = 1.4$, $b = 0.3$ for which the trajectory of the process approaches an attractor of non-integer box-counting dimension 1.26. As $|\det \mathrm{D}\phi| < 1$, the process is dissipative and the attractor is a compact subset of the ambient space. Therefore[13] we can apply Theorem 2.

A direct application would lead to the embedding dimension $d = \lfloor 2 \cdot 1.26 + 1 \rfloor = 3$. Nevertheless, we know from Eq. (26) that two dimensions are sufficient and will use $d = 2$ in our experiments instead of Takens' upper bound $d = 3$ to

[12] To this end, the system matrix from (17) is first transformed into the prewavelet basis, see e.g. [4], then, the inverse of its diagonal is taken as preconditioner.

[13] One can easily see that $\tilde{X}_l$ is finite for $l = 1, 2, 3$. Nevertheless, there exist points $\mathbf{x} \in \mathbb{R}^2$ for which $\left(\mathrm{D}\phi^l\right)_{\mathbf{x}}$ has eigenvalues with algebraic multiplicity 2 for $l = 2, 3$.

Table 1 Resulting parameters and errors after threefold cross-validation for regular sparse grids

T	t	$\log_2(\lambda)$	$\text{RMSE}_{\text{train}}$	$\text{RMSE}_{\text{test}}$
50	3	-17	$5.42 \cdot 10^{-3}$	$\mathbf{1.41 \cdot 10^{-2}}$
500	6	-25	$1.03 \cdot 10^{-4}$	$\mathbf{2.95 \cdot 10^{-4}}$
5,000	7	-22	$9.25 \cdot 10^{-5}$	$\mathbf{1.01 \cdot 10^{-4}}$

Table 2 Resulting parameters and errors after threefold cross-validation for the support vector machine

T	$\log_2(C)$	$\log_2(\gamma)$	$\log_2(\varepsilon)$	$\text{RMSE}_{\text{train}}$	$\text{RMSE}_{\text{test}}$
50	14	-5	-10	$1.46 \cdot 10^{-3}$	$\mathbf{1.60 \cdot 10^{-3}}$
500	10	1	-15	$2.51 \cdot 10^{-4}$	$\mathbf{2.57 \cdot 10^{-4}}$
5,000	8	3	-17	$2.07 \cdot 10^{-4}$	$\mathbf{2.06 \cdot 10^{-4}}$

a priori reduce the ambient space dimension as good as possible. The first $N = 20{,}000$ values of the Hénon map are taken into account to construct three different scenarios:

1. The first $T = 50$ points (training data) are used to learn the target function, the remaining $N - T = 19{,}950$ points (test data) are used to measure the forecasting error.
2. The first $T = 500$ points (training data) are used to learn the target function, the remaining $N - T = 19{,}500$ points (test data) are used to measure the forecasting error.
3. The first $T = 5{,}000$ points (training data) are used to learn the target function, the remaining $N - T = 15{,}000$ points (test data) are used to measure the forecasting error.

We compare our regular sparse grid approach to a standard support vector machine regression algorithm using radial basis functions (SVM = RBF ε-SVR). To find appropriate parameters we use three-fold cross-validation. We investigated $t \in \{2, \ldots, 8\}$ and $\lambda \in \{2^{-1}, \ldots, 2^{-25}\}$ for the sparse grid algorithm. For calculations concerning the SVM approach, we used libsvm, see [3] for implementations and parameters. Here, we performed a three-fold cross-validation over the parameters $C \in \{2^0, \ldots, 2^{15}\}$, $\varepsilon \in \{2^{-20}, \ldots, 2^{-1}\}$ and the kernel width $\gamma \in \{2^{-10}, \ldots, 2^5\}$ of the RBF ε-SVR. To measure the forecasting error of any function f on the embedded test data we used the root mean squared error (RMSE). Given the test data $(s_j)_{j=T+1}^{N}$, we can build the embedded vectors $\mathbf{t}_j$ as in (5) for $T + d \leq j \leq N - 1$. Then

$$\text{RMSE}_{\text{test}} := \sqrt{\frac{1}{N - T - d} \sum_{j=T+d}^{N-1} \big(s_{j+1} - f(\mathbf{t}_j)\big)^2}.$$

For the training data we define $\text{RMSE}_{\text{train}}$ analogously. The results, i.e. the determined best parameter values and the corresponding errors on training and test data for these parameters, are given in Table 1 for regular sparse grids and in Table 2 for

the support vector machine. We observe that the SVM algorithm performs somewhat better than the sparse grid algorithm for the moderate value $T = 50$. But as we increase the size of the training data set, the results for the sparse grid algorithm get successively better. For $T = 5{,}000$, the sparse grid algorithm reaches a slightly lower error than the RBF-SVM method while the involved computational costs are substantially less anyway. The sparse grid method is able to use the newly added training data points to discover more structure of the underlying process.

Note here that the number of possible parameter combinations is not the same for the sparse grid method and the SVM. Thus, it is not representative to compare the runtimes of their cross-validation processes. Nevertheless, this comparison gives a hint of the behavior of the overall runtime when changing T: The cross-validation process for the SVM algorithm was about 20 times faster than that of the sparse grid approach for $T = 50$. For $T = 500$ the runtimes were almost equal and for $T = 5{,}000$ the cross-validation process of the sparse grid algorithm was more than three times faster than that of SVM.

In summary, while the SVM algorithm is favorable for few training points, the benefits of the sparse grid algorithm with respect to both, achieved accuracy and necessary computational cost, begin to prevail in situations with more and more training points.

5.2 *Jump Map in 5d*

In this experiment we want to show the advantages of the space- and dimension-adaptive sparse grid algorithms compared to the regular sparse grid approach.

Following the rule

$$z_{n+1} := (z_n + z_{n-1}) \mod 1, \tag{27}$$

we get a time series by

$$\phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} (x_1 + x_2) \mod 1 \\ x_1 \end{pmatrix}$$

and

$$o\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = x_1,$$

where ϕ and o are defined on $[0, 1)^2$. Due to the modulo operation, ϕ has a jump at all points $\{(x_1, x_2)^T \in [0, 1)^2 \mid x_1 + x_2 = 1\}$. For each other point of the domain the process is conservative, i.e. $|\det \mathrm{D}\phi| = 1$. Since ϕ is not diffeomorphic, Theorem 2 cannot be invoked. Nevertheless, we will apply the delay embedding scheme and test if the sparse grid solutions are still able to give a suitable predictor for (27).

Again, we use the first $N = 20{,}000$ values of the time series and construct three scenarios with the same values of T, i.e. $T = 50, 500, 5{,}000$, as for the Hénon map. We now assume that we were just given the time series of length N and do

not know anything about the underlying process (27). Thus we have to estimate the dimension m to be able to use the delay embedding scheme with $d = \lfloor 2m + 1 \rfloor$ before applying our sparse grid algorithms. For the small training data set of size $T = 50$ we get an estimate of $m \approx 2.24$ with the Grassberger-Procaccia dimension estimator. For the other two scenarios, the estimated dimension is even closer to 2. Taking $m = 2$ determines our embedding dimension to be $d = 5$ and we therefore build the embedded vectors $\mathbf{t}_j$ in $\mathbb{R}^5$.

We use $\lambda = 10^{-4}$ in the following experiments. The results for a regular grid discretization and for both, a space- and a dimension-adaptive procedure with $\epsilon = 0.1$ and starting level 1, are given in Table 3.

With substantially fewer grid points, both adaptive algorithms achieve the same or even better RMSE values than the regular sparse grid method. The remarkably smaller amount of grid points used in the dimension-adaptive variant is due to the compression step before refinement starts.

Furthermore, note that the dimension-adaptive algorithm is able to reveal the lower-dimensional structure of the embedded process. We can observe from the constructed forecasting function how many grid points have been spent on its different ANOVA components. To this end, counting all grid points with exactly one non-zero coordinate we get the number of points spent on the representation of univariate functions in the ANOVA decomposition. We can continue analogously for bivariate functions (two non-zero coordinates) and so on.

In Fig. 2 we see the distribution of grid points among the ANOVA components of different order for the example of the $5d$ jump map. The dimension-adaptive algorithm successfully detected that there is no term of fifth order and thus grid points are only spent on the boundary of $\mathbf{H}_5$. Terms of third and fourth order are still present, but one observes that most grid points have been used for terms of order 1 and 2. Altogether, the inherent structure of the process was well detected by the algorithm.

As less points are spent by the space- and the dimension-adaptive algorithm there is a significant saving in storage for these methods. In addition, also the absolute runtime of the dimension-adaptive algorithm—especially for the case of few training points and high levels—is better than for the regular sparse grid case, as we see in Table 4. The runtimes of the space-adaptive method and the regular sparse grid algorithm are of the same order for the listed scenarios.

5.3 *Small Dataset of the ANN and CI Forecasting Competition 2006/2007*

We now consider the performance of the regular and the space-adaptive sparse grid algorithm in a practical application.[14] The reduced dataset of the "Artificial Neural

[14] Since we restricted ourselves to $d \leq 3$ in this experiment, we did not apply the dimension-adaptive algorithm to this problem.

Table 3 RMSE for a regular, a space-, and a dimension-adaptive sparse grid discretization for the jump map in $5d$

(**a**) Regular sparse grid (SG) of level t

T	t	$\lvert\Omega_t^s\rvert$	$\text{RMSE}_{\text{train}}$	$\text{RMSE}_{\text{test}}$
50	3	3,753	$7.67 \cdot 10^{-2}$	$\mathbf{2.34 \cdot 10^{-1}}$
50	4	12,033	$4.91 \cdot 10^{-2}$	$\mathbf{2.20 \cdot 10^{-1}}$
50	5	36,033	$2.57 \cdot 10^{-2}$	$\mathbf{1.47 \cdot 10^{-1}}$
500	3	3,753	$1.12 \cdot 10^{-1}$	$\mathbf{1.41 \cdot 10^{-1}}$
500	4	12,033	$7.12 \cdot 10^{-2}$	$\mathbf{1.11 \cdot 10^{-1}}$
500	5	36,033	$4.39 \cdot 10^{-2}$	$\mathbf{8.53 \cdot 10^{-2}}$
5,000	3	3,753	$1.27 \cdot 10^{-1}$	$\mathbf{1.32 \cdot 10^{-1}}$
5,000	4	12,033	$9.11 \cdot 10^{-2}$	$\mathbf{9.61 \cdot 10^{-2}}$
5,000	5	36,033	$6.43 \cdot 10^{-2}$	$\mathbf{6.99 \cdot 10^{-2}}$

(**b**) Space-adp. SG after $\#It$ iterations

T	$\#It$	$\lvert\Omega_{\text{adp}}^s\rvert$	$\text{RMSE}_{\text{train}}$	$\text{RMSE}_{\text{test}}$
50	3	3,159	$7.68 \cdot 10^{-2}$	$\mathbf{2.34 \cdot 10^{-1}}$
50	4	4,887	$5.03 \cdot 10^{-2}$	$\mathbf{2.14 \cdot 10^{-1}}$
50	5	5,535	$2.75 \cdot 10^{-2}$	$\mathbf{1.33 \cdot 10^{-1}}$
500	3	2,349	$1.12 \cdot 10^{-1}$	$\mathbf{1.40 \cdot 10^{-1}}$
500	4	3,645	$7.21 \cdot 10^{-2}$	$\mathbf{1.10 \cdot 10^{-1}}$
500	5	4,293	$4.66 \cdot 10^{-2}$	$\mathbf{8.32 \cdot 10^{-2}}$
5,000	3	2,079	$1.27 \cdot 10^{-1}$	$\mathbf{1.32 \cdot 10^{-1}}$
5,000	4	2,727	$9.14 \cdot 10^{-2}$	$\mathbf{9.61 \cdot 10^{-2}}$
5,000	5	3,051	$6.47 \cdot 10^{-2}$	$\mathbf{6.96 \cdot 10^{-2}}$

(**c**) Dim.-adp. SG after $\#It$ iterations

T	$\#It$	$\lvert\Omega_{\text{adp}}^s\rvert$	$\text{RMSE}_{\text{train}}$	$\text{RMSE}_{\text{test}}$
50	2	576	$7.90 \cdot 10^{-2}$	$\mathbf{2.43 \cdot 10^{-1}}$
50	3	704	$5.26 \cdot 10^{-2}$	$\mathbf{2.22 \cdot 10^{-1}}$
50	4	832	$2.93 \cdot 10^{-2}$	$\mathbf{1.39 \cdot 10^{-1}}$
500	2	302	$1.15 \cdot 10^{-1}$	$\mathbf{1.40 \cdot 10^{-1}}$
500	3	368	$7.76 \cdot 10^{-2}$	$\mathbf{1.08 \cdot 10^{-1}}$
500	4	424	$5.29 \cdot 10^{-2}$	$\mathbf{8.12 \cdot 10^{-2}}$
5,000	2	310	$1.28 \cdot 10^{-1}$	$\mathbf{1.32 \cdot 10^{-1}}$
5,000	3	326	$9.21 \cdot 10^{-2}$	$\mathbf{9.60 \cdot 10^{-2}}$
5,000	4	342	$6.55 \cdot 10^{-2}$	$\mathbf{6.96 \cdot 10^{-2}}$

Network (ANN) and Computational Intelligence (CI) Forecasting Competition 2006/2007" consists of 11 empirical business time series.[15] Each of the time series consists of 144 real values where the first 126 values should be used as training data. The goal of the competition is to forecast the last 18 consecutive values. The symmetric mean absolute percent error

[15]Further information concerning the setting and the dataset can be found at http://www.neural-forecasting-competition.com/NN3/index.htm.

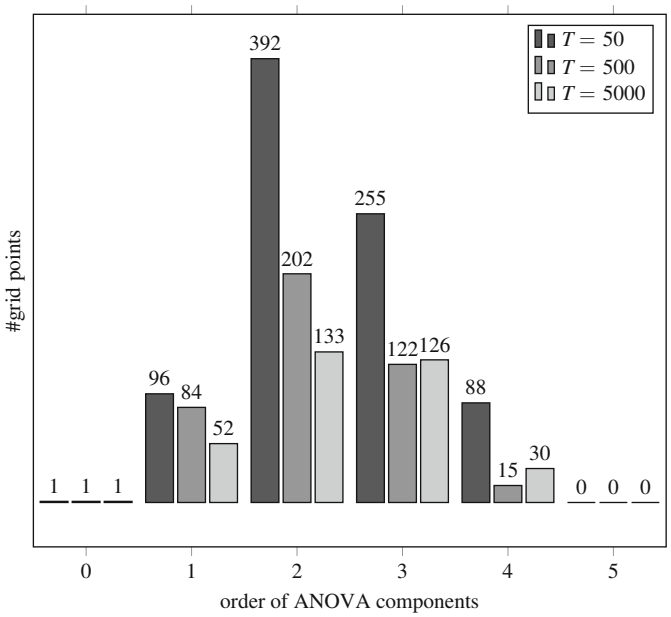


Fig. 2 Distribution of grid points among ANOVA components of different order for the $5d$ jump map and the dimension-adaptive algorithm

Table 4 Comparison of runtimes for the space- and the dimension-adaptive sparse grid algorithm. $R_{\text{dimadp}}(t, \#It)$ is defined as $\frac{R_{\text{reg}}(t)}{R_{\text{dimadp}}(\#It)}$, where $R_{\text{reg}}(t)$ denotes the runtime of the regular sparse grid algorithm with level t and $R_{\text{dimadp}}(\#It)$ denotes the runtime of $\#It$ iterations of the dimension-adaptive algorithm; $R_{\text{spadp}}(t, \#It)$ is defined analogously for the space-adaptive algorithm

T	t	$\#It$	$R_{\text{spadp}}(t, \#It)$	$R_{\text{dimadp}}(t, \#It)$
50	4	3	0.73	5.05
50	5	4	1.57	13.09
500	4	3	0.70	3.67
500	5	4	1.40	9.51
5,000	4	3	0.61	1.29
5,000	5	4	0.94	1.78

$$\text{SMAPE} := \frac{1}{18} \sum_{j=127}^{144} \frac{2|s_j - \hat{s_j}|}{|s_j| + |\hat{s}_j|}$$

determines the quality of the forecast of one particular time series. The arithmetic mean of the SMAPEs for each of the 11 time series determines the ranking of the algorithm. Here $\hat{s}_j$ denotes the prediction of the j-th test data value. As consecutive values have to be predicted, we cannot simply compute $\hat{s}_j = f(\mathbf{t}_{j-1})$ for a general,

Table 5 SMAPE and average SMAPE for the 11 time series of the reduced dataset of the ANN and CI Forecasting Competition 2006/2007

	(**a**) Regular sparse grids					(**b**) Space-adaptive sparse grids		
Time Series	t	$\log_2(\lambda)$	d	k	SMAPE in %	$\log_2(\lambda_{\text{adp}})$	k_{adp}	$\text{SMAPE}_{\text{adp}}$ in %
1	7	−13	1	12	2.8398	−11	14	**2.6309**
2	6	−10	2	11	25.6872	−8	11	**23.0448**
3	2	−12	2	11	**30.6692**	−9	10	39.9194
4	4	−9	2	12	**6.3690**	−9	10	8.8058
5	3	−1	1	1	**3.3801**	−10	11	5.1977
6	4	−10	2	10	**4.9186**	−8	9	5.6765
7	4	−13	1	1	6.7220	−3	2	**4.2933**
8	2	−8	2	14	30.3151	−1	12	**27.7111**
9	6	−12	1	4	11.2487	−4	3	**10.2458**
10	7	−6	3	10	**30.3352**	−6	10	30.7120
11	2	−15	2	11	18.5016	−5	11	**16.2794**
Av. SMAPE					**15.5442**			15.8651

computed forecasting function f since the coordinates of the embedded vector $\mathbf{t}_{j-1}$ might not only stem from training data but also from unknown test data. Therefore, we recursively define $\hat{s}_j = f(\hat{\mathbf{t}}_{j-1})$ with $\hat{\mathbf{t}}_{j-1} = \left(s_{j-d}, s_{j-d+1}, \ldots, \hat{s}_{j-2}, \hat{s}_{j-1}\right)^T$ where a coordinate is set to s_l if $l \leq 126$ and to $\hat{s}_l$ otherwise. Furthermore, we introduce the time step size k which determines which future value is learned. So, by building the vectors $\mathbf{t}_j$ from training data, we are learning s_{j+k}. In the examples introduced earlier we always had set $k = 1$.

In our experiments with the regular sparse grid approach, the first 108 values of a time series are used to learn a prediction model which is then evaluated on the last 18 values of the training data. This way, we determine the best combination of a regularization parameter $\lambda \in \{2^{-15}, \ldots, 2^{-1}\}$, a level $t \in \{2, \ldots, 7\}$, an embedding dimension $d \in \{1, 2, 3\}$ and a future step size $k \in \{1, \ldots, 18\}$. These parameters are then employed to learn a model using the whole training data set. This model is finally taken to predict the 18 values of the test data set. Proceeding in this fashion for every time series we achieve the SMAPEs that can be found in Table 5**a**.

Alternatively, we fixed $d = 3$ and $t = 1$, started the space-adaptive algorithm with a maximum iteration count of 7. We chose the optimal k and λ in the same fashion as for the regular sparse grid algorithm. The results are shown in Table 5**b**. Even though the space-adaptive algorithm performs better than the non-adaptive method for 6 of the 11 time series, the average SMAPE of the non-adaptive variant is still smaller. This is mostly due to the bad performance of the space-adaptive algorithm for time series 3. Anyway, with each of the two methods we perform better than 36 of the 44 participants of the competition. Furthermore, we also outperform 8 of the 13 statistical and CI methods that entered the competition as benchmarks. Note that the best score achieved in the competition was an average SMAPE of 13.07%.

By combining our two methods such that for every time series the SMAPE on the last 18 values of the training data decides if the space-adaptive or the non-adaptive

variant is chosen, we achieve an average SMAPE of 15.3082%. With this result we outscore one more participant and one more statistical benchmark.

Thus, even when dealing with rather short empirical time series where data-based approaches like SVM seem to be a more natural and promising approach, our methods still achieve competitive results.

6 Concluding Remarks

In this article we introduced a sparse grid-based discretization approach to solve the forecasting problem for time series. We gave a short review of Takens' theorem and showed how it can be applied to the field of time series forecasting if the box-counting dimension of the attractor is a priori known or at least properly estimated from the given data. We motivated the use of a regularized quadratic loss-functional and emphasized the difference between kernel-based approaches and arbitrary basis discretizations for the case of reproducing kernel Hilbert spaces. To avoid the curse of dimensionality we introduced regular sparse grids based on piecewise linear B-splines. Space- and dimension-adaptive refinement proved to be useful enhancements, which further reduce costs as e.g. most of the attractors are not uniformly spread across the embedding space. Finally, we have computed numerical results which showed that our algorithms achieve the same or even better results than common SVM-based regression algorithms. The experiments also proved that dimension-adaptive refinement is useful if the embedding dimension of the attractor has been overestimated.

The problem of finding reliable estimates for the box-counting dimension of an attractor has not been discussed in this paper. This is a crucial step for the application of Takens' theorem to real world data. In our further experiments, the Grassberger-Procaccia algorithm proved quite successful to this end.

Another main problem that leads to non-stationarity is noise. Almost all real world data are non-deterministic because of the occurrence of noise as a stochastic component. Several noise-reduction methods and dimension estimators can be found in the literature, see [21] for an overview. These techniques will be incorporated into the sparse grid approach in the future.

Note finally that there are similarities to other existing methods: In [5], a sparse grid approach for manifold learning applications was suggested. There also are links to a density estimation method with grid-based discretizations, see [14]. An approach that is closely related, but approximates the sparse grid solution by a combination technique, can be found in [6]. A comparison of cost complexity and achieved error between our approach and this technique has still to be done.

Finally, since the curse of dimensionality is still present with respect to the sparse grid level, it is desirable to reduce the dimension of the problem beforehand as good as possible by linear techniques like the well-known SVD [11] or the LT approach of [20].

References

1. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.
2. M. Casdagli, T. Sauer, and J. Yorke. Embedology. *Journal of Statistical Physics*, 65:576–616, 1991.
3. C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
4. C. Feuersänger. *Sparse Grid Methods for Higher Dimensional Approximation*. PhD thesis, Institute for Numerical Simulation, University of Bonn, 2010.
5. C. Feuersänger and M. Griebel. Principal manifold learning by sparse grids. *Computing*, 85(4), 2009.
6. J. Garcke. *Maschinelles Lernen durch Funktionsrekonstruktion mit verallgemeinerten dünnen Gittern*. PhD thesis, Institute for Numerical Simulation, University of Bonn, 2004.
7. J. Garcke. A dimension adaptive combination technique using localised adaptation criteria. In H. Bock, X. Hoang, R. Rannacher, and J. Schlöder, editors, *Modeling, Simulation and Optimization of Complex Processes*, pages 115–125. Springer, 2012.
8. J. Garcke, T. Gerstner, and M. Griebel. Intraday foreign exchange rate forecasting using sparse grids. In J. Garcke and M. Griebel, editors, *Sparse grids and applications*, 81–105, 2012.
9. J. Garcke and M. Hegland. Fitting multidimensional data using gradient penalties and the sparse grid combination technique. *Computing*, 84(1–2):1–25, 2009.
10. T. Gerstner and M. Griebel. Dimension–adaptive tensor–product quadrature. *Computing*, 71(1):65–87, 2003.
11. G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.
12. P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. *Physica*, D9:189–208, 1983.
13. M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.
14. M. Griebel and M. Hegland. A finite element method for density estimation with Gaussian priors. *SIAM Journal on Numerical Analysis*, 47(6), 2010.
15. M. Griebel and P. Oswald. Tensor product type subspace splitting and multilevel iterative methods for anisotropic problems. *Adv. Comput. Math.*, 4:171–206, 1995.
16. M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens, editors, *Iterative Methods in Linear Algebra*, pages 263–281. IMACS, Elsevier, North Holland, 1992.
17. M. Hegland. Adaptive sparse grids. *ANZIAM J.*, 44:C335–C353, 2003.
18. M. Hénon. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 50:69–77, 1976.
19. J. Huke. Embedding nonlinear dynamical systems: A guide to Takens' theorem, 2006. Manchester Institute for Mathematical Sciences EPrint: 2006.26.
20. J. Imai and K. Tan. Minimizing effective dimension using linear transformation. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, pages 275–292. Springer, 2004.
21. H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, 2004. 2nd edition.
22. A. Krueger. Implementation of a fast box-counting algorithm. *Computer Physics Communications*, 98:224–234, 1996.
23. L. Liebovitch and T. Toth. A fast algorithm to determine fractal dimensions by box counting. *Physics Letters A*, 141(8,9):386–390, 1989.
24. B. Schölkopf and A. Smola. *Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press – Cambridge, Massachusetts, 2002.

25. F. Takens. Detecting strange attractors in turbulence. *Dynamical Systems and Turbulence, Lecture Notes in Mathematics*, (898):366–381, 1981.
26. J. Theiler. Efficient algorithm for estimating the correlation dimension from a set of discrete points. *Physical Review A*, 36(9):4456–4462, 1987.
27. A. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963.
28. G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series In Applied Mathematics*. SIAM: Society for Industrial and Applied Mathematics, 1990.

Efficient Analysis of High Dimensional Data in Tensor Formats

Mike Espig, Wolfgang Hackbusch, Alexander Litvinenko, Hermann G. Matthies, and Elmar Zander

Abstract In this article we introduce new methods for the analysis of high dimensional data in tensor formats, where the underling data come from the stochastic elliptic boundary value problem. After discretisation of the deterministic operator as well as the presented random fields via KLE and PCE, the obtained high dimensional operator can be approximated via sums of elementary tensors. This tensors representation can be effectively used for computing different values of interest, such as maximum norm, level sets and cumulative distribution function. The basic concept of the data analysis in high dimensions is discussed on tensors represented in the canonical format, however the approach can be easily used in other tensor formats. As an intermediate step we describe efficient iterative algorithms for computing the characteristic and sign functions as well as pointwise inverse in the canonical tensor format. Since during majority of algebraic operations as well as during iteration steps the representation rank grows up, we use lower-rank approximation and inexact recursive iteration schemes.

1 Introduction

Let us give an example which motivates much of the following formulation and development. Assume that we are interested in the time evolution of some system, described by

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = A(p)(u(t)), \tag{1}$$

M. Espig (✉) · W. Hackbusch
Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany
e-mail: mike.espig@mis.mpg.de

A. Litvinenko · H.G. Matthies · E. Zander
Technische Universität Braunschweig, Braunschweig, Germany
e-mail: wire@tu-bs.de

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_2,

where $u(t)$ is in some Hilbert space $\mathcal{U}$ and $A(p)$ is some parameter dependent operator; in particular $A(p)$ could be some parameter-dependent differential operator, for example

$$\frac{\partial}{\partial t}u(x,t) = \nabla \cdot (\kappa(x,\omega)\nabla u(x,t)) + f(x,t), \quad x \in \mathcal{G} \subset \mathbb{R}^d, t \in [0,T] \tag{2}$$

where $\kappa(x,\omega)$ is a random field dependent on a random parameter in some probability space $\omega \in \Omega$, and one may take $\mathcal{U} = L_2(\mathcal{G})$.

One may for each $\omega \in \Omega$ seek for solutions in $L_2([0,T],\mathcal{U}) \cong L_2([0,T]) \otimes \mathcal{U}$. Assigning

$$\mathcal{S} = L_2([0,T]) \otimes L_2(\Omega),$$

one is looking for a solution in $\mathcal{U} \otimes \mathcal{S}$. $L_2(\Omega)$ can for random fields be further decomposed

$$L_2(\Omega) = L_2(\bigotimes_j \Omega_j) \cong \bigotimes_j L_2(\Omega_j) \cong \bigotimes_j L_2(\mathbb{R}, \Gamma_j).$$

with some measures Γ_j. Then the parametric solution is sought in the space

$$\mathcal{U} \otimes \mathcal{S} = L_2(\mathcal{G}) \otimes \left(L_2([0,T]) \otimes \bigotimes_j L_2(\mathbb{R}, \Gamma_j) \right). \tag{3}$$

The more tensor factors there are, the more difficult and high-dimensional the problem will be. But on the other hand a high number of tensor factors in Eq. (3) will also allow very sparse representation and highly effective algorithms—this is of course assuming that the solution is intrinsically on a low-dimensional manifold and we 'just' need to discover it.

This paper is about exploiting the tensor product structure which appears in Eq. (3) for efficient calculations to be performed on the solution. This tensor product structure—in this case multiple tensor product structure—is typical for such parametric problems. What is often desired, is a representation which allows for the approximate evaluation of the state of Eqs. (1) or (2) without actually solving the system again. Sometimes this is called a 'response surface'. Furthermore, one would like this representation to be inexpensive to evaluate, and for it to be convenient for certain post-processing tasks, for example like finding the minimum or maximum value over some or all parameter values.

1.1 Tensorial Quantities

Computations usually require that one chooses finite dimensional subspaces and bases in there, in the example case of Eq. (2) these are

$$\begin{aligned}
\text{span}\,\{\varphi_n\}_{n=1}^{N} &= \mathcal{U}_N \subset \mathcal{U}, \quad \dim \mathcal{U}_N = N,\\
\text{span}\,\{\tau_k\}_{k=1}^{K} &= \mathcal{T}_K \subset L_2([0,T]) = \mathcal{S}_I, \quad \dim \mathcal{T}_K = K,\\
\forall m &= 1,\dots,M:\\
\text{span}\,\{X_{j_m}\}_{j_m=1}^{J_m} &= \mathcal{S}_{II,J_m} \subset L_2(\mathbb{R},\Gamma_m) = \mathcal{S}_{II}, \quad \dim \mathcal{S}_{II,J_m} = J_m.
\end{aligned}$$

Let $\mathcal{P} := [0,T] \times \Omega$, an approximation to $u : \mathcal{P} \to \mathcal{U}$ is thus given by

$$u(x,t,\omega_1,\dots,\omega_M) \approx \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{j_1=1}^{J_1}\cdots\sum_{j_M=1}^{J_M} \hat{u}_{n,k}^{j_1,\dots,j_M}\varphi^n(x)\otimes\tau^k(t)\otimes\left(\bigotimes_{m=1}^{M} X_{j_m}(\omega_m)\right). \tag{4}$$

Via Eq. (4) the tensor $\hat{u}_{n,k}^{j_1,\dots,j_M}$ represents the state $u(x,t,\omega_1,\dots,\omega_M)$ and is thus a concrete example of a 'response surface'.

To allow easier interpretation later, assume that $\{x_1,\dots,x_N\} \subset \mathcal{G}$ are unisolvent points for $\{\varphi_n\}_{n=1}^{N}$, and similarly $\{t_1,\dots,t_K\} \subset [0,T]$ are unisolvent points for $\{\tau_k\}_{k=1}^{K}$, and for each $m = 1,\dots,M$ the points $\{\omega_m^1,\dots,\omega_M^{J_m}\} \subset \Omega_m$ are unisolvent points for $\{X_{j_m}\}_{j_m=1}^{J_m}$. Then the same information which is in Eq. (4) is also contained in the evaluation at those unisolvent points:

$$\begin{aligned}
&\forall n = 1,\dots,N,\ k = 1,\dots,K,\ m = 1,\dots,M,\ j_m = 1,\dots,J_m:\\
&u_{n,k}^{j_1,\dots,j_m,\dots,j_M} = u(x_n,t_k,\omega_1^{j_1},\dots,\omega_m^{j_m},\dots,\omega_M^{J_M}),
\end{aligned} \tag{5}$$

this is just a different choice of basis for the tensor. In keeping with symbolic index notation, we denote by $(u_{n,k}^{j_1,\dots,j_m,\dots,j_M})$ the whole tensor in Eq. (5).

Model reduction or sparse representation may be applied before, during, or after the computation of the solution to Eq. (1) for new values of t or $(\omega_1,\dots,\omega_M)$. It may be performed in a pure Galerkin fashion by choosing even smaller, but well adapted subspaces, say for example $\mathcal{U}_{N'} \subset \mathcal{U}_N$, and thus reducing the dimensionality and hopefully also the work involved in a new solution. This is sometimes termed 'flat' Galerkin. In this kind of reduction, the subspace $\mathcal{U}_{N''} = \mathcal{U}_N \ominus \mathcal{U}_{N'}$ is completely neglected.

In nonlinear Galerkin methods, the part $u_{N'} \in \mathcal{U}_{N'}$ is complemented by a possibly non-linear map $\upsilon : \mathcal{U}_{N'} \to \mathcal{U}_{N''}$ to $u_N \approx u_{N'} + \upsilon(u_{N'}) \in \mathcal{U}_{N'} \oplus \mathcal{U}_{N''} = \mathcal{U}_N$. The approximate solution is not in a flat subspace anymore, but in some possibly non-linear manifold, hence the name. Obviously this procedure may be applied to any of the approximating subspaces.

Another kind of reduction works directly with the tensor $(u_{n,k}^{j_1,\dots,j_M})$ in Eq. (5). It has formally $R'' = N \times K \times \prod_{m=1}^{M} J_m$ terms. The minimum number R of terms needed to represent the sum is defined as the rank of that tensor. One might try to

approximately express the sum with even fewer $R' \ll R \leq R''$ terms, this is termed a low-rank approximation. It may be seen as a non-linear model reduction.

In this way the quantity in Eq. (5) is expressed as

$$(u_{n,k}^{j_1,\ldots,j_m,\ldots,j_M}) \approx \sum_{\rho=1}^{R'} u^{\rho} \varphi_{\rho} \otimes \tau_{\rho} \otimes \left(\bigotimes_{m=1}^{M} X_{\rho_m} \right), \tag{6}$$

where $\varphi_{\rho} \in \mathbb{R}^N$, $\tau_{\rho} \in \mathbb{R}^K$, and for each $m = 1, \ldots, M$: $X_{\rho_m} \in \mathbb{R}^{J_m}$.

Hence Eq. (6) is an approximation for the response, another—sparse—'response surface'. With such a representation, one wants to perform numerous tasks, among them

- Evaluation for specific parameters $(t, \omega_1, \ldots, \omega_M)$,
- Finding maxima and minima,
- Finding 'level sets'.

2 Discretisation of Diffusion Problem with Uncertain Coefficient

Since the time dependence in Eq. (1) doesn't influence on the proposed further methods we demonstrate our theoretical and numerical results on the following stationary example

$$\begin{aligned} -\mathrm{div}(\kappa(x,\omega)\nabla u(x,\omega)) &= f(x,\omega) \quad \text{a.e. } x \in \mathcal{G}, \quad \mathcal{G} \subset \mathbb{R}^2, \\ u(x,\omega) &= 0 \qquad\qquad\quad \text{a.e. } x \in \partial\mathcal{G}. \end{aligned} \tag{7}$$

This is a stationary diffusion equation described by a conductivity parameter $\kappa(x, \omega)$. It may, for example, describe the groundwater flow through a porous subsurface rock/sand formation [6, 17, 22, 37, 48]. Since the conductivity parameter in such cases is poorly known, i.e. it may be considered as uncertain, one may model it as a random field.

Let us introduce a bounded spatial domain of interest $\mathcal{G} \subset \mathbb{R}^d$ together with the hydraulic head u appearing in *Darcy's* law for the seepage flow $-\kappa\nabla u$, and f as flow sinks and sources. For the sake of simplicity we only consider a scalar conductivity, although a conductivity tensor would be more appropriate. The conductivity κ and the source f are defined as random fields over the probability space Ω. By introduction of this stochastic model of uncertainties Eq. (7) is required to hold almost surely in ω, i.e. $\mathbb{P}$-almost everywhere.

As the conductivity κ has to be positive, and is thus restricted to a particular case in a vector space, we consider its logarithm as the primary quantity, which may have any value. We assume that it has finite variance and thus choose for maximum

entropy a Gaussian distribution. Hence the conductivity is initially log-normally distributed. Such kind of assumption is known as a priori information/distribution:

$$\kappa(x) := \exp(q(x)), \quad q(x) \sim N(0, \sigma_q^2). \tag{8}$$

In order to solve the stochastic forward problem we assume that $q(x)$ has covariance function of the exponential type $\mathrm{Cov}_q(x, y) = \sigma_q^2 \exp(-|x-y|/l_c)$ with prescribed covariance length l_c.

In order to make sure that the numerical methods will work well, we strive to have similar overall properties of the stochastic system Eq. (7) as in the deterministic case (for fixed ω). For this to hold, it is necessary that the operator implicitly described by Eq. (7) is continuous and continuously invertible, i.e. we require that both $\kappa(x, \omega)$ and $1/\kappa(x, \omega)$ are essentially bounded (have finite L_∞ norm) [2, 34, 37, 44]:

$$\kappa(x, \omega) > 0 \quad \text{a.e.}, \quad \|\kappa\|_{L_\infty(\mathcal{G}\times\Omega)} < \infty, \quad \|1/\kappa\|_{L_\infty(\mathcal{G}\times\Omega)} < \infty. \tag{9}$$

Two remarks are in order here: one is that for a heterogeneous medium each realisation $\kappa(x, \omega)$ should be modelled as a tensor field. This would entail a bit more cumbersome notation and not help to explain the procedure any better. Hence for the sake of simplicity we stay with the unrealistically simple model of a scalar conductivity field. The strong form given in Eq. (7) is not a good starting point for the Galerkin approach. Thus, as in the purely deterministic case, a variational formulation is needed, leading—via the *Lax-Milgram* lemma—to a well-posed problem. Hence, we search for $u \in \mathscr{U} := \mathcal{U} \otimes \mathcal{S}$ such that for all $v \in \mathscr{U}$ holds:

$$\mathbf{a}(v, u) := \mathbb{E}\left(\mathsf{a}(\omega)(v(\cdot, \omega), u(\cdot, \omega))\right) = \mathbb{E}\left(\langle \ell(\omega), v(\cdot, \omega)\rangle\right) =: \langle\!\langle \boldsymbol{\ell}, v\rangle\!\rangle. \tag{10}$$

Here $\mathbb{E}(b) := \mathbb{E}(b(\omega)) := \int_\Omega b(\omega)\ \mathbb{P}(\mathrm{d}\omega)$ is the expected value of the random variable (RV) b. The double bracket $\langle\!\langle \cdot, \cdot \rangle\!\rangle_{\mathscr{U}}$ is interpreted as duality pairing between $\mathscr{U}$ and its dual space $\mathscr{U}^*$.

The bi-linear form $\mathbf{a}$ in Eq. (10) is defined using the usual deterministic bi-linear (though parameter-dependent) form:

$$\mathsf{a}(\omega)(v, u) := \int_{\mathcal{G}} \nabla v(x) \cdot (\kappa(x, \omega) \nabla u(x))\ \mathrm{d}x, \tag{11}$$

for all $u, v \in \mathcal{U} := \mathring{H}^1(\mathcal{G}) = \{u \in H^1(\mathcal{G}) \mid u = 0 \text{ on } \partial\mathcal{G}\}$. The linear form $\boldsymbol{\ell}$ in Eq. (10) is similarly defined through its deterministic but parameter-dependent counterpart:

$$\langle \ell(\omega), v\rangle := \int_{\mathcal{G}} v(x) f(x, \omega)\ \mathrm{d}x, \quad \forall v \in \mathcal{U}, \tag{12}$$

where f has to be chosen such that $\ell(\omega)$ is continuous on $\mathcal{U}$ and the linear form $\boldsymbol{\ell}$ is continuous on $\mathscr{U}$, the Hilbert space tensor product of $\mathcal{U}$ and $\mathcal{S}$.

Let us remark that—loosely speaking—the stochastic weak formulation is just the expected value of its deterministic counterpart, formulated on the Hilbert tensor

product space $\mathcal{U} \otimes \mathcal{S}$, i.e. the space of $\mathcal{U}$-valued RVs with finite variance, which is isomorphic to $L_2(\Omega, \mathbb{P}; \mathcal{U})$. In this way the stochastic problem can have the same theoretical properties as the underlying deterministic one, which is highly desirable for any further numerical approximation.

2.1 Spatial Discretisation

Let us discretise the spatial part of Eq. (10) by a standard finite element method. However, any other type of discretisation technique may be used with the same success. Since we deal with Galerkin methods in the stochastic space, assuming this also in the spatial domain gives the more compact representation of the problem. Let us take a finite element ansatz $\mathcal{U}_N := \{\varphi_n(x)\}_{n=1}^N \subset \mathcal{U}$ [7,46,51] as a corresponding subspace, such that the solution may be approximated by:

$$u(x, \omega) = \sum_{n=1}^{N} u_n(\omega)\varphi_n(x), \tag{13}$$

where the coefficients $\{u_n(\omega)\}$ are now RVs in $\mathcal{S}$. Inserting the ansatz Eq. (13) back into Eq. (10) and applying the spatial Galerkin conditions [34, 37], we arrive at:

$$\boldsymbol{A}(\omega)[\boldsymbol{u}(\omega)] = \boldsymbol{f}(\omega), \tag{14}$$

where the parameter dependent symmetric and uniformly positive definite matrix $\boldsymbol{A}(\omega)$ is defined similarly to a usual finite element stiffness matrix as $(\boldsymbol{A}(\omega))_{m,n} := a(\omega)(\varphi_m, \varphi_n)$ with the bi-linear form $\mathsf{a}(\omega)$ given by Eq. (11). Furthermore, the right hand side (r.h.s.) is determined by $(\boldsymbol{f}(\omega))_m := \langle \ell(\omega), \varphi_m \rangle$ where the linear form $\ell(\omega)$ is given in Eq. (12), while $\boldsymbol{u}(\omega) = [u_1(\omega), \ldots, u_N(\omega)]^T$ is introduced as a vector of random coefficients as in Eq. (13).

The Eq. (14) represents a linear equation with random r.h.s. and random matrix. It is a semi-discretisation of some sort since it involves the variable ω and is still computationally intractable, as in general we need infinitely many coordinates to parametrise Ω.

2.2 Stochastic Discretisation

The semi-discretised Eq. (14) is approximated such that the stochastic input data $\boldsymbol{A}(\omega)$ and $\boldsymbol{f}(\omega)$ are described with the help of RVs of some known type. Namely, we employ a stochastic Galerkin (SG) method to do the stochastic discretisation of Eq. (14) [1–3, 14, 17, 22, 28, 36, 37, 43, 44, 48, 49]. Basic convergence of such an approximation may be established via Céa's lemma [34, 37].

In order to express the unknown coefficients (RVs) $u_n(\omega)$ in Eq. (13), let us choose as the ansatz functions multivariate *Hermite* polynomials $\{H_\alpha(\boldsymbol{\theta}(\omega))\}_{\alpha \in \mathcal{J}}$

in Gaussian RVs, also known under the name *Wiener's* polynomial chaos expansion (PCE) [17, 27, 34, 36, 37]

$$u_n(\boldsymbol{\theta}) = \sum_{\alpha \in \mathcal{J}} u_n^\alpha H_\alpha(\boldsymbol{\theta}(\omega)), \qquad \text{or} \qquad \boldsymbol{u}(\boldsymbol{\theta}) = \sum_{\alpha \in \mathcal{J}} \boldsymbol{u}^\alpha H_\alpha(\boldsymbol{\theta}(\omega)), \tag{15}$$

where $\boldsymbol{u}^\alpha := [u_1^\alpha, \ldots, u_n^\alpha]^T$. At this point we may forget the original probability space $(\Omega, \mathbb{P})$ and only work with (Θ, Γ) with a Gaussian measure, all RVs from now on to be considered as functions of $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_j, \ldots) \in \Theta$ instead of $\omega \in \Omega$.

The *Cameron-Martin* theorem assures us that the algebra of Gaussian variables is dense in $L_2(\Omega)$. Here the index set $\mathcal{J}$ is taken as a finite subset of $\mathbb{N}_0^{(\mathbb{N})}$, the set of all finite non-negative integer sequences, i.e. multi-indices. Although the set $\mathcal{J}$ is finite with cardinality $|\mathcal{J}| = R$ and $\mathbb{N}_0^{(\mathbb{N})}$ is countable, there is no natural order on it; and hence we do not impose one at this point.

Inserting the ansatz Eq. (15) into Eq. (14) and applying the Bubnov-Galerkin projection onto the finite dimensional subspace $\mathcal{U}_N \otimes \mathcal{S}_\mathcal{J}$, one requires that the weighted residuals vanish:

$$\forall \beta \in \mathcal{J} : \mathbb{E}\left([\boldsymbol{f}(\boldsymbol{\theta}) - \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{u}(\boldsymbol{\theta})] H_\beta(\boldsymbol{\theta})\right) = 0. \tag{16}$$

With $\boldsymbol{f}_\beta := \mathbb{E}\left(\boldsymbol{f}(\boldsymbol{\theta}) H_\beta(\boldsymbol{\theta})\right)$ and $\boldsymbol{A}_{\beta,\alpha} := \mathbb{E}\left(H_\beta(\boldsymbol{\theta}) \boldsymbol{A}(\boldsymbol{\theta}) H_\alpha(\boldsymbol{\theta})\right)$, Eq. (16) reads:

$$\forall \beta \in \mathcal{J} : \sum_{\alpha \in \mathcal{J}} \boldsymbol{A}_{\beta,\alpha} \boldsymbol{u}^\alpha = \boldsymbol{f}_\beta, \tag{17}$$

which further represents a linear, symmetric and positive definite system of equations of size $N \times R$. The system is well-posed in a sense of Hadamard since the Lax-Milgram lemma applies on the subspace $\mathcal{U}_N \otimes \mathcal{S}_\mathcal{J}$.

To expose the structure of and compute the terms in Eq. (17), the parametric matrix in Eq. (14) is expanded in the Karhunen-Loève expansion (KLE) [16, 18, 35, 37] as

$$\boldsymbol{A}(\boldsymbol{\theta}) = \sum_{j=0}^{\infty} \boldsymbol{A}_j \xi_j(\boldsymbol{\theta}) \tag{18}$$

with scalar RVs ξ_j. Together with Eq. (10), it is not too hard to see that $\boldsymbol{A}_j$ can be defined by the bilinear form

$$\mathsf{a}_j(v, u) := \int_\mathcal{G} \nabla v(x) \cdot \left(\kappa_j g_j(x) \nabla u(x)\right) \, \mathrm{d}x, \tag{19}$$

and $(\boldsymbol{A}_j)_{m,n} := \mathsf{a}_j(\varphi_m, \varphi_n)$ with $\kappa_j g_j(x)$ being the coefficient of the KL expansion of $\kappa(x, \omega)$:

$$\kappa(x, \omega) = \kappa_0(x) + \sum_{j=1}^{\infty} \kappa_j g_j(x) \xi_j(\boldsymbol{\theta}),$$

where

$$\xi_j(\boldsymbol{\theta}) = \frac{1}{\kappa_j} |\left(\kappa(\cdot,\omega) - \kappa_0, g_j\right)|_{L_2(\mathcal{G})} = \frac{1}{\kappa_j} \int_{\mathcal{G}} \left(\kappa(x,\omega) - \kappa_0(x)\right) g_j(x) \mathrm{d}x.$$

Now these $\boldsymbol{A}_j$ can be computed as "usual "finite element stiffness matrices with the "material properties "$\kappa_j g_j(x)$. It is worth noting that $\boldsymbol{A}_0$ is just the usual deterministic or mean stiffness matrix, obtained with the mean diffusion coefficient $\kappa_0(x)$ as parameter.

Knowing the polynomial chaos expansion of $\kappa(x,\omega) = \sum_\alpha \kappa^{(\alpha)}(x) H_\alpha(\boldsymbol{\theta})$, where coefficients $\kappa^{(\alpha)}(x)$ can be evaluated as multidimensional integrals over the measure space Θ with standard Gaussian measure:

$$\kappa^{(\alpha)}(x) = \frac{1}{\alpha!} \int_\Theta \kappa(x,\boldsymbol{\theta}) H_\alpha(\boldsymbol{\theta}) \mathbb{P}(\mathrm{d}\boldsymbol{\theta}), \tag{20}$$

compute the polynomial chaos expansion of the ξ_j as

$$\xi_j(\boldsymbol{\theta}) = \sum_{\alpha \in \mathcal{J}} \xi_j^{(\alpha)} H_\alpha(\boldsymbol{\theta}),$$

where

$$\xi_j^{(\alpha)} = \frac{1}{\kappa_j} \int_{\mathcal{G}} \kappa^{(\alpha)}(x) g_j(x) \mathrm{d}x.$$

Later on we, using the PCE coefficients $\kappa^{(\alpha)}(x)$ as well as eigenfunctions $g_j(x)$, compute the following tensor approximation

$$\xi_j^{(\alpha)} \approx \sum_{l=1}^{s} (\xi_l)_j \prod_{k=1}^{\infty} (\xi_{l,k})_{\alpha_k},$$

where $(\xi_l)_j$ means the j-th component in the spatial space and $(\xi_{l,k})_{\alpha_k}$ the α_k-th component in the stochastic space.

The parametric r.h.s. in Eq. (14) has an analogous expansion to Eq. (18), which may be either derived directly from the $\mathbb{R}^N$-valued RV $\boldsymbol{f}(\omega)$—effectively a finite dimensional KLE—or from the continuous KLE of the random linear form in Eq. (12). In either case

$$\boldsymbol{f}(\omega) = \sum_{i=0}^{\infty} \sqrt{\lambda_i} \psi_i(\omega) \boldsymbol{f}_i, \tag{21}$$

where the λ_i are the eigenvalues [23,24,33], and, as in Eq. (18), only a finite number of terms are needed. For sparse representation of KLE see [23,24]. The components in Eq. (17) may now be expressed as $\boldsymbol{f}_\beta = \sum_i \sqrt{\lambda_i} f_\beta^i \boldsymbol{f}_i$ with $f_\beta^i := \mathbb{E}\left(H_\beta \psi_i\right)$. Let us point out that the random variables describing the input to the problem are $\{\xi_j\}$ and $\{\psi_i\}$ (see examples of distribution functions in Fig. 1).

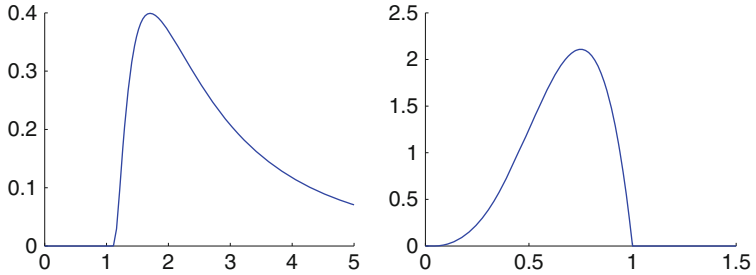


Fig. 1 Shifted lognormal distribution with parameters $\{\mu = 0.5,\ \sigma^2 = 1.0\}$ (on the *left*) and Beta distribution with parameters $\{4, 2\}$ (on the *right*)

Introducing the expansion Eq. (18) into Eq. (17) we obtain:

$$\forall \beta: \quad \sum_{j=0}^{\infty} \sum_{\alpha \in \mathcal{J}} \boldsymbol{\Delta}^{j}_{\boldsymbol{\beta},\boldsymbol{\alpha}} \boldsymbol{A}_j \boldsymbol{u}^{\alpha} = \boldsymbol{f}_{\beta}, \tag{22}$$

where $\boldsymbol{\Delta}^{j}_{\boldsymbol{\beta},\boldsymbol{\alpha}} = \mathbb{E}\left(H_\beta \xi_j H_\alpha\right)$. Denoting the elements of the tensor product space $\mathbb{R}^N \otimes \bigotimes_{\mu=1}^{M} \mathbb{R}^{R_\mu}$, where R_μ is the dimension of the space where ξ_μ is approximated, in an upright bold font, as for example $\mathbf{u}$, and similarly linear operators on that space, as for example $\mathbf{A}$, we may further rewrite Eq. (22) in terms of a tensor products [34, 37]:

$$\mathbf{Au} := \left(\sum_{j=0}^{\infty} \boldsymbol{A}_j \otimes \boldsymbol{\Delta}^{j}\right)\left(\sum_{\alpha \in \mathcal{J}} \boldsymbol{u}^{\alpha} \otimes \boldsymbol{e}^{\alpha}\right) = \left(\sum_{\alpha \in \mathcal{J}} \boldsymbol{f}_{\alpha} \otimes \boldsymbol{e}^{\alpha}\right) =: \mathbf{f}, \tag{23}$$

where $\boldsymbol{e}^{\alpha}$ denotes the canonical basis in $\bigotimes_{\mu=1}^{M} \mathbb{R}^{R_\mu}$. With the help of Eq. (21) and the relations directly following it, the r.h.s. in Eq. (23) may be rewritten as

$$\mathbf{f} = \sum_{\alpha \in \mathcal{J}} \sum_{i=0}^{\infty} \sqrt{\lambda_i}\, f^{i}_{\alpha}\, \boldsymbol{f}_i \otimes \boldsymbol{e}^{\alpha} = \sum_{i=0}^{\infty} \sqrt{\lambda_i}\, \boldsymbol{f}_i \otimes \boldsymbol{g}_i, \tag{24}$$

where $\boldsymbol{g}_i := \sum_{\alpha \in \mathcal{J}} f^{i}_{\alpha} \boldsymbol{e}^{\alpha}$. Later on, splitting $\boldsymbol{g}_i$ further (this result will be published soon), obtain

$$\mathbf{f} \approx \sum_{k=1}^{R} \tilde{\boldsymbol{f}}_k \otimes \bigotimes_{\mu=1}^{M} \boldsymbol{g}_{k\mu}. \tag{25}$$

The similar splitting work, but in application in another context was done in [5, 10–13]. Now the tensor product structure is exhibited also for the fully discrete counterpart to Eq. (10), and not only for the solution $\mathbf{u}$ and r.h.s. $\mathbf{f}$, but also for the operator or matrix $\mathbf{A}$.

The operator $\mathbf{A}$ in Eq. (23) inherits the properties of the operator in Eq. (10) in the sense of symmetry and positive definiteness [34, 37]. The symmetry may be verified

directly from Eq. (17), while the positive definiteness follows from the Galerkin projection and the uniform convergence in Eq. (23) on the finite dimensional space $\mathbb{R}^{(N\times N)}\otimes\bigotimes_{\mu=1}^{M}\mathbb{R}^{(R_\mu\times R_\mu)}$. In order to make the procedure computationally feasible, of course the infinite sum in Eq. (18) has to be truncated at a finite value, say at M. The choice of M is now part of the stochastic discretisation and not an assumption.

Due to the uniform convergence alluded to above the sum can be extended far enough such that the operators $\mathbf{A}$ in Eq. (23) are uniformly positive definite with respect to the discretisation parameters [34, 37]. This is in some way analogous to the use of numerical integration in the usual FEM [7, 46, 51].
The Eq. (23) is solved by iterative methods in the low-rank canonical tensor format in [38]. The corresponding matlab code is implemented in [50]. Additional interesting result in [38] is the research of different strategies for the tensor-rank truncation after each iteration. Other works devoted to the research of properties of the system matrix in Eq. (26), developing of Kronecker product preconditioning and to the iterative methods to solve system in Eq. (26) are in [8, 9, 47].

Applying further splitting to $\mathbf{\Delta}^j$ (this result will be published soon), the fully discrete forward problem may finally be announced as

$$\mathbf{Au} = \left(\sum_{l=1}^{s}\tilde{\boldsymbol{A}}_l\otimes\bigotimes_{\mu=1}^{M}\mathbf{\Delta}_{l\mu}\right)\left(\sum_{j=1}^{r}\boldsymbol{u}_j\otimes\bigotimes_{\mu=1}^{M}\boldsymbol{u}_{j\mu}\right)=\sum_{k=1}^{R}\tilde{\boldsymbol{f}}_k\otimes\bigotimes_{\mu=1}^{M}\boldsymbol{g}_{k\mu}=\mathbf{f}, \tag{26}$$

where $\tilde{\boldsymbol{A}}_l\in\mathbb{R}^{N\times N}$, $\mathbf{\Delta}_{l\mu}\in\mathbb{R}^{R_\mu\times R_\mu}$, $\boldsymbol{u}_j\in\mathbb{R}^N$, $\boldsymbol{u}_{j\mu}\in\mathbb{R}^{R_\mu}$, $\tilde{\boldsymbol{f}}_k\in\mathbb{R}^N$ and $\boldsymbol{g}_{k\mu}\in\mathbb{R}^{R_\mu}$. The similar splitting work, but in application in another context was done in [5, 10–13].

2.3 Quadrature Rules and Sparse Integration Grids

Sparse grids are usually used together with collocation method [39] or for computing PCE coefficents of random fields $\kappa(x,\omega)$, $f(x,\omega)$ and $u(x,\omega)$ (the last are used, for instance, for building the response surface of the solution [29, 30]). Since the number of deterministic code simulations in a stochastic PDE framework can be very large (e.g., collocation, MC, quasi-MC methods) the very efficient sparse grid methods are extremely important. With the usage of sparse grids the number of expansive simulations can be drastically reduced [29, 30].

The integral in Eq. (20) is of the following type

$$\Psi(x)=\mathbb{E}\left(\Psi(x,\omega)\right)=\int_{\Theta}\Psi(x,\boldsymbol{\theta})\,\Gamma(\mathrm{d}\boldsymbol{\theta}). \tag{27}$$

Such an integral may numerically be approximated by a weighted sum of samples of the integrand

$$\Psi(x)\approx\mathbf{\Psi}_Z=\sum_{z=1}^{Z}w_z\Psi(x,\boldsymbol{\theta}_z)=\sum_{z=1}^{Z}w_z\Psi_Z(\boldsymbol{\theta}_z), \tag{28}$$

where evaluation points are $\boldsymbol{\theta}_z \in \Theta$, and w_z are the weights. The textbook approach to an integral like Eq. (28) would be to take a good one-dimensional quadrature rule, and to iterate it in every dimension; this is the full tensor product approach.

Assume that we use one-dimensional Gauss-Hermite-formulas Q_k with $k \in \mathbb{N}$ integration points $\boldsymbol{\theta}_{j,k}$ and weights $w_{j,k}$, $j = 1, \ldots, k$. As is well-known, they integrate polynomials of degree less than $2k$ exactly, and yield an error of order $O(k^{-(2r-1)})$ for r-times continuously differentiable integrands, hence takes smoothness into full account.

If we take a tensor product of these rules by iterating them M times, we have

$$\boldsymbol{\Psi}_Z = Q_k^M(\Psi) := (Q_k \otimes \cdots \otimes Q_k)(\Psi) = \bigotimes_{j=1}^{M} Q_k(\Psi)$$

$$= \sum_{j_1=1}^{k} \cdots \sum_{j_M=1}^{k} w_{j_1,k} \cdots w_{j_M,k} \Psi_Z(\boldsymbol{\theta}_{j_1,k}, \ldots, \boldsymbol{\theta}_{j_M,k}).$$

This "full" tensor quadrature evaluates the integrand on a regular mesh of $Z = k^M$ points, and the approximation-error has order $\mathcal{O}(Z^{-(2r-1)/M})$. Due to the exponential growth of the number of evaluation points and hence the effort with increasing dimension, the application of full tensor quadrature is impractical for high stochastic dimensions, this has been termed the "curse of dimensions" [40].

Sparse grid, hyperbolic cross, or Smolyak quadrature [45] can be applied in much higher dimensions—for some recent work see e.g. [15, 39–42] and the references therein. A software package is available at [25, 26].

Like full tensor quadrature, a Smolyak quadrature formula is constructed from tensor products of one-dimensional quadrature formulas, but it combines quadrature formulas of high order in only some dimensions with formulas of lower order in the other dimensions. For a multi-index $\eta \in \mathbb{N}^M$ the Smolyak quadrature formula is

$$\boldsymbol{\Psi}_Z = S_k^M(\Psi) := \sum_{k \le |\eta| \le k+M-1} (-1)^{k+M-1-|\eta|} \binom{k-1}{|\eta|-k} \bigotimes_{j=1}^{M} Q_{\eta_j}(\Psi).$$

For a fixed k the number of evaluations grows significantly slower in the number of dimensions than for full quadrature. The price is a larger error: full quadrature integrates monomials $\boldsymbol{\theta}^\eta = \theta_1^{\eta_1} \cdots \theta_M^{\eta_M}$ exactly if their partial degree $\max_j \eta_j$ does not exceed $2k-1$. Smolyak formulas S_k^M integrate multivariate polynomials exactly only if their total polynomial degree $|\eta|$ is at most $2k-1$. But still the error is only $O(Z^{-r}(\log Z)^{(M-1)(r+1)})$ with $Z = O\left(\left[2^k/(k!)\right] M^k\right)$ evaluation points for a r-times differentiable function [33]. This has been used up to several hundred dimensions.

It may be seen that smoothness of the function is taken into account very differently by the various integration methods, as this shows up in the asymptotic error behaviour. The "roughest" is Monte Carlo, it only feels the variance, quasi

Monte Carlo feels the function's variation, and the Smolyak rules actually take smoothness into account.

In [31, 32, 37] some numerical experiments are shown. The finding there is that for low M normal quadrature is best. For higher to moderately high (several hundred) M, sparse or Smolyak quadrature [15,25,26,42,45] is advisable. For very high dimensions, we come into the region where first Quasi Monte Carlo [4], and then finally for extremely high dimension Monte Carlo methods should be most effective.

3 The Canonical Tensor Format

Let $\mathcal{T} := \bigotimes_{\mu=1}^{d} \mathbb{R}^{n_\mu}$ be the tensor space constructed from $(\mathbb{R}^{n_\mu}, \langle , \rangle_{\mathbb{R}^{n_\mu}})$ $(d \geq 3)$. From a mathematical point of view, a tensor representation U is a multilinear map from a parameter space P onto $\mathcal{T}$, i.e. $U : P \to \mathcal{T}$. The parameter space $P = \times_{\nu=1}^{D} P_\nu$ $(d \leq D)$ is the Cartesian product of tensor spaces P_ν, where in general the order of every P_ν is (much) smaller then d. Further, P_ν depends on some representation rank parameter $r_\nu \in \mathbb{N}$. A standard example of a tensor representation is the canonical tensor format.

Definition 1 (r-Terms, Tensor Rank, Canonical Tensor Format, Elementary Tensor, Representation System). The set $\mathcal{R}_r$ of tensors which can be represented in $\mathcal{T}$ with *r-terms* is defined as

$$\mathcal{R}_r(\mathcal{T}) := \mathcal{R}_r := \left\{ \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} v_{i\mu} \in \mathcal{T} : v_{i\mu} \in \mathbb{R}^{n_\mu} \right\}. \tag{29}$$

Let $v \in \mathcal{T}$. The *tensor rank of v in* $\mathcal{T}$ is

$$\operatorname{rank}(v) := \min \{ r \in \mathbb{N}_0 : v \in \mathcal{R}_r \}. \tag{30}$$

The *canonical tensor format in* $\mathcal{T}$ for variable r is defined by the mapping

$$U_{cp} : \times_{\mu=1}^{d} \mathbb{R}^{n_\mu \times r} \to \mathcal{R}_r, \tag{31}$$

$$\hat{v} := (v_{i\mu} : 1 \leq i \leq r,\ 1 \leq \mu \leq d) \mapsto U_{cp}(\hat{v}) := \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} v_{i\mu}.$$

We call the sum of elementary tensors $v = \sum_{i=1}^{r} \otimes_{\mu=1}^{d} v_{i\mu} \in \mathcal{R}_r$ a tensor represented in the canonical tensor format with r terms, where an *elementary tensor* is of the form $\bigotimes_{\mu=1}^{d} v_\mu \in \mathcal{R}_1$, $v_\mu \in V_\mu$. The system of vectors $\left(v_{i\mu} : 1 \leq i \leq r,\ 1 \leq \mu \leq d\right)$ is a *representation system* of v with *representation rank* r.

Note that the representation rank refers to the representation system $(v_{i\mu} : 1 \leq i \leq r,\ 1 \leq \mu \leq d)$, not to the represented tensor. In our applications we work

only with tensors represented in a tensor format. A tensor $u \in \mathcal{R}_r \subset \mathcal{T}$ with $\prod_{\mu=1}^{d} n_\mu$ entities is represented on a computer system with a representation system $\hat{u} = \left(u_{i\mu} \in \mathbb{R}^{n_\mu} : 1 \le i \le r,\ 1 \le \mu \le d\right)$ and the use of U_{cp}, i.e. $u = U_{cp}(\hat{u})$. The memory requirement for the representation system $\hat{u}$ is only $r \sum_{\mu=1}^{d} n_\mu$. Later we will see that the efficient data representation in tensor formats has several benefits for the data analysis in high dimensions. For the data analysis we need operations described in Lemma 1.

Lemma 1. *Let $r_1, r_2 \in \mathbb{N}$, $u \in \mathcal{R}_{r_1}$ and $v \in \mathcal{R}_{r_2}$. We have*

(i) $\langle u, v\rangle_{\mathcal{T}} = \sum_{j_1=1}^{r_1} \sum_{j_2=1}^{r_2} \prod_{\mu=1}^{d} \langle u_{j_1\mu}, v_{j_2\mu}\rangle_{\mathbb{R}^{n_\mu}}$. *The computational cost of* $\langle u, v\rangle_{\mathcal{T}}$ *is* $\mathcal{O}\left(r_1 r_2 \sum_{\mu=1}^{d} n_\mu\right)$.

(ii) $u + v \in \mathcal{R}_{r_1+r_2}$.

(iii) $u \odot v \in \mathcal{R}_{r_1 r_2}$, *where* $\odot$ *denotes the point wise Hadamard product. Further,* $u \odot v$ *can be computed in the canonical tensor format with* $r_1 r_2 \sum_{\mu=1}^{d} n_\mu$ *arithmetic operations.*

Proof. (i) and (ii) are trivial. For (iii), let $u = \sum_{j_1=1}^{r_1} \bigotimes_{\mu=1}^{d} u_{j_1\mu}$ and $v = \sum_{j_2=1}^{r_2} \bigotimes_{\mu=1}^{d} v_{j_2\mu}$. We have

$$u \odot v = \sum_{j_1=1}^{r_1} \sum_{j_2=1}^{r_2} \left[\bigotimes_{\mu=1}^{d} u_{j_1\mu}\right] \odot \left[\bigotimes_{\mu=1}^{d} v_{j_2\mu}\right] = \sum_{j_1=1}^{r_1} \sum_{j_2=1}^{r_2} \bigotimes_{\mu=1}^{d} \left[u_{j_1\mu} \odot_{n_\mu} v_{j_2\mu}\right],$$

where $\odot_{n_\mu}$ denotes the Hadamard product in $\mathbb{R}^{n_\mu}$. Obviously, we need $r_1 r_2 \sum_{\mu=1}^{d} n_\mu$ operations to determine a representation system of $u \odot v$.

Later we will use operations like the Hadamard product and the addition of tensors in the canonical format in iterative procedures. From Lemma 1 it follows that the numerical cost grows only linear respect to the order d and the representation rank of the resulting tensors will increase. The last fact makes our iterative process not feasible. Therefore, we need an approximation method which approximates a given tensor represented in the canonical format with lower rank tensors up to a given accuracy.

Definition 2 (Approximation Problem). For given $v \in \mathcal{R}_R$ and $\varepsilon > 0$ we are looking for minimal $r_\varepsilon \le R$ and $\hat{x}^* \in \times_{\mu=1}^{d} \mathbb{R}_{\mu}^{n_\mu \times r_\varepsilon}$ such that:

(i) $\|v - U_{cp}(\hat{x}^*)\| \le \varepsilon \|v\|$,

(ii) $\|v - U_{cp}(\hat{x}^*)\| = \operatorname{dist}(v, \mathcal{R}_{r_\varepsilon}) = \min_{\hat{x} \in \times_{\mu=1}^{d} \mathbb{R}^{n_\mu \times r}} \|v - U_{cp}(\hat{x})\|$, where $\hat{x} \in \times_{\mu=1}^{d} \mathbb{R}^{n_\mu \times r}$ is bounded.

The solution of this problem was already discussed in [10, 12, 13]. In the following we will denote a solution of the approximation problem from Definition 2 with $\mathfrak{App}_\varepsilon(v)$.

Note 1. Let $v \in \mathcal{R}_R$, $\varepsilon > 0$ and $U_{cp}(\hat{x}^*)$ a solution of the approximation problem as analysed in [10, 12, 13]. During the article, $U_{cp}(\hat{x}^*)$ is denote by

$$\mathfrak{App}_\varepsilon(v) := U_{cp}(\hat{x}^*). \tag{32}$$

4 Analysis of High Dimensional Data

In the following section let $\mathcal{I} = \times_{\mu=1}^{d} \mathcal{I}_\mu$, where $\mathcal{I}_\mu = \{i \in \mathbb{N} : 1 \le i \le n_\mu\}$. For the analysis of tensor structured data in high dimensions, the focus of attention is a problem depended recursively defined sequence $(u_k)_{k\in\mathbb{N}_{\ge 0}}$ represented in the canonical tensor format, i.e. we have a map $\Phi_P : \mathcal{T} \to \mathcal{T}$ such that

$$u_k := \Phi_P(u_{k-1}), \tag{33}$$

where $u_0 \in \mathcal{R}_{r_0}$ is given. The map Φ_P is constructed with the help of addition, scalar and pointwise Hadamard multiplications of tensors represented in the canonical tensor format. According to Lemma 1, the representation rank of u_k from Eq. (33) will increase. Therefore, we have to compute lower representation ranks approximations and continue the iterative process. This results in the following general inexact iteration scheme:

$$\begin{aligned} z_k &:= \Phi_P(u_{k-1}), \\ u_k &:= \mathfrak{App}_{\varepsilon_k}(z_k), \end{aligned} \tag{34}$$

Where the convergence of such inexact iterations is analysed in [21].

4.1 Computation of the Maximum Norm and Corresponding Index

We describe a approach for computing the maximum norm of $u = \sum_{j=1}^{r} \bigotimes_{\mu=1}^{d} u_{j\mu} \in \mathcal{R}_r$,

$$\|u\|_\infty := \max_{\underline{i}:=(i_1,\dots,i_d)\in\mathcal{I}} |u_{\underline{i}}| = \max_{\underline{i}:=(i_1,\dots,i_d)\in\mathcal{I}} \left| \sum_{j=1}^{r} \prod_{\mu=1}^{d} \left(u_{j\mu}\right)_{i_\mu} \right|, \tag{35}$$

and the corresponding multi-index. Since the cardinality of $\mathcal{I}$ grows exponential with d, $\#\mathcal{I} = \prod_{\mu=1}^{d} n_\mu$, the known methods are already inefficient for small values of n_μ and d. To build an efficient algorithm we use the special tensor structure of u and show that computing $\|u\|_\infty$ is equivalent to a very simple tensor structured eigenvalue problem. Let $\underline{i}^* := (i_1^*, \dots, i_d^*) \in \mathcal{I}$ be the index with

$$\|u\|_\infty = |u_{\underline{i}^*}| = \left| \sum_{j=1}^{r} \prod_{\mu=1}^{d} \left(u_{j\mu}\right)_{i_\mu^*} \right| \text{ and } e^{(\underline{i}^*)} := \bigotimes_{\mu=1}^{d} e_{i_\mu^*},$$

where $e_{i_\mu^*} \in \mathbb{R}^{n_\mu}$ the i_μ^*-th canonical vector in $\mathbb{R}^{n_\mu}$ ($\mu \in \mathbb{N}_{\leq d}$). Then for the pointwise Hadamard product of $u \odot e^{(\underline{i}^*)}$ have

$$\begin{aligned} u \odot e^{(\underline{i}^*)} &= \left[\sum_{j=1}^{r} \bigotimes_{\mu=1}^{d} u_{j\mu} \right] \odot \left[\bigotimes_{\mu=1}^{d} e_{i_\mu^*} \right] = \sum_{j=1}^{r} \bigotimes_{\mu=1}^{d} u_{j\mu} \odot e_{i_\mu^*} \\ &= \sum_{j=1}^{r} \bigotimes_{\mu=1}^{d} \left[(u_{j\mu})_{i_\mu^*} \, e_{i_\mu^*} \right] = \underbrace{\left[\sum_{j=1}^{r} \prod_{\mu=1}^{d} (u_{j\mu})_{i_\mu^*} \right]}_{u_{\underline{i}^*} =} \bigotimes_{\mu=1}^{d} e_{(i_\mu^*)}, \end{aligned}$$

from which follows

$$u \odot e^{(\underline{i}^*)} = u_{\underline{i}^*} e^{(\underline{i}^*)}. \tag{36}$$

Equation (36) is an eigenvalue problem. By defining the following diagonal matrix

$$D(u) := \sum_{j=1}^{r} \bigotimes_{\mu=1}^{d} \operatorname{diag}\left((u_{j\mu})_{l_\mu} \right)_{l_\mu \in \mathbb{N}_{\leq n_\mu}} \tag{37}$$

with representation rank r, obtain $D(u)v = u \odot v$ for all $v \in \mathcal{T}$:

Corollary 1. *Let u, $\underline{i}^*$ and $D(u)$ are defined as described above. Then elements of u are the eigenvalues of $D(u)$ and all eigenvectors $e^{(\underline{i})}$ are of the following form:*

$$e^{(\underline{i})} = \bigotimes_{\mu=1}^{d} e_{i_\mu}, \tag{38}$$

where $\underline{i} := (i_1, \ldots, i_d) \in \mathrm{i}$ is the index of $u_{\underline{i}}$. Therefore $\|u\|_\infty$ is the largest eigenvalue of $D(u)$ with the corresponding eigenvector $e^{(\underline{i}^*)}$.

There are different methods for the computation of the largest eigenvalue and corresponding eigenvector [19]. In this example, we simple use the power iteration to solve the eigenvalue problem. Since the tensor rank of z_k grows up monotonically, the power method described in Algorithm 4 is modified accordingly to Eq. (34). Accordingly to [20] there are

$$\mathcal{O}\left(\frac{d \log \max\{n_1, \ldots, n_d\} - \log \varepsilon}{\varepsilon} \right). \tag{39}$$

Algorithm 4 Computing the maximum norm of $u \in \mathcal{R}_r$ by vector iteration

1: Choose $y_0 := \bigotimes_{\mu=1}^{d} \frac{1}{n_\mu} \underline{1}_\mu$, where $\underline{1}_\mu := (1, \ldots, 1)^T \in \mathbb{R}^{n_\mu}$, $k_{\max} \in \mathbb{N}$, and take $\varepsilon := 1 \times 10^{-7}$.
2: **for** $k = 1, 2, \ldots, k_{\max}$ **do**
3:

$$q_k = u \odot y_{k-1}, \quad \lambda_k = \langle y_{k-1}, q_k \rangle, \quad z_k = q_k / \sqrt{\langle q_k, q_k \rangle},$$
$$y_k = \mathfrak{Approx}_\varepsilon(z_k).$$

4: **end for**

iteration steps necessary to compute the maximum norm of u up to the relative error $\varepsilon \in \mathbb{R}_{>0}$. To guaranty convergence one takes the initial guess y_0 as

$$y_0 := \sum_{l_1=1}^{n_1} \cdots \sum_{l_d=1}^{n_d} \bigotimes_{\mu=1}^{d} \frac{1}{n_\mu} e_{l_\mu} = \bigotimes_{\mu=1}^{d} \left(\frac{1}{n_\mu} \sum_{l_\mu=1}^{n} e_{l_\mu} \right) = \bigotimes_{\mu=1}^{d} \frac{1}{n_\mu} \underline{1}_\mu. \tag{40}$$

We recall that the presented method is only an approximate method to compute $\|u\|_\infty$ and $e^{(\underline{i}^*)}$. In general the vector iteration is not appropriate for solving eigenvalue problems. A possible improvement is the inverse vector iteration method, which is applied on a spectrum shift of u. Therefore is computing of the pointwise inverse necessary. Many other well-known methods require orthogonalisation, which seems for sums of elementary tensors not practicable.

4.2 Computation of the Characteristic

The key object of the following approaches is a tensor which we call characteristic of $u \in \mathcal{T}$ in $I \subset \mathbb{R}$.

Definition 3 (Characteristic, Sign). The *characteristic* $\chi_I(u) \in \mathcal{T}$ of $u \in \mathcal{T}$ in $I \subset \mathbb{R}$ is for every multi-index $\underline{i} \in \mathcal{I}$ pointwise defined as

$$(\chi_I(u))_{\underline{i}} := \begin{cases} 1, & u_{\underline{i}} \in I; \\ 0, & u_{\underline{i}} \notin I. \end{cases} \tag{41}$$

Furthermore, the $\operatorname{sign}(u) \in \mathcal{T}$ is for all $\underline{i} \in \mathcal{I}$ pointwise defined by

$$(\operatorname{sign}(u))_{\underline{i}} := \begin{cases} 1, & u_{\underline{i}} > 0; \\ -1, & u_{\underline{i}} < 0; \\ 0, & u_{\underline{i}} = 0. \end{cases} \tag{42}$$

Similar to the computation of the maximum norm, the computational cost of standard methods grows exponential with d, since we have to visit $\prod_{\mu=1}^{d} n_\mu$ entries of u. If u is represented in the canonical tensor format with r terms, i.e. $u = \sum_{j=1}^{r} \bigotimes_{\mu=1}^{d} u_{j\mu}$, there is a possibility to compute the characteristic $\chi_I(u)$ since there are methods to compute the sign(u).

Lemma 2. *Let $u \in \mathcal{T}$, $a, b \in \mathbb{R}$, and $\mathbb{1} = \bigotimes_{\mu=1}^{d} \underline{1}_\mu$, where $\underline{1}_\mu := (1, \ldots, 1)^t \in \mathbb{R}^{n_\mu}$.*

(i) If $I = \mathbb{R}_{<b}$, then we have $\chi_I(u) = \frac{1}{2}(\mathbb{1} + sign(b\mathbb{1} - u))$.
(ii) If $I = \mathbb{R}_{>a}$, then we have $\chi_I(u) = \frac{1}{2}(\mathbb{1} - sign(a\mathbb{1} - u))$.
(iii) If $I = (a, b)$, then we have $\chi_I(u) = \frac{1}{2}(sign(b\mathbb{1} - u) - sign(a\mathbb{1} - u))$.

Proof. Let $\underline{i} \in \mathcal{I}$. (i) If $u_{\underline{i}} < b \Rightarrow 0 < b - u_{\underline{i}} \Rightarrow \operatorname{sign}(b - u_{\underline{i}}) = 1 \Rightarrow \frac{1}{2}(1 + \operatorname{sign}(b - u_{\underline{i}})) = 1 = (\chi_I(u))_{\underline{i}}$. If $u_{\underline{i}} > b \Rightarrow b - u_{\underline{i}} < 0 \Rightarrow \operatorname{sign}(b - u_{\underline{i}}) = -1 \Rightarrow \frac{1}{2}(1 + \operatorname{sign}(b - u_{\underline{i}})) = 0 = (\chi_I(u))_{\underline{i}}$.
(ii) Analog to (i). (iii) Follows from (i) and (ii).

In the following part we analyse bounds for the representation rank of the characteristic $\chi_I(u)$.

Definition 4 (Cartesian Index Set, Cartesian Covering). Let $M \subset \mathcal{I}$ be a subset of multi-indices. We call M a *Cartesian index set* if there exist $M_\mu \subset \mathcal{I}_\mu$ such that $M = \times_{\mu=1}^{d} M_\mu$. We call a set $\operatorname{ccov}(M) = \{U \subset \mathcal{I} : U \text{ is Cartesian}\}$ a *Cartesian covering* of M if

$$M = \dot{\bigcup}_{U \in \operatorname{ccov}(M)} U,$$

where the symbol $\dot{\bigcup}$ stands for disjoint union.

Note that for every set $M \subseteq \mathcal{I}$ there exist a Cartesian covering.

Lemma 3. *Let $I \subseteq \mathbb{R}$, $u \in \mathcal{T}$, and $M := \operatorname{supp} \chi_I(u)$. We have*

$$rank(\chi_I(u)) \leq \min\{m_1, m_2 + 1\}, \tag{43}$$

where $m_1 := \min\{\#C_1 \in \mathbb{N} : C_1$ is a Cartesian covering of $M\}$ and $m_2 := \min\{\#C_2 \in \mathbb{N} : C_2$ is a Cartesian covering of $M^c := \mathcal{I} \setminus M\}$.

Proof. Let $\{M_l = \times_{\mu=1}^{d} M_{l,\mu} : 1 \leq l \leq m_1\}$ a Cartesian covering of M and $\{N_l = \times_{\mu=1}^{d} N_{l,\mu} : 1 \leq l \leq m_2\}$ a Cartesian covering of M^c. We have

$$\chi_I(u) = \sum_{\underline{i} \in M} \bigotimes_{\mu=1}^{d} e_{i_\mu} = \sum_{l=1}^{m_1} \sum_{i_1 \in M_{l,1}} \cdots \sum_{i_d \in M_{l,d}} \bigotimes_{\mu=1}^{d} e_{i_\mu}$$

$$= \sum_{l=1}^{m_1} \bigotimes_{\mu=1}^{d} \left[\sum_{i_\mu \in M_{l,\mu}} e_{i_\mu} \right] \Rightarrow \operatorname{rank}(\chi_I(u)) \leq m_1,$$

where $e_{i_\mu} \in \mathbb{R}^{n_\mu}$ is the i_μ-th canonical vector in $\mathbb{R}^{n_\mu}$. Further, we have

$$\chi_I(u) = \mathbb{1} - \sum_{\underline{i} \in M^c} \bigotimes_{\mu=1}^{d} e_{i_\mu} = \mathbb{1} - \sum_{l=1}^{m_2} \sum_{i_1 \in N_{l,1}} \cdots \sum_{i_d \in N_{l,d}} \bigotimes_{\mu=1}^{d} e_{i_\mu}$$

$$= \mathbb{1} - \sum_{l=1}^{m_2} \bigotimes_{\mu=1}^{d} \left[\sum_{i_\mu \in N_{l,\mu}} e_{i_\mu} \right] \Rightarrow \operatorname{rank}(\chi_I(u)) \leq m_2 + 1.$$

The most widely used and analysed method for computing the sign function $\operatorname{sign}(A)$ of a matrix A is the Newton iteration,

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}), \quad X_0 = A. \tag{44}$$

The connection of the iteration with the sign function is not immediately obvious. The iteration can be derived by applying the Newton's method to the equation $X^2 = \boldsymbol{I}$. It is also well known that the convergence of the Newton iteration is quadratically, i.e. we have

$$\|X_{k+1} - \operatorname{sign}(A)\| \leq \frac{1}{2}\|X_k^{-1}\|\|X_k - \operatorname{sign}(A)\|^2.$$

The Newton iteration is one of the seldom circumstances in numerical analysis where the explicit computation of the inverse is required. One way to try to remove the inverse in Eq. (44) is to approximate it by one step of the Newton's method for the inverse, which has the form $Y_{k+1} = Y_k(2\boldsymbol{I} - BY_k)$ for computing B^{-1}. This leads to the Newton-Schulz iteration adapted to our tensor setting

$$u_{k+1} = \frac{1}{2} u_k \odot (3\mathbb{1} - u_k \odot u_k), \quad u_0 := u. \tag{45}$$

It is known that the Newton-Schulz iteration retains the quadratic convergence of the Newton's method. However, it is only locally convergent, with convergence guaranteed for $\|\mathbb{1} - u_0 \odot u_0\| < 1$ in some suitable norm. According to Eq. (34) the inexact Newton-Schulz iteration in tensor formats is described by Algorithm 5, where the computation of the pointwise inverse is described in Sect. 4.4.

4.3 Computation of Level Sets, Frequency, Mean Value, and Variance

For the computation of cumulative distribution functions it is important to compute level sets of a given tensor $u \in \mathcal{T}$.

Definition 5 (Level Set, Frequency). Let $I \subset \mathbb{R}$ and $u \in \mathcal{T}$. The *level set* $\mathcal{L}_I(u) \in \mathcal{T}$ of u respect to I is point wise defined by

Algorithm 5 Computing $\operatorname{sign}(u)$, $u \in \mathcal{R}_r$ (Hybrid Newton-Schulz Iteration)

Choose $u_0 := u$ and $\varepsilon \in \mathbb{R}_+$.
while $\|\mathbb{1} - u_{k-1} \odot u_{k-1}\| < \varepsilon \|u\|$ **do**
  if $\|\mathbb{1} - u_{k-1} \odot u_{k-1}\| < \|u\|$ **then**
    $z_k := \frac{1}{2} u_{k-1} \odot (3\mathbb{1} - u_{k-1} \odot u_{k-1})$
  else
    $z_k := \frac{1}{2}(u_{k-1} + u_{k-1}^{-1})$
  end if
  $u_k := \mathfrak{App}_{\varepsilon_k}(z_k)$
end while

$$(\mathcal{L}_I(u))_{\underline{i}} := \begin{cases} u_{\underline{i}}, u_{\underline{i}} \in I \ ; \\ 0, u_{\underline{i}} \notin I \ , \end{cases} \tag{46}$$

for all $\underline{i} \in \mathcal{I}$. The *frequency* $\mathcal{F}_I(u) \in \mathbb{N}$ of u respect to I is defined as

$$\mathcal{F}_I(u) := \#\operatorname{supp} \chi_I(u), \tag{47}$$

where $\chi_I(u)$ is the characteristic of u in I, see Definition 3.

Proposition 1. *Let $I \subset \mathbb{R}$, $u \in \mathcal{T}$, and $\chi_I(u)$ its characteristic. We have*

$$\mathcal{L}_I(u) = \chi_I(u) \odot u \tag{48}$$

and $\operatorname{rank}(\mathcal{L}_I(u)) \le \operatorname{rank}(\chi_I(u))\operatorname{rank}(u)$. Furthermore, the frequency $\mathcal{F}_I(u) \in \mathbb{N}$ of u respect to I can by computed by

$$\mathcal{F}_I(u) = \langle \chi_I(u), \mathbb{1} \rangle , \tag{49}$$

where $\mathbb{1} = \bigotimes_{\mu=1}^{d} \tilde{1}_\mu$, $\tilde{1}_\mu := (1, \dots, 1)^T \in \mathbb{R}^{n_\mu}$.

Proposition 2. *Let $u = \sum_{j=1}^{r} \bigotimes_{\mu=1}^{d} u_{j\mu} \in \mathcal{R}_r$, then the mean value $\overline{u}$ can be computed as a scalar product*

$$\overline{u} = \left\langle \left(\sum_{j=1}^{r} \bigotimes_{\mu=1}^{d} u_{j\mu} \right), \left(\bigotimes_{\mu=1}^{d} \frac{1}{n_\mu} \tilde{1}_\mu \right) \right\rangle = \sum_{j=1}^{r} \bigotimes_{\mu=1}^{d} \frac{\langle u_{j\mu}, \tilde{1}_\mu \rangle}{n_\mu} = \sum_{j=1}^{r} \prod_{\mu=1}^{d} \frac{1}{n_\mu} \left(\sum_{k=1}^{n_\mu} u_{j\mu} \right), \tag{50}$$

where $\tilde{1}_\mu := (1, \dots, 1)^T \in \mathbb{R}^{n_\mu}$. According to Lemma 1, the numerical cost is $\mathcal{O}\left(r \cdot \sum_{\mu=1}^{d} n_\mu\right)$.

Proposition 3. *Let $u \in \mathcal{R}_r$ and*

$$\tilde{u} := u - \overline{u} \bigotimes_{\mu=1}^{d} \frac{1}{n_\mu} \underline{1} = \sum_{j=1}^{r+1} \bigotimes_{\mu=1}^{d} \tilde{u}_{j\mu} \in \mathcal{R}_{r+1}, \tag{51}$$

then the variance var(u) of u can be computed as follows

$$\begin{aligned} var(u) &= \frac{1}{\prod_{\mu=1}^{d} n_\mu} \langle \tilde{u}, \tilde{u} \rangle = \frac{1}{\prod_{\mu=1}^{d} n_\mu} \left\langle \left(\sum_{i=1}^{r+1} \bigotimes_{\mu=1}^{d} \tilde{u}_{i\mu} \right), \left(\sum_{j=1}^{r+1} \bigotimes_{\nu=1}^{d} \tilde{u}_{j\nu} \right) \right\rangle \\ &= \sum_{i=1}^{r+1} \sum_{j=1}^{r+1} \prod_{\mu=1}^{d} \frac{1}{n_\mu} \left\langle \tilde{u}_{i\mu}, \tilde{u}_{j\mu} \right\rangle . \end{aligned}$$

According to Lemma 1, the numerical cost is $\mathcal{O}\left((r+1)^2 \cdot \sum_{\mu=1}^{d} n_\mu\right)$.

4.4 Computation of the Pointwise Inverse

Computing the pointwise inverse u^{-1} is of interest, e.g. by improved computation of the maximum norm and by iterative computations of sign(u) or $\sqrt{u}$. Let us further assume that $u_{\underline{i}} \neq 0$ for all $\underline{i} \in \mathcal{I}$. The mapping $\Phi_k : \mathcal{T} \to \mathcal{T}$ from Eq. (34) is defined as follows:

$$x \mapsto \Phi(x)_{u^{-1}} := x \odot (2\mathbb{1} - u \odot x). \tag{52}$$

This recursion is motivated through application of the Newton method on the function $f(x) := u - x^{-1}$, see [21]. After defining the error by $e_k := \mathbb{1} - u \odot x_k$, we obtain

$$e_k = \mathbb{1} - ux_k = \mathbb{1} - ux_{k-1}\,(\mathbb{1} + e_{k-1}) = e_{k-1} - ux_{k-1}e_{k-1} = (\mathbb{1} - ux_{k-1})\,e_{k-1} = e_0^{2^k}$$

and $(x_k)_{k\in\mathbb{N}}$ converges quadratically for $\|e_0\| < 1$. Then for e_k have

$$u^{-1} - x_k = u^{-1}e_k = \left(u^{-1} - x_{k-1}\right) u \left(u^{-1} - x_{k-1}\right) = u \left(u^{-1} - x_{k-1}\right)^2 .$$

The abstract method explained in Eq. (34) is for the pointwise inverse of u specified by Algorithm 6.

5 Complexity Analysis

All discussed methods can be viewed as an inexact iteration procedure as mentioned in Eq. (34). For given initial guess and $\Phi_P : \mathcal{T} \to \mathcal{T}$ we have a recursive procedure defined in the following Algorithm 7. According to Lemma 1 and the problem depended definition of Φ_P the numerical cost of a function evaluation $z_k = \Phi_P(u_{k-1})$ is cheap if the tensor u_{k-1} is represented in the canonical tensor

Algorithm 6 Computing u^{-1}, $u \in \mathcal{R}_r$, $u_{\underline{i}} \neq 0$ for all $\underline{i} \in \mathcal{I}$

1: Choose $u_0 \in \mathcal{T}$ such that $\|\mathbb{1} - u \odot u_0\| < \|u\|$ and $\varepsilon \in \mathbb{R}_+$.
2: **while** $\|\mathbb{1} - u \odot u_{k-1}\| < \varepsilon \|u\|$ **do**
3:

$$z_k := u_{k-1} \odot (2\mathbb{1} - u \odot u_{k-1}),$$
$$u_k := \mathfrak{Appp}_{\varepsilon_k}(z_k),$$

4: **end while**

Algorithm 7 Inexact recursive iteration

1: Choose $u_0 \in \mathcal{T}$ and $\varepsilon \in \mathbb{R}_+$.
2: **while** error$(u_{k-1}) < \varepsilon$ **do**
3:

$$z_k := \Phi_P(u_{k-1}),$$
$$u_k := \mathfrak{Appp}_{\varepsilon_k}(z_k),$$

4: **end while**

format with moderate representation rank. The dominant part of the inexact iteration method is the approximation procedure $\mathfrak{Appp}_{\varepsilon_k}(z_k)$.

Remark 1. The complexity of the method $\mathfrak{Appp}_{\varepsilon_k}(z_k)$ described in [10, 12] is

$$\mathcal{O}\left(\sum_{r=r_{k-1}}^{r_\varepsilon} m_r \cdot \left[r \cdot (r + \operatorname{rank}(z_k)) \cdot d^2 + d \cdot r^3 + r \cdot (r + \operatorname{rank}(z_k) + d) \cdot \sum_{\mu=1}^{d} n_\mu\right]\right) \tag{53}$$

and for the method described in [13] we have

$$\mathcal{O}\left(\sum_{r=r_{k-1}}^{r_\varepsilon} \tilde{m}_r \cdot \left[d \cdot r^3 + r \cdot (r + \operatorname{rank}(z_k)) \cdot \sum_{\mu=1}^{d} n_\mu\right]\right) \tag{54}$$

where $r_{k-1} = \operatorname{rank}(u_{k-1})$ and m_r is the number of iterations in the regularised Newton method [10, 12] and $\tilde{m}_r$ is the number of iterations in the accelerated gradient method [13] for the rank-r approximation.

6 Numerical Experiments

The following numerical experiments were performed on usual 2-year-old PC. The multi-dimensional problem to be solved is defined in Eq. (7). The computational domain is 2D L-shape domain with $N = 557$ degrees of freedom (see Fig. 2).

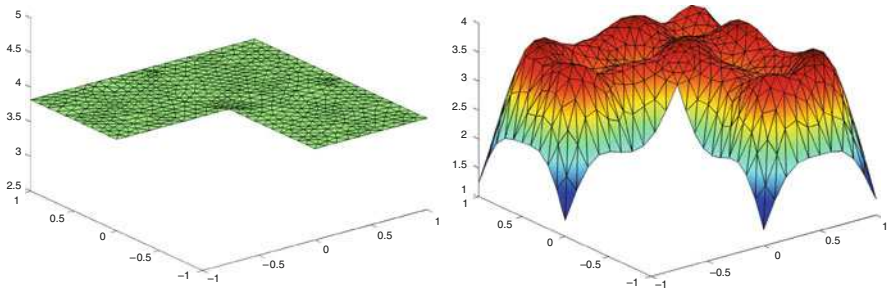


Fig. 2 Mean (on the *left*) and standard deviation (on the *right*) of $\kappa(x,\omega)$ (lognormal random field with parameters $\mu = 0.5$ and $\sigma = 1$)

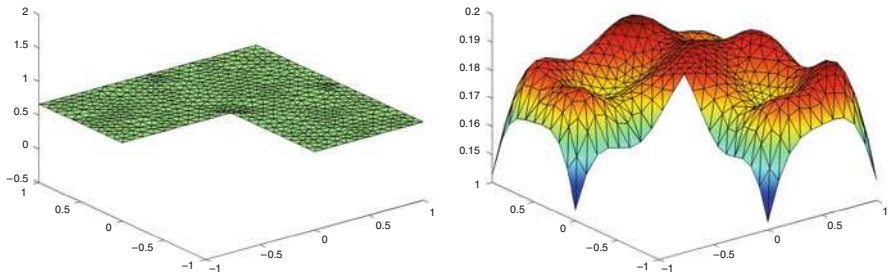


Fig. 3 Mean (on the *left*) and standard deviation (on the *right*) of $f(x,\omega)$ (beta distribution with parameters $\alpha = 4$, $\beta = 2$ and Gaussian cov. function)

The number of KLE terms for q in Eq. (8) is $l_k = 10$, the stochastic dimension is $m_k = 10$ and the maximal order of Hermite polynomials is $p_k = 2$. We took the shifted lognormal distribution for $\kappa(x,\omega)$ (see Eq. (8)), i.e., $\log(\kappa(x,\omega) - 1.1)$ has normal distribution with parameters $\{\mu = 0.5,\ \sigma^2 = 1.0\}$. The isotropic covariance function is of the Gaussian type with covariance lengths $\ell_x = \ell_y = 0.3$. The mean value and the standard deviation of $\kappa(x,\omega)$ are shown in Fig. 2.

For the right-hand side we took $l_f = 10$, $m_f = 10$ and $p_f = 2$ as well as Beta distribution with parameters $\{4, 2\}$ for random variables. The covariance function is also of the Gaussian type with covariance lengths $\ell_x = \ell_y = 0.6$. The mean value and the standard deviation of $\kappa(x,\omega)$ are shown in Fig. 3.

The Dirichlet boundary conditions in Eq. (7) were chosen as deterministic. Thus the total stochastic dimension of the solution u is $m_u = m_k + m_f = 20$, i.e. the multi-index α will consist of $m_u = 20$ indices ($\alpha = (\alpha_1, \ldots, \alpha_{m_u})$). The cardinality of the set of multi-indices $\mathcal{J}$ is $|\mathcal{J}| = \frac{(m_u+p_u)!}{m_u! p_u!}$, where $p_u = 2$. The solution tensor

$$u = \sum_{j=1}^{231} \bigotimes_{\mu=1}^{21} u_{j\mu} \in \mathbb{R}^{557} \otimes \bigotimes_{\mu=1}^{20} \mathbb{R}^3$$

with representation rank 231 was computed with the use of the stochastic Galerkin library [50]. The number 21 is a sum of the deterministic dimension 1 and the

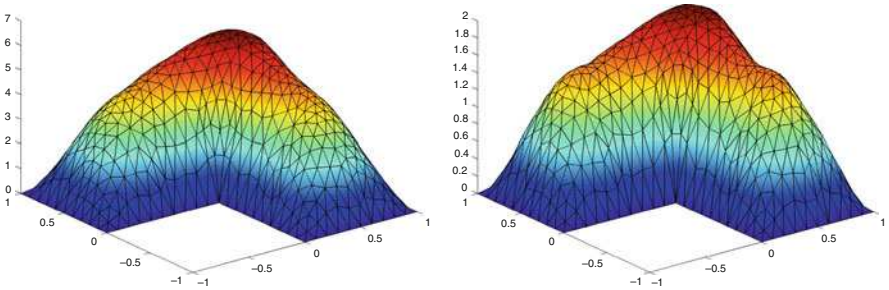


Fig. 4 Mean (on the *left*) and standard deviation (on the *right*) of the solution u

Table 1 Computation of $\operatorname{sign}(b\|u\|_\infty \mathbb{1} - u)$, where u is represented in the canonical tensor format with canonical rank 231, $d = 21$, $n_1 = 557$, and $p = 2$. The computing time to get any row is around 10 min. Note that the tensor u has $3^{20} * 557 = 1,942,138,911,357$ entries.

b	$\operatorname{rank}(\operatorname{sign}(b\|u\|_\infty \mathbb{1} - u))$	$\max_{1 \le k \le k_{\max}} \operatorname{rank}(u_k)$	$k_{\max}$	Error
0.2	12	24	12	2.9×10^{-8}
0.4	12	20	20	1.9×10^{-7}
0.6	8	16	12	1.6×10^{-7}
0.8	8	15	8	1.2×10^{-7}

stochastic dimension 20. The number 557 is the number of degrees of freedom in the computational domain. In the stochastic space we used polynomials of the maximal order 2 from 20 random variables and thus the solution belongs to the tensor space $\mathbb{R}^{557} \otimes \bigotimes_{\mu=1}^{20} \mathbb{R}^3$. The mean value and the standard deviation of the solution $u(x, \omega)$ are shown in Fig. 4.

Further we computed the maximal entry $\|u\|_\infty$ of u respect to the absolute value as described in Algorithm 4. The algorithm computed after 20 iterations the maximum norm $\|u\|_\infty$ effectually. The maximal representation rank of the intermediate iterants $(u_k)_{k=1}^{20}$ was 143, where we set the approximation error $\varepsilon_k = 1.0 \times 10^{-6}$ and $(u_k)_{k=1}^{20} \subset \mathcal{R}_{143}$ is the sequence of tensors generated by Algorithm 4. Finally, we computed $\operatorname{sign}(b\|u\|_\infty \mathbb{1} - u)$ for $b \in \{0.2, 0.4, 0.6, 0.8\}$. The results of the computation are documented in Table 1. The representation ranks of $\operatorname{sign}(b\|u\|_\infty \mathbb{1} - u)$ are given in the second column. In this numerical example, the ranks are smaller then 13. The iteration from Algorithm 5 determined after $k_{\max}$ steps the sign of $(b\|u\|_\infty \mathbb{1} - u)$, where the maximal representation rank of the iterants u_k from Algorithm 5 is documented in the third column. The error $\|\mathbb{1} - u_{k_{\max}} \odot u_{k_{\max}}\| / \|(b\|u\|_\infty \mathbb{1} - u)\|$ is given in the last column.

7 Conclusion

In this work we used sums of elementary tensors for the data analysis of solutions from stochastic elliptic boundary value problems. Particularly we explained how the new methods compute the maximum, minimum norms (Sect. 4.1), sign and

characteristic functions (Sect. 4.2), level sets (Sect. 4.3), mean, variance (Sect. 4.3), and pointwise inverse (Sect. 4.4). In the numerical example we considered a stochastic boundary value problem in the L-shape domain with stochastic dimension 20. Table 1 illustrates computation of quantiles of the solution (via sign function). Here the computation showed that the computational ranks are of moderate size. The computing time to get any row of Table 1 is around 10 min. To be able to perform the offered algorithms the solution u must already be approximated in a efficient tensor format. In this article we computed the stochastic solution in a sparse data format and then approximated it in the canonical tensors format. In a upcoming paper which will be submitted soon we compute the stochastic solution direct in the canonical tensor format and no transformation step is necessary.

References

1. S. Acharjee and N. Zabaras. A non-intrusive stochastic Galerkin approach for modeling uncertainty propagation in deformation processes. *Computers & Structures*, 85:244–254, 2007.
2. I. Babuška, R. Tempone, and G. E. Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM J. Numer. Anal.*, 42(2):800–825, 2004.
3. I. Babuška, R. Tempone, and G. E. Zouraris. Solving elliptic boundary value problems with uncertain coefficients by the finite element method: the stochastic formulation. *Comput. Methods Appl. Mech. Engrg.*, 194(12–16):1251–1294, 2005.
4. R. E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7:1–49, 1998.
5. S. R. Chinnamsetty, M. Espig, B. N. Khoromskij, W. Hackbusch, and H. J. Flad. Tensor product approximation with optimal rank in quantum chemistry. *The Journal of chemical physics*, 127(8):084–110, 2007.
6. G. Christakos. *Random Field Models in Earth Sciences*. Academic Press, San Diego, CA, 1992.
7. P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, Amsterdam, 1978.
8. O. G. Ernst, C. E. Powell, D. J. Silvester, and E. Ullmann. Efficient solvers for a linear stochastic Galerkin mixed formulation of diffusion problems with random data. *SIAM J. Sci. Comput.*, 31(2):1424–1447, 2008/09.
9. O. G. Ernst and E. Ullmann. Stochastic Galerkin matrices. *SIAM Journal on Matrix Analysis and Applications*, 31(4):1848–1872, 2010.
10. M. Espig. *Effiziente Bestapproximation mittels Summen von Elementartensoren in hohen Dimensionen*. PhD thesis, Dissertation, Universität Leipzig, 2008.
11. M. Espig, L. Grasedyck, and W. Hackbusch. Black box low tensor rank approximation using fibre-crosses. *Constructive approximation*, 2009.
12. M. Espig and W. Hackbusch. A regularized newton method for the efficient approximation of tensors represented in the canonical tensor format. *submitted Num. Math.*, 2011.
13. M. Espig, W. Hackbusch, T. Rohwedder, and R. Schneider. Variational calculus with sums of elementary tensors of fixed rank. *paper submitted to: Numerische Mathematik*, 2009.
14. P. Frauenfelder, Ch. Schwab, and R. A. Todor. Finite elements for elliptic problems with stochastic coefficients. *Comput. Methods Appl. Mech. Engrg.*, 194(2–5):205–228, 2005.
15. T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numer. Algorithms*, 18(3–4):209–232, 1998.
16. R. Ghanem. Ingredients for a general purpose stochastic finite elements implementation. *Comput. Methods Appl. Mech. Engrg.*, 168(1–4):19–34, 1999.
17. R. Ghanem. Stochastic finite elements for heterogeneous media with multiple random non-Gaussian properties. *Journal of Engineering Mechanics*, 125:24–40, 1999.

18. R. G. Ghanem and R. M. Kruger. Numerical solution of spectral stochastic finite element systems. *Computer Methods in Applied Mechanics and Engineering*, 129(3):289–303, 1996.
19. G. H. Golub and C. F. Van Loan. *Matrix Computations*. Wiley-Interscience, New York, 1984.
20. L. Grasedyck. *Theorie und Anwendungen Hierarchischer Matrizen*. Doctoral thesis, Universität Kiel, 2001.
21. W. Hackbusch, B. Khoromskij, and E. Tyrtyshnikov. Approximate iterations for structured matrices. *Numerische Mathematik*, 109:365–383, 2008.
22. M. Jardak, C.-H. Su, and G. E. Karniadakis. Spectral polynomial chaos solutions of the stochastic advection equation. In *Proceedings of the Fifth International Conference on Spectral and High Order Methods (ICOSAHOM-01) (Uppsala)*, volume 17, pages 319–338, 2002.
23. B. N. Khoromskij and A. Litvinenko. Data sparse computation of the Karhunen-Loève expansion. *Numerical Analysis and Applied Mathematics: Intern. Conf. on Num. Analysis and Applied Mathematics, AIP Conf. Proc.*, 1048(1):311–314, 2008.
24. B. N. Khoromskij, A. Litvinenko, and H. G. Matthies. Application of hierarchical matrices for computing Karhunen-Loève expansion. *Computing*, 84(1–2):49–67, 2009.
25. A. Klimke. Sparse Grid Interpolation Toolbox – user's guide. Technical Report IANS report 2007/017, University of Stuttgart, 2007.
26. A. Klimke and B. Wohlmuth. Algorithm 847: spinterp: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB. *ACM Transactions on Mathematical Software*, 31(4), 2005.
27. P. Krée and Ch. Soize. *Mathematics of random phenomena*, volume 32 of *Mathematics and its Applications*. D. Reidel Publishing Co., Dordrecht, 1986. Random vibrations of mechanical structures, Translated from the French by Andrei Iacob, With a preface by Paul Germain.
28. O. P. Le Maître, H. N. Najm, R. G. Ghanem, and O. M. Knio. Multi-resolution analysis of Wiener-type uncertainty propagation schemes. *J. Comput. Phys.*, 197(2):502–531, 2004.
29. A. Litvinenko and H. G. Matthies. Sparse data formats and efficient numerical methods for uncertainties quantification in numerical aerodynamics. Informatikbericht-Nr. 2010-01, http://www.digibib.tu-bs.de/?docid=00036490, Technische Universität Braunschweig, Braunschweig, 2010.
30. A. Litvinenko and H. G. Matthies. Sparse data formats and efficient numerical methods for uncertainties quantification in numerical aerodynamics. In *Proceedings of the IV European Congress on Computational Mechanics*, http://www.eccm2010.org/complet/fullpaper_1036.pdf, Paris, France, 2010.
31. H. G. Matthies. Computational aspects of probability in non-linear mechanics. In A. Ibrahimbegović and B. Brank, editors, *Engineering Structures under Extreme Conditions. Multi-physics and multi-scale computer models in non-linear analysis and optimal design of engineering structures under extreme conditions*, volume 194 of *NATO Science Series III: Computer and System Sciences*. IOS Press, Amsterdam, 2005.
32. H. G. Matthies. Quantifying uncertainty: Modern computational representation of probability and applications. In A. Ibrahimbegović, editor, *Extreme Man-Made and Natural Hazards in Dynamics of Structures*, NATO-ARW series. Springer Verlag, Berlin, 2007.
33. H. G. Matthies. Uncertainty quantification with stochastic finite elements. 2007. Part 1. Fundamentals. Encyclopedia of Computational Mechanics, John Wiley and Sons, Ltd.
34. H. G. Matthies. Stochastic finite elements: Computational approaches to stochastic partial differential equations. *Zeitschr. Ang. Math. Mech.(ZAMM)*, 88(11):849–873, 2008.
35. H. G. Matthies, Ch. E. Brenner, Ch. G. Bucher, and C. Guedes Soares. Uncertainties in probabilistic numerical analysis of structures and solids-stochastic finite elements. *Structural Safety*, 19(3):283–336, 1997.
36. H. G. Matthies and Ch. Bucher. Finite elements for stochastic media problems. *Comput. Meth. Appl. Mech. Eng.*, 168(1–4):3–17, 1999.
37. H. G. Matthies and A. Keese. Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Comput. Methods Appl. Mech. Engrg.*, 194(12–16):1295–1331, 2005.

38. H. G. Matthies and E. Zander. Solving stochastic systems with low-rank tensor compression. *Linear Algebra and its Applications*, 436:3819–3838, 2012.
39. F. Nobile, R. Tempone, and C. G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2309–2345, 2008.
40. E. Novak and K. Ritter. The curse of dimension and a universal method for numerical integration. In *Multivariate approximation and splines (Mannheim, 1996)*, volume 125 of *Internat. Ser. Numer. Math.*, pages 177–187. Birkhäuser, Basel, 1997.
41. E. Novak and K. Ritter. Simple cubature formulas with high polynomial exactness. *Constr. Approx.*, 15(4):499–522, 1999.
42. K. Petras. Fast calculation of coefficients in the Smolyak algorithm. *Numer. Algorithms*, 26(2):93–109, 2001.
43. L. J. Roman and M. Sarkis. Stochastic Galerkin method for elliptic SPDEs: a white noise approach. *Discrete Contin. Dyn. Syst. Ser. B*, 6(4):941–955 (electronic), 2006.
44. Ch. Schwab and C. J. Gittelson. Sparse tensor discretizations of high-dimensional parametric and stochastic pdes. *Acta Numerica*, 20:291–467, 2011.
45. S. A. Smoljak. Quadrature and interpolation formulae on tensor products of certain function classes. *Dokl. Akad. Nauk SSSR*, 148:1042–1045, 1963.
46. G. Strang and G. J. Fix. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, Wellesley, MA, 1988.
47. E. Ullmann. A Kronecker product preconditioner for stochastic Galerkin finite element discretizations. *SIAM Journal on Scientific Computing*, 32(2):923–946, 2010.
48. D. Xiu and G. E. Karniadakis. Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos. *Comput. Meth. Appl. Mech. Eng.*, 191:4927–4948, 2002.
49. X. Frank Xu. A multiscale stochastic finite element method on elliptic problems involving uncertainties. *Comput. Methods Appl. Mech. Engrg.*, 196(25–28):2723–2736, 2007.
50. E. Zander. Stochastic Galerkin library. *Technische Universität Braunschweig*, http://github.com/ezander/sglib, 2008.
51. O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method*. Butterwort-Heinemann, Oxford, 5th ed., 2000.

Sparse Grids in a Nutshell

Jochen Garcke

Abstract The technique of sparse grids allows to overcome the curse of dimensionality, which prevents the use of classical numerical discretization schemes in more than three or four dimensions, under suitable regularity assumptions. The approach is obtained from a multi-scale basis by a tensor product construction and subsequent truncation of the resulting multiresolution series expansion. This entry level article gives an introduction to sparse grids and the sparse grid combination technique.

1 Introduction

The sparse grid method is a special discretization technique, which allows to cope with the curse of dimensionality of grid based approaches to some extent. It is based on a hierarchical basis [13, 42, 43], a representation of a discrete function space which is equivalent to the conventional nodal basis, and a sparse tensor product construction.

The sparse grid method was originally developed for the solution of partial differential equations [5, 22, 45]. Besides working directly in the hierarchical basis a sparse grid representation of a function can also be computed using the combination technique [25], here a certain sequence of partial functions represented in the conventional nodal basis is linearly combined. The sparse grid method in both its formulations is nowadays successfully used in many applications.

The underlying idea of sparse grids can be traced back to the Russian mathematician Smolyak [37], who used it for numerical integration. The concept is also closely related to hyperbolic crosses [2, 38–40], boolean methods [11, 12], discrete blending methods [4] and splitting extrapolation methods [31].

J. Garcke (✉)
Institut für Numerische Simulation, Universität Bonn, Wegelerstr. 6, D-53115, Bonn, Germany
e-mail: garcke@ins.uni-bonn.de

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_3,

For the representation of a function f defined over a d-dimensional domain the sparse grid approach employs $\mathcal{O}(h_n^{-1} \cdot \log(h_n^{-1})^{d-1})$ grid points in the discretization process, where $h_n := 2^{-n}$ denotes the mesh size and n is the discretization level. It can be shown that the order of approximation to describe a function f, under certain smoothness conditions, is $\mathcal{O}(h_n^2 \cdot \log(h_n^{-1})^{d-1})$. This is in contrast to conventional grid methods, which need $\mathcal{O}(h_n^{-d})$ for an accuracy of $\mathcal{O}(h_n^2)$. Therefore, to achieve a similar approximation quality sparse grids need much less points in higher dimensions than regular full grids. The curse of dimensionality of full grid method arises for sparse grids to a much smaller extent and they can be used for higher dimensional problems.

For ease of presentation we will consider the domain $\Omega = [0,1]^d$ in the following. This situation can be achieved for bounded rectangular domains by a proper rescaling.

2 Sparse Grids

We introduce some notation while describing the conventional case of a piecewise linear finite element basis. Let $\underline{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$ denote a multi-index. We define the anisotropic grid $\Omega_{\underline{l}}$ on $\bar{\Omega}$ with mesh size $h_{\underline{l}} := (h_{l_1}, \ldots, h_{l_d}) = 2^{-\underline{l}} := (2^{-l_1}, \ldots, 2^{-l_d})$; $\Omega_{\underline{l}}$ has different, but equidistant mesh sizes h_{l_t} in each coordinate direction t, $t = 1, \ldots, d$. This way the grid $\Omega_{\underline{l}}$ consists of the points

$$x_{\underline{l},\underline{j}} := (x_{l_1,j_1}, \ldots, x_{l_d,j_d}), \tag{1}$$

with $x_{l_t,j_t} := j_t \cdot h_{l_t} = j_t \cdot 2^{-l_t}$ and $j_t = 0, \ldots, 2^{l_t}$. For a grid $\Omega_{\underline{l}}$ we define an associated space $V_{\underline{l}}$ of piecewise d-linear functions[1]

$$V_{\underline{l}} := \operatorname{span}\{\varphi_{\underline{l},\underline{j}} \mid j_t = 0, \ldots, 2^{l_t},\ t = 1, \ldots, d\} = \operatorname{span}\{\varphi_{\underline{l},\underline{j}} \mid 0 \leq \underline{j} \leq 2^{\underline{l}}\}, \tag{2}$$

which is spanned by the usual basis of d-dimensional piecewise d-linear hat functions

$$\varphi_{\underline{l},\underline{j}}(\underline{x}) := \prod_{t=1}^{d} \varphi_{l_t,j_t}(x_t). \tag{3}$$

The one-dimensional functions $\varphi_{l,j}(x)$ with support $[x_{l,j} - h_l, x_{l,j} + h_l] \cap [0,1] = [(j-1)h_l, (j+1)h_l] \cap [0,1]$ are defined by:

[1] "$\leq$" refers to the element-wise relation for multi-indices: $\underline{k} \leq \underline{l} :\Leftrightarrow \forall_t k_t \leq l_t$. Furthermore, $a \leq \underline{l}$ implies $\forall_t a \leq l_t$.

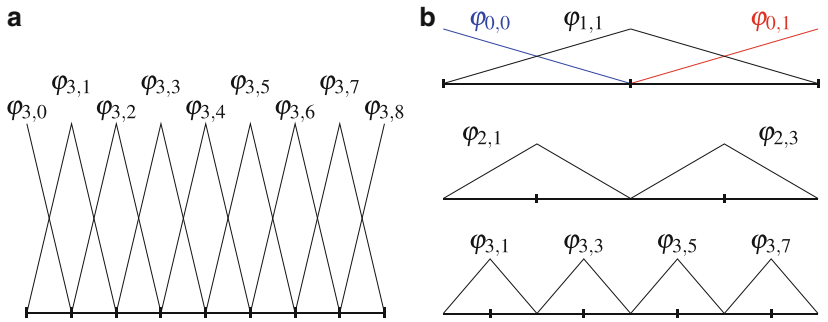


Fig. 1 Nodal and hierarchical basis of level $n = 3$. (**a**) Nodal basis for V_3. (**b**) Hierarchical basis for V_3

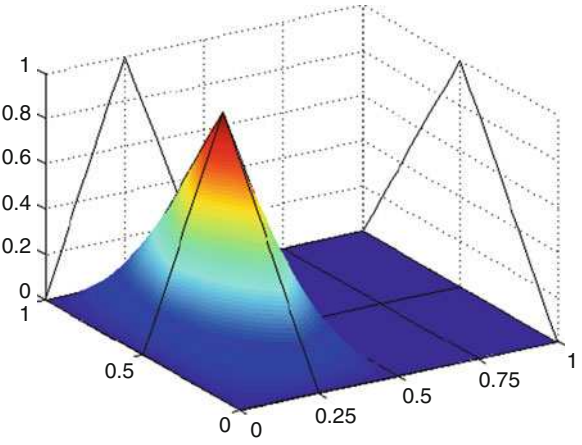


Fig. 2 Basis function $\varphi_{1,1}$ on grid $\Omega_{2,1}$

$$\varphi_{l,j}(x) = \begin{cases} 1 - |x/h_l - j|, & x \in [(j-1)h_l, (j+1)h_l] \cap [0,1], \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

In Fig. 1a we give an example for the one-dimensional case and show all $\varphi_{l,j} \in V_3$. Figure 2 shows a two-dimensional basis function.

2.1 Hierarchical Subspace-Splitting

Till now and in the following the multi-index $\underline{l} \in \mathbb{N}^d$ denotes the level, i.e. the discretization resolution, be it of a grid $\Omega_{\underline{l}}$, a space $V_{\underline{l}}$, or a function $f_{\underline{l}}$, whereas

the multi-index $\underline{j} \in \mathbb{N}^d$ gives the spatial position of a grid point $x_{\underline{l},\underline{j}}$ or the corresponding basis function $\varphi_{\underline{l},\underline{j}}(\cdot)$.

We now define a hierarchical difference space $W_{\underline{l}}$ via

$$W_{\underline{l}} := V_{\underline{l}} \setminus \bigoplus_{t=1}^{d} V_{\underline{l}-\underline{e}_t}, \tag{5}$$

where $\underline{e}_t$ is the t-th unit vector. In other words, $W_{\underline{l}}$ consists of all $\varphi_{\underline{k},\underline{j}} \in V_{\underline{l}}$ (using the hierarchical basis) which are not included in any of the spaces $V_{\underline{k}}$ smaller[2] than $V_{\underline{l}}$. To complete the definition, we formally set $V_{\underline{l}} := 0$, if $l_t = -1$ for at least one $t \in \{1, \ldots, d\}$. As can easily be seen from (2) and (5), the definition of the index set

$$\mathsf{B}_{\underline{l}} := \left\{ \underline{j} \in \mathbb{N}^d \;\middle|\; \begin{array}{ll} j_t = 1, \ldots, 2^{l_t} - 1, & j_t \text{ odd}, t = 1, \ldots, d, \text{ if } l_t > 0, \\ j_t = 0, 1, & t = 1, \ldots, d, \text{ if } l_t = 0 \end{array} \right\} \tag{6}$$

leads to

$$W_{\underline{l}} = \operatorname{span}\{\varphi_{\underline{l},\underline{j}} \,|\, \underline{j} \in \mathsf{B}_{\underline{l}}\}. \tag{7}$$

These hierarchical difference spaces now allow us the definition of a multilevel subspace decomposition. We can write $V_n := V_{\underline{n}}$ as a direct sum of subspaces

$$V_n := \bigoplus_{l_1=0}^{n} \cdots \bigoplus_{l_d=0}^{n} W_{\underline{l}} = \bigoplus_{|\underline{l}|_\infty \le n} W_{\underline{l}}. \tag{8}$$

Here, $|\underline{l}|_\infty := \max_{1 \le t \le d} l_t$ and $|\underline{l}|_1 := \sum_{t=1}^{d} l_t$ are the discrete ℓ_∞- and the discrete ℓ_1-norm of $\underline{l}$, respectively.

The family of functions

$$\{\varphi_{\underline{l},\underline{j}} \,|\, \underline{j} \in \mathsf{B}_{\underline{l}}\}_{\underline{l}=\underline{0}}^{\underline{n}} \tag{9}$$

is just the hierarchical basis [13, 42, 43] of V_n, which generalizes the one-dimensional hierarchical basis [13], see Fig. 1b, to the d-dimensional case with a tensor product ansatz. Observe that the supports of the basis functions $\varphi_{\underline{l},\underline{j}}(\underline{x})$, which span $W_{\underline{l}}$, are disjunct for $\underline{l} > 0$. See Fig. 3 for a representation of the supports of the basis functions of the difference spaces W_{l_1,l_2} forming V_3.

Now, each function $f \in V_n$ can be represented as

$$f(\underline{x}) = \sum_{|\underline{l}|_\infty \le n} \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \cdot \varphi_{\underline{l},\underline{j}}(\underline{x}) = \sum_{|\underline{l}|_\infty \le n} f_{\underline{l}}(\underline{x}), \quad \text{with } f_{\underline{l}} \in W_{\underline{l}}, \tag{10}$$

[2] We call a discrete space $V_{\underline{k}}$ smaller than a space $V_{\underline{l}}$ if $\forall_t k_t \le l_t$ and $\exists t : k_t < l_t$. In the same way a grid $\Omega_{\underline{k}}$ is smaller than a grid $\Omega_{\underline{l}}$.

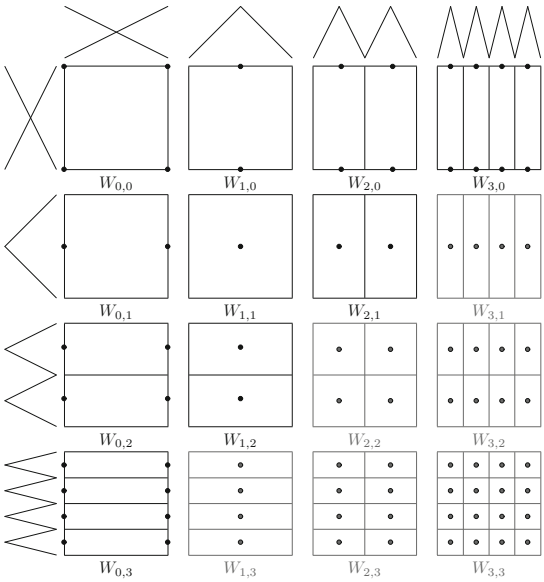


Fig. 3 Supports of the basis functions of the hierarchical subspaces $W_{\underline{l}}$ of the space V_3. The sparse grid space V_3^s contains the upper triangle of spaces shown in *black*

where $\alpha_{\underline{l},\underline{j}} \in \mathbb{R}$ are the coefficients of the representation in the hierarchical tensor product basis and $f_{\underline{l}}$ denotes the hierarchical component functions. The number of basis functions which describe a $f \in V_n$ in nodal or hierarchical basis is $(2^n + 1)^d$. For example a resolution of 17 points in each dimensions, i.e. $n = 4$, for a ten-dimensional problem therefore needs $2 \cdot 10^{12}$ coefficients, we encounter the curse of dimensionality.

Furthermore, we can define

$$V := \lim_{n \to \infty} \bigoplus_{\underline{k} \leq n} W_{\underline{k}},$$

which by a completion with respect to the H^1-norm, is simply the underlying Sobolev space H^1, i.e. $\overline{V}^{H^1} = H^1$. Any function $f \in V$ can be uniquely decomposed as [8]

$$f(\underline{x}) = \sum_{\underline{l} \in \mathbb{N}^d} f_{\underline{l}}(\underline{x}), \quad \text{with } f_{\underline{l}} \in W_{\underline{l}}.$$

Note also, that for the spaces $V_{\underline{l}}$ the following decomposition holds

$$V_{\underline{l}} := \bigoplus_{k_1=0}^{l_1} \cdots \bigoplus_{k_d=0}^{l_d} W_{\underline{k}} = \bigoplus_{\underline{k} \leq \underline{l}} W_{\underline{k}}.$$

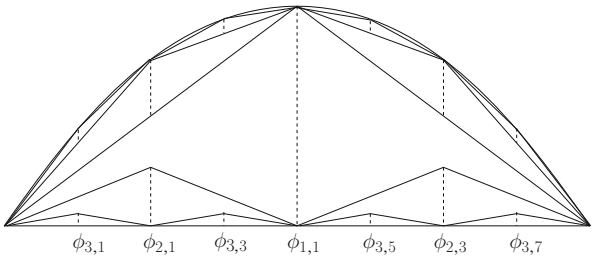


Fig. 4 Interpolation of a parabola with the hierarchical basis of level $n = 3$

2.2 Properties of the Hierarchical Subspaces

Now consider the d-linear interpolation of a function $f \in V$ by a $f_n \in V_n$, i.e. a representation as in (10). First we look at the linear interpolation in one dimension, for the hierarchical coefficients $\alpha_{l,j}$, $l \geq 1$, j odd, holds

$$\alpha_{l,j} = f(x_{l,j}) - \frac{f(x_{l,j} - h_l) + f(x_{l,j} + h_l)}{2} = f(x_{l,j}) - \frac{f(x_{l,j-1}) + f(x_{l,j+1})}{2}$$

$$= f(x_{l,j}) - \frac{f(x_{l-1,(j-1)/2}) + f(x_{l-1,(j+1)/2})}{2}.$$

This and Fig. 4 illustrate why the $\alpha_{l,j}$ are also called *hierarchical surplus*, they specify what has to be added to the hierarchical representation from level $l-1$ to obtain the one of level l. We can rewrite this in the following operator form

$$\alpha_{l,j} = \left[-\frac{1}{2} \quad 1 \quad -\frac{1}{2} \right]_{l,j} f$$

and with that we generalize to the d-dimensional hierarchization operator as follows

$$\alpha_{\underline{l},\underline{j}} = \left(\prod_{t=1}^{d} \left[-\frac{1}{2} \quad 1 \quad -\frac{1}{2} \right]_{l_t,j_t} \right) f. \tag{11}$$

Note that the coefficients for the basis functions associated to the boundary are just $\alpha_{0,j} = f(x_{0,j})$, $j = 0, 1$.

Now let us define the so-called Sobolev-space with dominating mixed derivative H^2_{mix} in which we then will show approximation properties of the hierarchical basis. First we consider mixed derivatives and define

$$D^{\underline{k}} f := \frac{\partial^{|\underline{k}|_1} f}{\partial \underline{x}^{\underline{k}}} = \frac{\partial^{|\underline{k}|_1} f}{\partial x_1^{k_1} \cdots \partial x_d^{k_d}}$$

With that we define the norm as

$$\|f\|^2_{H^s_{mix}} = \sum_{0\le \underline{k}\le s} \left| \frac{\partial^{|\underline{k}|_1}}{\partial \underline{x}^{\underline{k}}} f \right|^2_2 = \sum_{0\le \underline{k}\le s} \left| D^{\underline{k}} f \right|^2_2,$$

and the space H^s_{mix} in the usual way:

$$H^s_{mix} := \left\{ f : \Omega \to \mathbb{R} \,\middle|\, \|f\|^2_{H^s_{mix}} < \infty \right\}.$$

Obviously it holds $H^s_{mix} \subset H^s$. Furthermore we define the semi-norm $|f|_{H^2_{mix}} := |f|_{H^{\underline{2}}_{mix}}$ by

$$|f|_{H^{\underline{k}}_{mix}} := \left| \frac{\partial^{|\underline{k}|_1}}{\partial \underline{x}^{\underline{k}}} f \right|_2 = \left| D^{\underline{k}} f \right|_2.$$

Note that the continuous function spaces H^s_{mix}, like the discrete spaces $V_{\underline{l}}$, have a tensor product structure [24, 28, 29, 41] and can be represented as a tensor product of one dimensional spaces:

$$H^s_{mix} = H^s \otimes \cdots \otimes H^s.$$

We now look at the properties of the hierarchical representation of a function f, especially at the size of the hierarchical surpluses. We recite the following proofs from [5–8], see these references for more details on the following and results in other norms like $\|\cdot\|_\infty$ or $\|\cdot\|_E$. For ease of presentation we assume $f \in H^2_{0,mix}(\bar{\Omega})$, i.e. zero boundary values, and $\underline{l} > 0$ to avoid the special treatment of level 0, i.e. the boundary functions in the hierarchical representation.

Lemma 1. *For any piecewise d-linear basis function $\varphi_{\underline{l},\underline{j}}$ holds*

$$\|\varphi_{\underline{l},\underline{j}}\|_2 = \left(\frac{2}{3}\right)^{d/2} \cdot 2^{-|\underline{l}|_1/2}.$$

Proof. Follows by straightforward calculation.

Lemma 2. *For any hierarchical coefficient $\alpha_{\underline{l},\underline{j}}$ of $f \in H^2_{0,mix}(\bar{\Omega})$ in (10) it holds*

$$\alpha_{\underline{l},\underline{j}} = \prod_{t=1}^{d} -\frac{h_{l_t}}{2} \int_\Omega \varphi_{\underline{l},\underline{j}} \cdot D^{\underline{2}} f(\underline{x}) d\underline{x}. \tag{12}$$

Proof. In one dimension partial integration provides

$$\begin{aligned}
\int_\Omega \varphi_{l,j}(x) \cdot \frac{\partial^2 f(x)}{\partial x^2} dx &= \int_{x_{l,j}-h_l}^{x_{l,j}+h_l} \varphi_{l,j}(x) \cdot \frac{\partial^2 f(x)}{\partial x^2} dx \\
&= \left[\varphi_{l,j}(x) \cdot \frac{\partial f(x)}{\partial x} \right]_{x_{l,j}-h_l}^{x_{l,j}+h_l} - \int_{x_{l,j}-h_l}^{x_{l,j}+h_l} \frac{\partial \varphi_{l,j}(x)}{\partial x} \cdot \frac{\partial f(x)}{\partial x} dx \\
&= -\int_{x_{l,j}-h_l}^{x_{l,j}} \frac{1}{h_l} \cdot \frac{\partial f(x)}{\partial x} dx + \int_{x_{l,j}}^{x_{l,j}+h_l} \frac{1}{h_l} \cdot \frac{\partial f(x)}{\partial x} dx \\
&= \frac{1}{h_l} \cdot \big(f(x_{l,j} - h_l) - 2 f(x_{l,j}) + f(x_{l,j} + h_l) \big) \\
&= -\frac{2}{h_l} \cdot \alpha_{l,j}.
\end{aligned}$$

The d-dimensional result is achieved via the tensor product formulation (11).

Lemma 3. *Let $f \in H^2_{0,mix}(\bar{\Omega})$ be in hierarchical representation as above, it holds*

$$|\alpha_{\underline{l},\underline{j}}| \leq \frac{1}{6^{d/2}} \cdot 2^{-(3/2)\cdot|\underline{l}|_1} \cdot \left| f|_{supp(\varphi_{\underline{l},\underline{j}})} \right|_{H^2_{mix}}.$$

Proof.

$$\begin{aligned}
|\alpha_{\underline{l},\underline{j}}| &= \left| \prod_{t=1}^{d} -\frac{h_{l_t}}{2} \int_\Omega \varphi_{\underline{l},\underline{j}} \cdot D^2 f(\underline{x}) d\underline{x} \right| \leq \prod_{t=1}^{d} \frac{2^{-l_t}}{2} \cdot \|\varphi_{\underline{l},\underline{j}}\|_2 \cdot \left\| D^2 f|_{\mathrm{supp}(\varphi_{\underline{l},\underline{j}})} \right\|_2 \\
&\leq 2^{-d} \cdot \left(\frac{2}{3} \right)^{d/2} \cdot 2^{-(3/2)\cdot|\underline{l}|_1} \cdot \left| f|_{\mathrm{supp}(\varphi_{\underline{l},\underline{j}})} \right|_{H^2_{mix}}
\end{aligned}$$

Lemma 4. *For the components $f_{\underline{l}} \in W_{\underline{l}}$ of $f \in H^2_{0,mix}(\bar{\Omega})$ from (10) holds*

$$\| f_{\underline{l}} \|_2 \leq 3^{-d} \cdot 2^{-2\cdot|\underline{l}|_1} \cdot |f|_{H^2_{mix}}. \tag{13}$$

Proof. Since the supports of all $\varphi_{\underline{l},\underline{j}}$ of $f_{\underline{l}}$ are mutually disjoint we can write

$$\| f_{\underline{l}} \|_2^2 = \left\| \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \cdot \varphi_{\underline{l},\underline{j}} \right\|_2^2 = \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} |\alpha_{\underline{l},\underline{j}}|^2 \cdot \|\varphi_{\underline{l},\underline{j}}\|_2^2.$$

With Lemmas 3 and 1 it now follows

$$\|f_{\underline{l}}\|_2^2 \leq \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \frac{1}{6^d} \cdot 2^{-3 \cdot |\underline{l}|_1} \cdot \left| f|_{\mathrm{supp}(\varphi_{\underline{l},\underline{j}})} \right|_{H^2_{mix}}^2 \cdot \left(\frac{2}{3}\right)^d \cdot 2^{-|\underline{l}|_1}$$

$$\leq \frac{1}{3^{2d}} \cdot 2^{-4 \cdot |\underline{l}|_1} \cdot |f|_{H^2_{mix}}^2$$

which completes the proof.

2.3 Sparse Grids

Motivated by the relation (13) of the "importance" of the hierarchical components $f_{\underline{l}}$ Zenger [45] introduced the so-called *sparse grids*, where hierarchical basis functions with a small support, and therefore a small contribution to the function representation, are not included in the discrete space of level n anymore.

Formally we define the sparse grid function space $V_n^s \subset V_n$ as

$$V_n^s := \bigoplus_{|\underline{l}|_1 \leq n} W_{\underline{l}}. \tag{14}$$

We replace in the definition (8) of V_n in terms of hierarchical subspaces the condition $|\underline{l}|_\infty \leq n$ with $|\underline{l}|_1 \leq n$. In Fig. 3 the employed subspaces $W_{\underline{l}}$ are given in black, whereas in grey are given the difference spaces $W_{\underline{l}}$ which are omitted in comparison to (8). Every $f \in V_n^s$ can now be represented, analogue to (10), as

$$f_n^s(\underline{x}) = \sum_{|\underline{l}|_1 \leq n} \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \varphi_{\underline{l},\underline{j}}(\underline{x}) = \sum_{|\underline{l}|_1 \leq n} f_{\underline{l}}(\underline{x}), \quad \text{with } f_{\underline{l}} \in W_{\underline{l}}. \tag{15}$$

The resulting grid which corresponds to the approximation space V_n^s is called sparse grid. Examples in two and three dimensions are given in Fig. 5.

Note that sparse grids were introduced in [22,45], and are often used in this form, with a slightly different selection of hierarchical spaces using the definition

$$V_{0,n}^s := \bigoplus_{|\underline{l}|_1 \leq n+d-1} W_{\underline{l}}, \qquad l_t > 0. \tag{16}$$

This definition is especially useful when no degrees of freedom exist on the boundary, e.g. for the numerical treatment of partial differential equations with Dirichlet boundary conditions. Using $V_{0,n}^s$ the finest mesh size which comes from the level of refinement n in the sparse grid corresponds to the full grid case again when only interior points are considered.

The following results hold for both definitions V_n^s and $V_{0,n}^s$. The proofs are somewhat easier without basis functions on the boundary, therefore we only

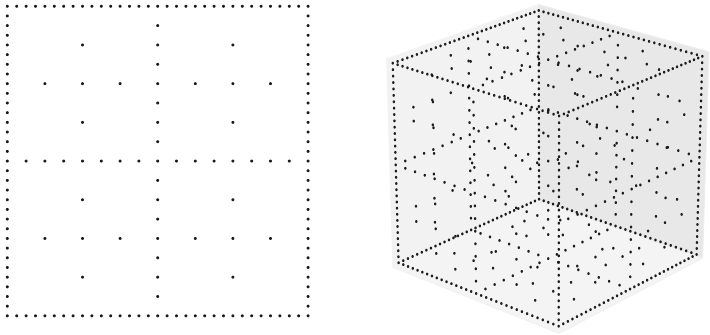


Fig. 5 Two-dimensional sparse grid (*left*) and three-dimensional sparse grid (*right*) of level $n = 5$

consider this case here, full results can be found in the given literature, e.g. [5–8, 30]. Furthermore, in the following we use the sparse grid space $V_{0,n}^s$, this allows us to have the same smallest mesh size h^{-n} inside the sparse grid of level n as in the corresponding full grid of level n and more closely follows the referenced original publications.

First we look at the approximation properties of sparse grids. For the proof we follow [8] and estimate the error for the approximation of a function $f \in H_{0,mix}^2$, which can be represented as $\sum_{\underline{l}} f_{\underline{l}}$, i.e. an infinite sum of partial functions from the hierarchical subspaces, by $f_{0,n}^s \in V_{0,n}^s$ which can be written as a corresponding finite sum. The difference therefore is

$$f - f_{0,n}^s = \sum_{\underline{l}} f_{\underline{l}} - \sum_{|\underline{l}|_1 \leq n+d-1} f_{\underline{l}} = \sum_{|\underline{l}|_1 > n+d-1} f_{\underline{l}}.$$

For any norm now holds

$$\|f - f_{0,n}^s\| \leq \sum_{|\underline{l}|_1 > n+d-1} \|f_{\underline{l}}\|. \tag{17}$$

We need the following technical lemma to estimate the interpolation error

Lemma 5. *For $s \in \mathbb{N}$ it holds*

$$\begin{aligned} \sum_{|\underline{l}|_1 > n+d-1} 2^{-s|\underline{l}|_1} &= 2^{-s\cdot n} \cdot 2^{-s\cdot d} \sum_{i=0}^{\infty} 2^{-s\cdot i} \cdot \binom{i+n+d-1}{d-1} \\ &\leq 2^{-s\cdot n} \cdot 2^{-s\cdot d} \cdot 2 \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}\left(n^{d-2}\right) \right), \end{aligned}$$

Proof. First we use that there are $\binom{i-1}{d-1}$ possibilities to represent i as a sum of d natural numbers

$$\sum_{|\underline{l}|_1 > n+d-1} 2^{-s|\underline{l}|_1} = \sum_{i=n+d}^{\infty} 2^{-s\cdot i} \cdot \sum_{|\underline{l}|_1 = i} 1$$
$$= \sum_{i=n+d}^{\infty} 2^{-s\cdot i} \cdot \binom{i-1}{d-1}$$
$$= 2^{-s\cdot n} \cdot 2^{-s\cdot d} \cdot \sum_{i=0}^{\infty} 2^{-s\cdot i} \cdot \binom{i+n+d-1}{d-1}.$$

We now represent the sum as the $(d-1)$-derivative of a function and get

$$\sum_{i=0}^{\infty} x^i \cdot \binom{i+n+d-1}{d-1}$$
$$= \frac{x^{-n}}{(d-1)!} \left(\sum_{i=0}^{\infty} x^{i+n+d-1} \right)^{(d-1)} = \frac{x^{-n}}{(d-1)!} \cdot \left(x^{n+d-1} \cdot \frac{1}{1-x} \right)^{(d-1)}$$
$$= \frac{x^{-n}}{(d-1)!} \cdot \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \left(x^{n+d-1}\right)^{(k)} \cdot \left(\frac{1}{1-x} \right)^{(d-1-k)}$$
$$= \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \frac{(n+d-1)!}{(n+d-1-k)!} \cdot x^{d-1-k} \cdot \frac{(d-1-k)!}{(d-1)!} \cdot \left(\frac{1}{1-x} \right)^{d-1-k+1}$$
$$= \sum_{k=0}^{d-1} \binom{n+d-1}{k} \cdot \left(\frac{x}{1-x} \right)^{d-1-k} \cdot \frac{1}{1-x}.$$

With $x = 2^{-s}$ it follows

$$\sum_{i=0}^{\infty} 2^{-s\cdot i} \cdot \binom{i+n+d-1}{d-1} \leq 2 \cdot \sum_{k=0}^{d-1} \binom{n+d-1}{k}.$$

The summand for $k = d-1$ is the largest one and it holds

$$2 \cdot \frac{(n+d-1)!}{(d-1)!n!} = 2 \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}\left(n^{d-2}\right) \right)$$

which finishes the proof.

Theorem 1. *For the interpolation error of a function $f \in H^2_{0,mix}$ in the sparse grid space $V^s_{0,n}$ holds*

$$||f - f^s_n||_2 = \mathcal{O}(h_n^2 \log(h_n^{-1})^{d-1}). \tag{18}$$

Proof. Using Lemmas 4 and 5 we get

$$||f - f_n^s||_2 \leq \sum_{|\underline{l}|_1 > n+d-1} \|f_{\underline{l}}\|_2 \leq 3^{-d} \cdot 2^{-2|\underline{l}|_1} \cdot |f|_{H^2_{mix}}$$
$$\leq 3^{-d} \cdot 2^{-2 \cdot n} \cdot |f|_{H^2_{mix}} \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2}) \right),$$

which gives the desired relation.

Note that corresponding results hold in the maximum-norm as well:

$$||f - f_n^s||_\infty = \mathcal{O}\left(h_n^2 \log(h_n^{-1})^{d-1}\right)$$

for $f \in H^2_{0,mix}$ and that for the energy norm one achieves $\mathcal{O}(h_n)$, here the order is the same as in the full grid case [8].

We see that the approximation properties in the L_2-norm for functions from $H^2_{0,mix}$ when using sparse grids are somewhat worse in comparison to full grids, which achieve $\mathcal{O}(h_n^2)$. But this is offset by the much smaller number of grid points needed, as we will see when we now look at the size of the sparse grid space.

Lemma 6. *The dimension of the sparse grid space $\mathring{V}^s_{0,n}$, i.e. the number of inner grid points, is given by*

$$\left|\mathring{V}^s_{0,n}\right| = \mathcal{O}\left(h_n^{-1} \cdot \log(h_n^{-1})^{d-1}\right) \tag{19}$$

Proof. We again follow [8] and use in the first part the definition (16) and the size of a hierarchical subspace $|W_{\underline{l}}| = 2^{|\underline{l}-\underline{1}|_1}$. The following steps use similar arguments as in the preceding Lemma 5.

$$\left|\mathring{V}^s_{0,n}\right| = \left| \bigoplus_{|\underline{l}|_1 \leq n+d-1} W_{\underline{l}} \right| = \sum_{|\underline{l}|_1 \leq n+d-1} 2^{|\underline{l}-\underline{1}|_1} = \sum_{i=d}^{n+d-1} 2^{i-d} \cdot \sum_{|\underline{l}|_1 = i} 1$$
$$= \sum_{i=d}^{n+d-1} 2^{i-d} \cdot \binom{i-1}{d-1} = \sum_{i=0}^{n-1} 2^i \cdot \binom{i+d-1}{d-1}.$$

We now represent the summand as the $(d-1)$-derivative of a function evaluated at $x = 2$

$$\sum_{i=0}^{n-1} x^i \cdot \binom{i+d-1}{d-1}$$
$$= \frac{1}{(d-1)!} \sum_{i=0}^{n-1} \left(x^{i+d-1}\right)^{(d-1)} = \frac{1}{(d-1)!} \left(x^{d-1} \cdot \frac{1-x^n}{1-x} \right)^{(d-1)}$$

$$
\begin{aligned}
&= \frac{1}{(d-1)!} \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \left(x^{d-1} - x^{n+d-1}\right)^{(k)} \cdot \left(\frac{1}{1-x}\right)^{(d-1-k)} \\
&= \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \frac{(d-1)!}{(d-1-k)!} \cdot x^{d-1-k} \cdot \frac{(d-1-k)!}{(d-1)!} \cdot \left(\frac{1}{1-x}\right)^{d-1-k+1} \\
&- \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \frac{(n+d-1)!}{(n+d-1-k)!} \cdot x^{n+d-1-k} \cdot \frac{(d-1-k)!}{(d-1)!} \cdot \left(\frac{1}{1-x}\right)^{d-1-k+1} \\
&= \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \left(\frac{x}{1-x}\right)^{d-1-k} \cdot \frac{1}{1-x} \\
&- x^n \cdot \sum_{k=0}^{d-1} \binom{n+d-1}{k} \cdot \left(\frac{x}{1-x}\right)^{d-1-k} \cdot \frac{1}{1-x}.
\end{aligned}
$$

We observe that the first sum is constant in n and therefore not relevant for the order, but note that for $x = 2$ that sum falls down to $(-1)^d$ anyway and get

$$
(-1)^d + 2^n \cdot \sum_{k=0}^{d-1} \binom{n+d-1}{k} \cdot (-2)^{d-1-k}.
$$

The summand for $k = d-1$ is again the largest one and it holds

$$
2^n \cdot \frac{(n+d-1)!}{(d-1)!n!} = 2^n \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2})\right)
$$

which gives a total order of $\mathcal{O}(2^n \cdot n^{d-1})$ or in other notation, with $h_n = 2^{-n}$, of $\mathcal{O}(h_n^{-1} \cdot \log(h_n^{-1})^{d-1})$.

This is far less than the size of the corresponding full grid space $|\mathring{V}_n| = \mathcal{O}(h_n^{-d}) = \mathcal{O}(2^{d \cdot n})$ and allows the treatment of higher dimensional problems while still achieving good accuracy.

Note that a practical realisation of sparse grids involves suitable data structures and special algorithms, e.g. for efficient matrix-vector multiplications in Galerkin methods for the numerical solution of partial differential equations. Further details and references can be found for example in [14, 34, 44].[3] Also note that sparse grid functions do not possess some properties which full grid functions have, e.g. a sparse grid function need not be monotone [32, 34].

[3] Note that for the purpose of interpolation a sparse grid toolbox for Matlab is available at http://www.ians.uni-stuttgart.de/spinterp/.

The sparse grid structure introduced so far defines an a priori selection of grid points that is optimal if certain smoothness conditions are met, i.e. if the function has bounded second mixed derivatives, and no further knowledge of the function is known or used. If the aim is to approximate functions which do not fulfil this smoothness condition, or to represent functions that show significantly differing characteristics, e.g. very steep regions beyond flat ones, spatially adaptive refinement may be used as well. Depending on the characteristics of the problem and function at hand adaptive refinement strategies decide which points, and corresponding basis functions, should be incrementally added to the sparse grid representation to increase the accuracy.

In the sparse grid setting, usually an error indicator coming directly from the hierarchical basis is employed [14, 23, 34, 35]: depending on the size of the hierarchical surplus $\alpha_{\underline{l},\underline{j}}$ it is decided whether a basis function should be marked for further improvement or not. This is based on two observations: First, the hierarchical surplus gives the absolute change in the discrete representation at point $x_{\underline{l},\underline{j}}$ due to the addition of the corresponding basis function $\varphi_{\underline{l},\underline{j}}$, it measures its contribution in a given sparse grid representation (15) in the maximum-norm. And second, a hierarchical surplus represents discrete second derivatives according to (12) and hence can be interpreted as a measure of the smoothness of the considered function at point $x_{\underline{l},\underline{j}}$. Further details on spatially adaptive sparse grids, their realisation and the state of the art can be found in [14, 34, 35].

2.4 Hierarchy Using Constant Functions

An alternative hierarchical representation of a function in V_n is based on a slightly different hierarchy, which starts at level -1 with the constant. To be precise, we define the one-dimensional basis functions $\widetilde{\varphi}_{l,j}(x)$ by

$$\begin{aligned} \widetilde{\varphi}_{-1,0} &:= 1, \\ \widetilde{\varphi}_{0,0} &:= \varphi_{0,1}, \\ \widetilde{\varphi}_{l,j} &:= \varphi_{l,j} \quad \text{for } l \geq 1, \end{aligned}$$

with $\varphi_{l,j}$ defined as in (4). Obviously it holds $\varphi_{0,0} = \widetilde{\varphi}_{-1,0} - \widetilde{\varphi}_{0,0}$. The d-dimensional basis functions are constructed as a tensor product as before

$$\widetilde{\varphi}_{\underline{l},\underline{j}}(\underline{x}) := \prod_{t=1}^{d} \widetilde{\varphi}_{l_t,j_t}(x_t). \tag{20}$$

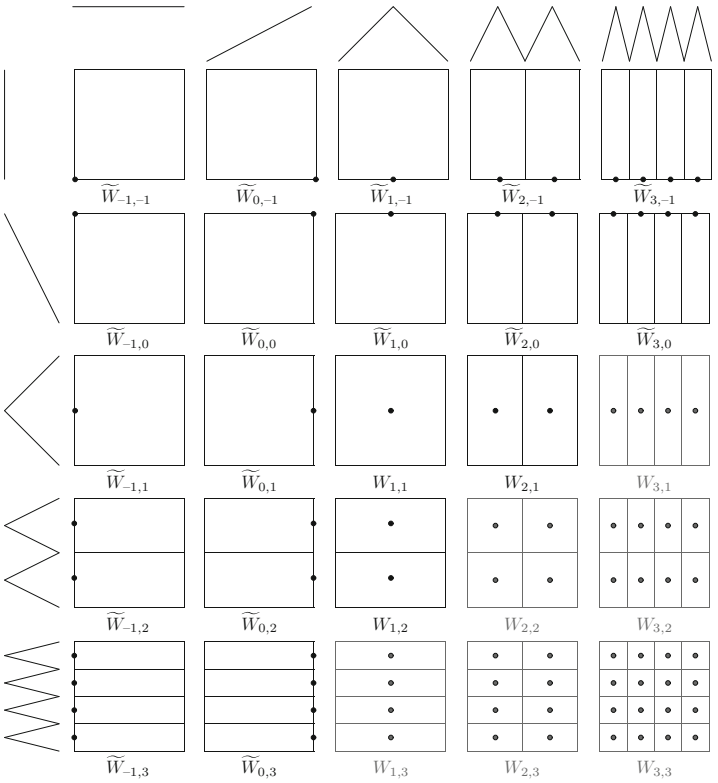


Fig. 6 Supports of the basis functions of the hierarchical subspaces $W_{\underline{l}}$ and $\widetilde{W}_{\underline{l}}$ of the space V_3. The sparse grid space V_3^s contains the upper triangle of spaces shown in *black*, the space $\widetilde{V}_3^s$ includes also $\widetilde{W}_{4,-1}$ and $\widetilde{W}_{-1,4}$, which are not shown

We introduce index sets $\widetilde{\mathsf{B}}_{\underline{l}}$ analogue to (6)

$$\widetilde{\mathsf{B}}_{\underline{l}} := \left\{ \underline{j} \in \mathbb{N}^d \;\middle|\; \begin{array}{ll} j_t = 1, \ldots, 2^{l_t} - 1, & j_t \text{ odd}, t = 1, \ldots, d, \text{ if } l_t > 0, \\ j_t = 0, & t = 1, \ldots, d, \text{ if } l_t \in \{0, -1\} \end{array} \right\}.$$

Now we can define slightly modified hierarchical difference spaces $\widetilde{W}_{\underline{l}}$ analogue to (7) by

$$\widetilde{W}_{\underline{l}} = \text{span}\{\widetilde{\varphi}_{\underline{l},\underline{j}}, \underline{j} \in \widetilde{\mathsf{B}}_{\underline{l}}\}, \tag{21}$$

see Fig. 6.

It is easy to see that $\widetilde{W}_{\underline{l}} = W_{\underline{l}}$ holds for $\underline{l} \geq \underline{0}$. We now can define a full grid space $\widetilde{V}_n$ by using the newly defined modified hierarchical subspaces

$$\widetilde{V}_n := \bigoplus_{l_1=-1}^{n} \cdots \bigoplus_{l_d=-1}^{n} \widetilde{W}_{\underline{l}} = \bigoplus_{|\underline{l}|_\infty \leq n} \widetilde{W}_{\underline{l}}. \tag{22}$$

Again it holds $\widetilde{V}_n = V_n$ for $n \geq 0$.

A corresponding definition of a sparse grid space $\widetilde{V}_n^s$ using

$$\widetilde{V}_n^s := \bigoplus_{|\underline{l}|_1 \leq n} \widetilde{W}_{\underline{l}} \tag{23}$$

on the other hand does not give the original sparse grid space V_n^s. But if we exclude a few spaces it holds

$$V_n^s = \widetilde{V}_n^s \setminus \bigoplus_{\substack{|\underline{l}|_1 = n \text{ and} \\ \exists l_t = -1}} \widetilde{W}_{\underline{l}} \quad \text{for } n \geq 0, \tag{24}$$

see Fig. 6.

As before, every $f \in \widetilde{V}_n^s$ can now be represented, analogue to (15), as

$$f(\underline{x}) = \sum_{|\underline{l}|_1 \leq n} \sum_{\underline{j} \in \widetilde{\mathsf{B}}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \varphi_{\underline{l},\underline{j}}(\underline{x}) = \sum_{|\underline{l}|_1 \leq n} f_{\underline{l}}(\underline{x}) \quad \text{with } f_{\underline{l}} \in \widetilde{W}_{\underline{l}}. \tag{25}$$

The key observation is now that the partial functions $f_{\underline{l}}$ with $\exists l_t = -1$ are lower-dimensional functions: they are constant in those dimensions t where $l_t = -1$; $f_{\underline{l}}$ possesses no degree of freedom in these dimensions. Such a function representation for $f(\underline{x})$ can therefore be formally written in the *ANalysis Of VAriance* (ANOVA) form, which is well known from statistics,

$$f(\underline{x}) = f_{\underline{-1}} + \sum_{\substack{|\underline{l}|_1 \leq n \text{ and} \\ |\{l_t | l_t = -1\}| = d-1}} f_{\underline{l}} + \cdots + \sum_{\substack{|\underline{l}|_1 \leq n \text{ and} \\ |\{l_t | l_t = -1\}| = 1}} f_{\underline{l}} + \sum_{\substack{|\underline{l}|_1 \leq n \text{ and} \\ |\{l_t | l_t = -1\}| = 0}} f_{\underline{l}}, \tag{26}$$

with $f_{\underline{l}} \in \widetilde{W}_{\underline{l}}$. The ANOVA order, the number of relevant non-constant dimensions, of the component functions $f_{\underline{l}}$ grows from 0 on the left to d on the right.

At this stage this is just a formal play with the representation, but it becomes quite relevant when one can build such a representation for a given function in an adaptive fashion, i.e. one chooses which component functions up to which ANOVA order are used for a reasonable approximation of some f. If the ANOVA order can be limited to q with $q \ll d$, the complexity estimates do not depend on the dimension d but on the ANOVA order q, allowing the treatment of even higher dimensional problems. An ANOVA-based dimension adaptive refinement algorithm in the hierarchical sparse grid basis is presented and evaluated in [14].

3 Sparse Grid Combination Technique

The so-called *combination technique* [25], which is based on multi-variate extrapolation [10], is another method to achieve a function representation on a sparse grid. The function is discretized on a certain sequence of grids using a nodal discretization. A linear combination of these partial functions then gives the sparse grid representation. This approach can have numerical advantages over working directly in the hierarchical basis, where e.g. the stiffness matrix is not sparse and efficient computations of the matrix-vector-product are challenging in the implementation [1, 3, 6, 14, 34, 44]. There are close connections of the combination technique to boolean [11, 12] and discrete blending methods [4], as well as the splitting extrapolation-method [31].

In particular, we discretize a function f on a certain sequence of anisotropic grids $\Omega_{\underline{l}} = \Omega_{l_1,\ldots,l_d}$ with uniform mesh sizes $h_t = 2^{-l_t}$ in the t-th coordinate direction. These grids possess in general different mesh sizes for the different coordinate directions. To be precise, we consider all grids $\Omega_{\underline{l}}$ with

$$|\underline{l}|_1 := l_1 + \ldots + l_d = n - q, \quad q = 0, .., d-1, \quad l_t \geq 0. \tag{27}$$

The grids employed by the combination technique of level 4 in two dimensions are shown in Fig. 7.

Note that in the original [25] and other papers, a slightly different definition was used:

$$|\underline{l}|_1 := l_1 + \ldots + l_d = n + (d-1) - q, \quad q = 0, .., d-1, \quad l_t > 0.$$

This is again in view of situations where no degrees of freedom are needed on the boundary, e.g. for Dirichlet boundary conditions, see (16) and the remarks afterwards.

A finite element approach with piecewise d-linear functions $\varphi_{\underline{l},\underline{j}}(\underline{x})$ on each grid $\Omega_{\underline{l}}$ now gives the representation in the nodal basis

$$f_{\underline{l}}(\underline{x}) = \sum_{j_1=0}^{2^{l_1}} \cdots \sum_{j_d=0}^{2^{l_d}} \alpha_{\underline{l},\underline{j}} \varphi_{\underline{l},\underline{j}}(\underline{x}).$$

Finally, we linearly combine the discrete partial functions $f_{\underline{l}}(\underline{x})$ from the different grids $\Omega_{\underline{l}}$ according to the combination formula

$$f_n^c(\underline{x}) := \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{|\underline{l}|_1 = n-q} f_{\underline{l}}(\underline{x}). \tag{28}$$

The resulting function f_n^c lives in the sparse grid space V_n^s, where the combined interpolant is identical with the hierarchical sparse grid interpolant f_n^s [25]. This can be seen by rewriting each $f_{\underline{l}}$ in their hierarchical representation (10) and some

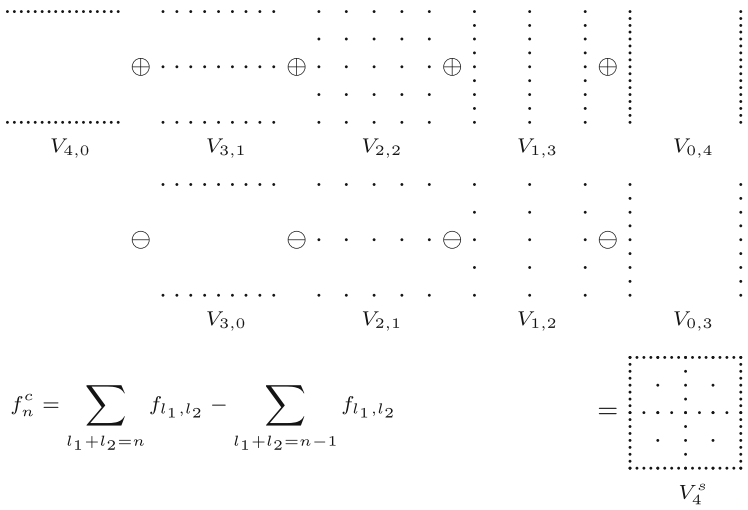


Fig. 7 Combination technique with level $n = 4$ in two dimensions

straightforward calculation using the telescope sum property, i.e. the hierarchical functions get added and subtracted.

Lemma 7. *For a given function f the interpolant f_n^c using the combination technique (28) is the hierarchical sparse grid interpolant f_n^s from (15).*

Proof. We write it exemplary in the two dimensional case, using $\hat{f}_{l_1+l_2} \in W_{l_1,l_2}$ instead of all the basis functions of W_{l_1,l_2} for ease of presentation:

$$
\begin{aligned}
f_n^c &= \sum_{l_1+l_2=n} f_{l_1,l_2} - \sum_{l_1+l_2=n-1} f_{l_1,l_2} \\
&= \sum_{l_1\le n} \sum_{k_1\le l_1} \sum_{k_2\le n-l_1} \hat{f}_{k_1,k_2} - \sum_{l_1\le n-1} \sum_{k_1\le l_1} \sum_{k_2\le n-l_1-1} \hat{f}_{k_1,k_2} \\
&= \sum_{k_1\le l_1=n} \sum_{k_2=0} \hat{f}_{k_1,k_2} + \sum_{l_1\le n-1} \sum_{k_1\le l_1} \left(\sum_{k_2\le n-l_1} \hat{f}_{k_1,k_2} - \sum_{k_2\le n-l_1-1} \hat{f}_{k_1,k_2} \right) \\
&= \sum_{k_1\le l_1=n} \sum_{k_2=n-l_1} \hat{f}_{k_1,k_2} + \sum_{l_1\le n-1} \sum_{k_1\le l_1} \sum_{k_2=n-l_1} \hat{f}_{k_1,k_2} \\
&= \sum_{l_1\le n} \sum_{k_2=n-l_1} \sum_{k_1\le n-k_2} \hat{f}_{k_1,k_2} \\
&= \sum_{k_2\le n} \sum_{k_1\le n-k_2} \hat{f}_{k_1,k_2} = \sum_{k_1+k_2\le n} \hat{f}_{k_1,k_2}
\end{aligned}
$$

This last expression is exactly (15).

Alternatively, one can view the combination technique as an approximation of a projection into the underlying sparse grid space. The combination technique is then an exact projection into the sparse grid space if and only if the partial projections commute, i.e. the commutator $[P_{V_1}, P_{V_2}] := P_{V_1} P_{V_2} - P_{V_2} P_{V_1}$ is zero for all pairs of involved grids [27].

Note that the solution obtained with the combination technique f_n^c for the numerical treatment of partial differential equations, i.e. when the solutions on the partial grids are combined according to the combination formula (28), is in general not the sparse grid solution f_n^s. However, the approximation property is of the same order as long as a certain series expansion of the error exists [25]. Its existence was shown for model-problems in [9].

Lemma 8. *Assume that the exact solution f is sufficiently smooth and that the pointwise error expansion*

$$f - f_{\underline{l}} = \sum_{i=1}^{d} \sum_{j_1,\ldots,j_m \subset 1,\ldots,d} c_{j_1,\ldots,j_m}(h_{j_1}, \ldots, h_{j_m}) \cdot h_{j_1}^p \cdot \ldots \cdot h_{j_m}^p, \tag{29}$$

with bounded $c_{j_1,\ldots,j_m}(h_{j_1}, \ldots, h_{j_m}) \leq \kappa$, holds for $\underline{l} \leq n$. Then

$$|f - f_n^c| = \mathcal{O}\left(h_n^2 \cdot \log(h_n^{d-1})\right). \tag{30}$$

Proof. Let us again consider the two dimensional case and consider the pointwise error of the combined solution $f - f_n^c$ following [25]. We have

$$\begin{aligned} f - f_n^c &= f - \sum_{l_1+l_2=n} f_{l_1,l_2} + \sum_{l_1+l_2=n-1} f_{l_1,l_2} \\ &= \sum_{l_1+l_2=n} (f - f_{l_1,l_2}) - \sum_{l_1+l_2=n-1} (f - f_{l_1,l_2}). \end{aligned}$$

Plugging in the error expansion (29) leads to

$$\begin{aligned} f - f_n^c &= \sum_{l_1+l_2=n} \left(c_1(h_{l_1}) \cdot h_{l_1}^2 + c_2(h_{l_2}) \cdot h_{l_2}^2 + c_{1,2}(h_{l_1}, h_{l_2}) \cdot h_{l_1}^2 h_{l_2}^2\right) \\ &\quad - \sum_{l_1+l_2=n-1} \left(c_1(h_{l_1}) \cdot h_{l_1}^2 + c_2(h_{l_2}) \cdot h_{l_2}^2 + c_{1,2}(h_{l_1}, h_{l_2}) \cdot h_{l_1}^2 h_{l_2}^2\right) \\ &= \Bigg(c_1(h_n) + c_2(h_n) + \sum_{l_1+l_2=n} c_{1,2}(h_{l_1}, h_{l_2}) \\ &\quad - 4 \sum_{l_1+l_2=n-1} c_{1,2}(h_{l_1}, h_{l_2}) \Bigg) \cdot h_n^2. \end{aligned}$$

And using $c_i \leq \kappa$ we get the estimate (30)

$$\begin{aligned}
|f - f_n^c| &\leq 2\kappa \cdot h_n^2 + \left| \sum_{l_1+l_2=n} c_{1,2}(h_{l_1}, h_{l_2}) - 4 \sum_{l_1+l_2=n-1} c_{1,2}(h_{l_1}, h_{l_2}) \right| \cdot h_n^2 \\
&\leq 2\kappa \cdot h_n^2 + \sum_{l_1+l_2=n} |c_{1,2}(h_{l_1}, h_{l_2})| \cdot h_n^2 + 4 \sum_{l_1+l_2=n-1} |c_{1,2}(h_{l_1}, h_{l_2})| \cdot h_n^2 \\
&\leq 2\kappa \cdot h_n^2 + \kappa \cdot n h_n^2 + 4\kappa(n-1)h_n^2 \\
&= \kappa \cdot h_n^2(5n-2) = \kappa \cdot h_n^2(5\log(h_n^{-1}) - 2) \\
&= \mathcal{O}\left(h_n^2 \cdot \log(h_n^{-1})\right).
\end{aligned}$$

Observe that cancellation occurs for h_{l_i} with $l_i \neq n$ and the accumulated $h_{l_1}^2 h_{l_2}^2$-terms result in the $\log(h_n^{-1})$-term. The approximation order $\mathcal{O}(h_n^2 \cdot \log(h_n^{-1}))$ is just as in Theorem 1. See [25, 33, 36] for results in higher dimensions.

Similar to (26) one can consider an ANOVA representation in the form of a combination technique, which in general terms is a function representation for $f(\underline{x})$ of the type

$$f(\underline{x}) = \sum_{\{j_1,\ldots,j_q\} \subset \{1,\ldots,d\}} c_{j_1,\ldots,j_q} f_{j_1,\ldots,j_q}(x_{j_1}, \ldots, x_{j_q}), \tag{31}$$

where each $f_{j_1,\ldots,j_q}(x_{j_1}, \ldots, x_{j_q})$ depends only on a subset of size q of the dimensions and may have different refinement levels for each dimension. Again, one especially assumes here that $q \ll d$, so that the computational complexity depends on the so-called *superposition* (or *effective*) dimension q. The hierarchy here again starts with a level -1 of constant functions and we note again that if one builds the tensor product between a constant in one dimension and a $(d-1)$-linear function the resulting d-dimensional function is still $(d-1)$-linear, one gains no additional degrees of freedom. But formally introducing a level -1, and using this as coarsest level, will allow us to write a combined function in the ANOVA-style (31), in other words each partial function might only depend on a subset of all dimensions. The size of each grid $\Omega_{\underline{l}}$ is now of order $\mathcal{O}(2^q(|\underline{l}|_1 + (d-q))$, where $q = \#\{l_i | l_i \geq 0\}$.

An advantage of such a viewpoint arises if one can select which grids to employ and does not use the grid sequence (27). In such a so-called dimension adaptive procedure one considers an index set I which only needs to fulfil the following *admissibility condition* [21, 26]

$$\underline{k} \in \mathsf{I} \text{ and } \underline{j} \leq \underline{k} \quad \Rightarrow \quad \underline{j} \in \mathsf{I}, \tag{32}$$

in other words an index $\underline{k}$ can only belong to the index set I if all smaller grids $\underline{j}$ belong to it. The combination coefficients for a dimension adaptive combination

technique, which are related to the "inclusion/exclusion" principle from combinatorics, depend only on the index set [20, 26, 27]:

$$f_{\mathsf{I}}^{c}(\underline{x}) := \sum_{\underline{k}\in\mathsf{I}} \left(\sum_{\underline{z}=\underline{0}}^{\underline{1}} (-1)^{|\underline{z}|_1} \cdot \chi^{\mathsf{I}}(\underline{k}+\underline{z}) \right) f_{\underline{k}}(\underline{x}) \tag{33}$$

where χ^{I} is the characteristic function of I defined by

$$\chi^{\mathsf{I}}(\underline{k}) := \begin{cases} 1 & \text{if } \underline{k} \in \mathsf{I}, \\ 0 & \text{otherwise.} \end{cases}$$

Further details on dimension adaptive algorithms and suitable refinement strategies for the sparse combination technique can be found in [17, 19, 21].

3.1 Optimised Combination Technique

As mentioned, the combination technique only gives the same order if the above error expansion exists. In some cases even divergence of the combination technique can be observed [15, 16, 27]. But an optimised combination technique [27] can be used instead to achieve good approximations with a combination technique and especially to avoid the potential divergence. Here the combination coefficients are not fixed, but depend on the underlying problem and the function to be represented. Optimised combination coefficients are in particular relevant for dimension adaptive approaches [17, 19].

For ease of presentation we assume a suitable numbering of the involved spaces from (28) for now. To compute the optimal combination coefficients c_i one minimises the functional

$$J(c_1, \ldots, c_m) = \left\| P_n^s f - \sum_{i=1}^{m} c_i P_i f \right\|^2,$$

where one uses a suitable scalar product and a corresponding orthogonal projection P stemming from the problem under consideration. By $P_n^s f$ we denote the projection into the sparse grid space V_n^s, by $P_i f$ the projection into one of the spaces from (28).

By simple expansion one gets

$$J(c_1, \ldots, c_m) = \sum_{i,j=1}^{m} c_i c_j \langle P_i f, P_j f \rangle - 2 \sum_{i=1}^{m} c_i \| P_i f \|^2 + \| P_n^s f \|^2.$$

While this functional depends on the unknown quantity $P_n^s f$, the location of the minimum of J does not. By differentiating with respect to the combination coefficients c_i and setting each of these derivatives to zero we see that minimising this expression corresponds to finding c_i which have to satisfy

$$\begin{bmatrix} \|P_1 f\|^2 & \cdots & \langle P_1 f, P_m f\rangle \\ \langle P_2 f, P_1 f\rangle & \cdots & \langle P_2 f, P_m f\rangle \\ \vdots & \ddots & \vdots \\ \langle P_m f, P_1 f\rangle & \cdots & \|P_m f\|^2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} \|P_1 f\|^2 \\ \|P_2 f\|^2 \\ \vdots \\ \|P_m f\|^2 \end{bmatrix}.$$

The solution of this small system creates little overhead. However, in general to compute the scalar product $\langle P_{\underline{i}} f, P_{\underline{j}} f\rangle$ of the two projections into the discrete spaces $V_{\underline{i}}$ and $V_{\underline{j}}$ one needs to embed both spaces into the joint space $V_{\underline{k}}$, with $k_t = \max(i_t, j_t)$, into which the partial solutions $P_{\underline{l}} f = f_{\underline{l}}, \underline{l} = \underline{i}, \underline{j}$ have to be interpolated. One easily observes that $V_{\underline{k}}$ is of size $\mathcal{O}(h_n^{-2})$ in the worst case, as opposed to $\mathcal{O}(h_n^{-1})$ for the $V_{\underline{l}}, \underline{l} = \underline{i}, \underline{j}$; an increase in computational complexity thus results, but does not depend on $\overline{d}$. In specific situations the computational complexity can be smaller though [16].

Using these optimal coefficients c_i the combination formula for a sparse grid of level n is now just

$$f_n^c(\underline{x}) := \sum_{q=0}^{d-1} \sum_{|\underline{l}|_1 = n-q} c_{\underline{l}} f_{\underline{l}}(\underline{x}). \tag{34}$$

Finally note that one also can interpret the optimised combination technique as a Galerkin formulation which uses the partial solutions as ansatz functions. That way one can formulate an optimised combination technique for problems where the projection arguments do not hold and are replaced by Galerkin conditions, which for example is the case for eigenvalue problems [18].

References

1. S. Achatz. Higher order sparse grid methods for elliptic partial differential equations with variable coefficients. *Computing*, 71(1):1–15, 2003.
2. K. I. Babenko. Approximation of periodic functions of many variables by trigonometric polynomials. *Dokl. Akad. Nauk SSSR*, 132:247–250, 1960. Russian, Engl. Transl.: Soviet Math. Dokl. 1:513–516, 1960.
3. R. Balder. *Adaptive Verfahren für elliptische und parabolische Differentialgleichungen auf dünnen Gittern*. Dissertation, Technische Universität München, 1994.
4. G. Baszenski, F.-J. Delvos, and S. Jester. Blending approximations with sine functions. In D. Braess, editor, *Numerical Methods in Approximation Theory IX*, ISNM 105, pages 1–19. Birkhäuser, Basel, 1992.
5. H.-J. Bungartz. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung*. Dissertation, Institut für Informatik, Technische Universität München, 1992.

6. H.-J. Bungartz. *Finite Elements of Higher Order on Sparse Grids*. Habilitation, Institut für Informatik, Technische Universität München and Shaker Verlag, Aachen, 1998.
7. H.-J. Bungartz and M. Griebel. A note on the complexity of solving Poisson's equation for spaces of bounded mixed derivatives. *J. Complexity*, 15:167–199, 1999. also as Report No 524, SFB 256, Univ. Bonn, 1997.
8. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.
9. H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Pointwise convergence of the combination technique for the Laplace equation. *East-West J. Numer. Math.*, 2:21–45, 1994.
10. H.-J. Bungartz, M. Griebel, and U. Rüde. Extrapolation, combination, and sparse grid techniques for elliptic boundary value problems. *Comput. Methods Appl. Mech. Eng.*, 116: 243–252, 1994. Also in C. Bernardi and Y. Maday, Editoren, International conference on spectral and high order methods, ICOSAHOM 92. Elsevier, 1992.
11. F.-J. Delvos. d-variate Boolean interpolation. *J. Approx. Theory*, 34:99–114, 1982.
12. F.-J. Delvos and W. Schempp. *Boolean Methods in Interpolation and Approximation*. Pitman Research Notes in Mathematics Series 230. Longman Scientific & Technical, Harlow, 1989.
13. G. Faber. Über stetige Funktionen. *Mathematische Annalen*, 66:81–94, 1909.
14. C. Feuersänger. *Sparse Grid Methods for Higher Dimensional Approximation*. Dissertation, Institut für Numerische Simulation, Universität Bonn, Sept. 2010.
15. J. Garcke. *Maschinelles Lernen durch Funktionsrekonstruktion mit verallgemeinerten dünnen Gittern*. Doktorarbeit, Institut für Numerische Simulation, Universität Bonn, 2004.
16. J. Garcke. Regression with the optimised combination technique. In W. Cohen and A. Moore, editors, *Proceedings of the 23rd ICML '06*, pages 321–328, New York, NY, USA, 2006. ACM Press.
17. J. Garcke. A dimension adaptive sparse grid combination technique for machine learning. In W. Read, J. W. Larson, and A. J. Roberts, editors, *Proceedings of the 13th Biennial Computational Techniques and Applications Conference, CTAC-2006*, volume 48 of *ANZIAM J.*, pages C725–C740, 2007.
18. J. Garcke. An optimised sparse grid combination technique for eigenproblems. *PAMM*, 7(1):1022301–1022302, 2007.
19. J. Garcke. A dimension adaptive combination technique using localised adaptation criteria. In H. G. Bock, X. P. Hoang, R. Rannacher, and J. P. Schlöder, editors, *Modeling, Simulation and Optimization of Complex Processes*, pages 115–125. Springer Berlin Heidelberg, 2012.
20. T. Gerstner and M. Griebel. Numerical Integration using Sparse Grids. *Numer. Algorithms*, 18:209–232, 1998.
21. T. Gerstner and M. Griebel. Dimension–Adaptive Tensor–Product Quadrature. *Computing*, 71(1):65–87, 2003.
22. M. Griebel. A parallelizable and vectorizable multi-level algorithm on sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for partial differential equations, Notes on Numerical Fluid Mechanics*, volume 31, pages 94–100. Vieweg Verlag, Braunschweig, 1991. also as SFB Bericht, 342/20/90 A, Institut für Informatik, TU München, 1990.
23. M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.
24. M. Griebel and S. Knapek. Optimized tensor-product approximation spaces. *Constructive Approximation*, 16(4):525–540, 2000.
25. M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens, editors, *Iterative Methods in Linear Algebra*, pages 263–281. IMACS, Elsevier, North Holland, 1992.
26. M. Hegland. Adaptive sparse grids. In K. Burrage and R. B. Sidje, editors, *Proc. of 10th CTAC-2001*, volume 44 of *ANZIAM J.*, pages C335–C353, 2003.
27. M. Hegland, J. Garcke, and V. Challis. The combination technique and some generalisations. *Linear Algebra and its Applications*, 420(2–3):249–275, 2007.
28. R. Hochmuth. *Wavelet Bases in numerical Analysis and Restrictal Nonlinear Approximation*. Habilitation, Freie Universität Berlin, 1999.

29. R. Hochmuth, S. Knapek, and G. Zumbusch. Tensor products of Sobolev spaces and applications. Technical Report 685, SFB 256, Univ. Bonn, 2000.
30. S. Knapek. *Approximation und Kompression mit Tensorprodukt-Multiskalenräumen*. Doktorarbeit, Universität Bonn, April 2000.
31. C. B. Liem, T. Lü, and T. M. Shih. *The Splitting Extrapolation Method*. World Scientific, Singapore, 1995.
32. J. Noordmans and P. Hemker. Application of an adaptive sparse grid technique to a model singular perturbation problem. *Computing*, 65:357–378, 2000.
33. C. Pflaum and A. Zhou. Error analysis of the combination technique. *Numer. Math.*, 84(2): 327–350, 1999.
34. D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München, Aug. 2010. Dissertation.
35. D. Pflüger, B. Peherstorfer, and H.-J. Bungartz. Spatially adaptive sparse grids for high-dimensional data-driven problems. *Journal of Complexity*, 26(5):508–522, 2010.
36. C. Reisinger. *Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2004. in Vorbereitung.
37. S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk SSSR*, 148:1042–1043, 1963. Russian, Engl. Transl.: Soviet Math. Dokl. 4:240–243, 1963.
38. V. N. Temlyakov. Approximation of functions with bounded mixed derivative. *Proc. Steklov Inst. Math.*, 1, 1989.
39. V. N. Temlyakov. *Approximation of Periodic Functions*. Nova Science, New York, 1993.
40. V. N. Temlyakov. On approximate recovery of functions with bounded mixed derivative. *J. Complexity*, 9:41–59, 1993.
41. G. Wahba. *Spline models for observational data*, volume 59 of *Series in Applied Mathematics*. SIAM, Philadelphia, 1990.
42. H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.
43. H. Yserentant. Hierarchical bases. In J. R. E. O'Malley et al., editors, *Proc. ICIAM'91*, Philadelphia, 1992. SIAM.
44. A. Zeiser. Fast Matrix-Vector Multiplication in the Sparse-Grid Galerkin Method. *Journal of Scientific Computing*, 47(3):328–346, Nov. 2010.
45. C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990*, volume 31 of *Notes on Num. Fluid Mech.*, pages 241–251. Vieweg-Verlag, 1991.

Intraday Foreign Exchange Rate Forecasting Using Sparse Grids

Jochen Garcke, Thomas Gerstner, and Michael Griebel

Abstract We present a machine learning approach using the sparse grid combination technique for the forecasting of intraday foreign exchange (FX) rates. The aim is to learn the impact of trading rules used by technical analysts just from the empirical behaviour of the market. To this end, the problem of analyzing a time series of transaction tick data is transformed by delay embedding into a D-dimensional regression problem using derived measurements from several different exchange rates. Then, a grid-based approach is used to discretize the resulting high-dimensional feature space. To cope with the curse of dimensionality we employ sparse grids in the form of the combination technique. Here, the problem is discretized and solved for a collection of conventional grids. The sparse grid solution is then obtained by linear combination of the solutions on these grids. We give the results of this approach to FX forecasting using real historical exchange data of the Euro, the US dollar, the Japanese Yen, the Swiss Franc and the British Pound from 2001 to 2005.

J. Garcke (✉)
Institut für Numerische Simulation, Universität Bonn, Wegelerstr. 6, D-53115, Bonn, Germany

Fraunhofer SCAI, Schloss Birlinghoven, D-53754, Sankt Augustin, Germany
e-mail: garcke@ins.uni-bonn.de

T. Gerstner
Institut für Mathematik, Johann Wolfgang Goethe-Universität, D-60054, Frankfurt am Main, Germany
e-mail: gerstner@math.uni-frankfurt.de

M. Griebel
Institut für Numerische Simulation, Universität Bonn, Wegelerstr. 6, D-53115, Bonn, Germany
e-mail: griebel@ins.uni-bonn.de

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_4,

1 Introduction

After the breakdown of the Bretton Woods system of fixed exchange rates in 1973, the forecasting of exchange rates became more and more important. Nowadays the amounts traded in the foreign exchange market are over three trillion US dollars every day. With the emergence of the Euro in 1999 as a second world currency which rivals the US dollar [29], the forecasting of FX rates got more necessary but also more complicated. Besides incorporating basic economic and financial news, reports by market analysts, and opinions expressed in financial journals, many investors and traders employ in their decision process (besides their guts feelings) technical tools to analyze the transaction data. These data consist of huge amounts of quoted exchange rates, where each new transaction generates a so-called *tick*, often many within a second.

Several academic studies have evaluated the profitability of trading strategies based on daily or weekly data. However, such investigations of trading in the foreign exchange market have not been consistent with the practice of technical analysis [24, 27]. Technical traders transact at a high frequency and aim to finish the trading day with a net open position of zero. In surveys of participants in the foreign exchange market, 90 % of respondents use technical analysis in intraday trading [25,32], whereas 25–30 % of all traders base most of their trades on technical signals [4]. Evidence was presented in [28] that so-called support and resistance levels, i.e. points at which an exchange rate trend is likely to be suspended or reversed, indeed help to predict intraday trend interruptions. On the other hand, the authors of [5] examined filter rules supplied by technical analysts and did not find evidence for profit. Nevertheless, the existence of profit-making rules might be explained from a statistical perspective by the more complex, nonlinear dynamics of foreign exchange rates as observed in [18]. In [6] two computational learning strategies, reinforcement learning and genetic programming, were compared to two simpler methods, a Markov decision problem and a simple heuristic. These methods were able to generate profits in intraday trading when transaction costs were zero, although none produced significant profits for realistic values. In [27], with the use of a genetic program and an optimized linear forecasting model with realistic transaction costs, no evidence of excess returns was found, but some remarkable stable patterns in the data were nevertheless discovered. In [37] multiple foreign exchange rates were used simultaneously in connection with neural networks. There, better performance was observed using multiple exchange rates than in a separate analysis of each single exchange rate.

In this paper we tackle the problem of forecasting intraday exchange rates by transforming it into a machine learning regression problem [9, 12, 15]. The idea behind this approach is that the market will behave similarly in similar situations due to the use of technical analysis by many market participants. In particular the trading rules employed by traders result in a behaviour of the time series which does not follow a pure Markovian process. The machine learning algorithm now attempts to learn the impact of the trading rules just from the empirical behaviour of the market.

To this end, the time series of transaction tick data is cast into a number of data points in a D-dimensional feature space together with a label. The label represents the difference between the exchange rates of the current time and a fixed time step into the future. Therefore one obtains a regression problem. Here, the D features are derived from a delay embedding of the data [26,31]. For example, approximations of first or second derivatives at each time step of the exchange rate under consideration can be used. Furthermore, we investigate especially the additional use of tick data from further exchange rates to improve the quality of the prediction of one exchange rate. Note that although the technical elements are present in this domain, the foreign exchange market is still one of the most challenging forecasting domains.

Delay embedding is a powerful tool to analyze dynamical systems. Taken's theorem [31] gives the conditions under which a chaotic dynamical system can be reconstructed from a sequence of observations. In essence, it states that if the state space of the dynamical system is a k-dimensional manifold, then it can be embedded in $(2k + 1)$-dimensional Euclidean space using the $2k + 1$ delay values $f(t), f(t - \tau), f(t - 2\tau), \ldots, f(t - 2k\tau)$. Here, heuristic computational methods, such as the Grassberger-Procaccia algorithm [16] can be used to estimate the embedding dimension k.

In this work we apply our recent approach for data mining problems [11,12]. It is based on the regularization network formulation [15] and uses a grid, independent of the data positions, with associated local ansatz functions to discretize the feature space. This is similar to the numerical treatment of partial differential equations with finite elements. To avoid the curse of dimensionality, at least to some extent, a so-called sparse grid [3, 36] is used in the form of the combination technique [17]. The approach is based on a hierarchical subspace splitting and a sparse tensor product decomposition of the underlying function space. To this end, the regularized regression problem is discretized and solved on a certain sequence of conventional grids. The sparse grid solution is then obtained by a linear combination of the solutions from the different grids. It turns out that this method scales only linearly with the number of data points to be treated [12]. Thus, this approach is well suited for machine learning applications where the dimension D of the feature space is moderately high, but the amount of data is very large, which is the case in FX forecasting. This is in contrast to support vector machines and related kernel based techniques whose cost scale quadratically or even cubically with the number of data points (but allow to deal with very high-dimensional feature spaces). We show in this article that sparse grid regression can indeed be a useful tool for intraday foreign exchange rate forecasting using real transaction tick data and we present realizable and practical trading strategies. Thereby, we achieve prediction accuracies of almost 60 %, profits of up to 25 % of the maximum attainable profit and we measure average revenues per transaction larger than typical transaction costs.

The remainder of this article is organized as follows: In Sect. 2 we describe how we transform the problem of forecasting a foreign exchange rate into a data mining problem via delay embedding and discuss the resulting regularized regression problem. Section 3 gives the basics of the sparse grid combination technique, our employed numerical algorithm. Then, in Sect. 4, we give the results for our new

approach using real historical exchange data of the Euro, the US dollar, the Japanese Yen, the Swiss Franc and the British Pound from the years 2001 to 2005 and compare its performance to a conventional strategy [2]. Here, we also comment on tradeable strategies and transaction costs. Finally, Sect. 5 gives some concluding remarks.

2 Exchange Rate Forecasting as a Data Mining Problem

We show how the historical intraday exchange rate data are given and discuss how we can convert the FX forecast problem into a regularized least squares regression problem in a high-dimensional space by delay embedding.

2.1 Input Data

Foreign exchange rate tick data consist of the bid and ask quotes of market participants recorded by electronic transaction systems. Note that the tick data are not the real transaction prices for this market, but only the quotes at which market participants want to trade. This results in a small uncertainty in the data. Nevertheless, exchange rate data is presented in this form by most financial news agencies such as Reuters or Bloomberg, e.g. Such historical data are collected and sold by these agencies and other data vendors. For example, the database of Olsen Data has recorded in the year 2002 more than five million ticks for the EUR/USD exchange rate—the most heavily traded currency pair—which gives about 20,000 ticks per business day. The bid and ask prices are typically converted to midpoint prices: $\frac{1}{2}$(bid price + ask price). Note that the spread, that is the difference between bid and ask prices, has to be included in the analysis of the performance at some point in order to assess the expected trading efficiency of a forecasting tool.

In the following we assume that for each tick we have the date, time and one exchange rate value, the midpoint. The raw data of each considered currency pair therefore looks like

```
09.06.2002 09:18:54 0.95595
09.06.2002 09:18:55 0.95615
09.06.2002 09:18:58 0.95585
09.06.2002 09:18:59 0.95605
09.06.2002 09:19:11 0.95689.
```

Now the raw tick data are interpolated to equidistant points in time with a fixed distance of τ. Data from the future cannot be used, therefore the value at the latest raw tick is employed as the exchange rate at these points, i.e. piecewise constant upwind interpolation is applied. If the latest raw tick is more than τ in the past, which means it is the one used for the interpolated tick at the position before, the

exchange rate is set to *nodata*. Furthermore, for all currency pairs the same positions in time are taken. Some data providers already offer such mapped data in addition to the raw data. This way, given J data points from R currency pairs, the input data for exchange rate forecasting has the form

$$\{t_j, f_r(t_j)\} \text{ for } j = 1, \ldots, J \text{ and } r = 1, \ldots, R.$$

Here, t_j denotes the j-th point in time, f_r denotes the exchange rate of the r-th currency pair and $t_{j+1} = t_j + \tau$. Note here that, for reasons of simplicity, we furthermore assume that the nominal differences between the interest rates for the currencies are constant and therefore do not need to be taken into account.

2.2 Delay Embedding into a Feature Space

Based on these interpolated historical input data consisting of $J \cdot R$ data points we now want to predict the value or trend of the exchange rate of the first currency pair f_1. Given a point in time t_j we want to forecast the trend for f_1 at some time $t_j + \hat{k}\tau$ in the future, where $\hat{k}$ denotes the number of ticks considered. To this end, we convert the given series of transaction information through to time t_j into data in a D-dimensional feature space, also called attribute space, which is supposed to describe the market situation at time t_j. The D-dimensional vector in feature space, where D will be made precise in the following, is put together by delay embedding the given tick data (see, for example, [7, 21, 23]). For each exchange rate f_r we consider a fixed number K of delayed values

$$f_r(t_j), f_r(t_j - \tau), f_r(t_j - 2\tau), \ldots, f_r(t_j - (K-1)\tau),$$

where K defines our time horizon $[t_j - (K-1)\tau, t_j]$ backward in time. The resulting $R \cdot K$ delayed values could be directly used to give the D-dimensional feature space with $f_1(t)$ being the first coordinate, $f_1(t - \tau)$ the second, and so on up to $f_R(t - (K-1)\tau)$ being the $(R \cdot K)$-th coordinate.

Note that this is not the only way of delay embedding the data for time t_j. Instead of directly employing the exchange rates, (discrete) first derivatives $f'_{r,k}(t_j) := (f_r(t_j) - f_r(t_j - k\tau))/k\tau$ with $k = 1, \ldots, K-1$ can be used in our backward time horizon yielding $K-1$ coordinates for each exchange rate and $R(K-1)$ coordinates in total. Normalized first derivatives

$$\tilde{f}'_{r,k}(t_j) := \frac{f_r(t_j) - f_r(t_j - k\tau)}{k\tau f_r(t_j - k\tau)}$$

can be considered as well, this takes the assumption into account that trading strategies look for relative changes in the market and not absolute ones. Alternatively, a combination of exchange rates, first derivatives, higher order derivatives

or statistically derived values such as variances or frequencies can be employed as attributes. Note that the actual use of a given feature at all time positions of our backward time horizon of size K, e.g. all K values of the exchange rates or all $K-1$ values of the first derivative, is usually not necessary. A suitable selection from the possible time positions of a given attribute in the time horizon $[t_j - (K-1)\tau, t_j]$, or even only one, can be enough in many situations.

In any case, the number of features obtained by the delay embedding can easily grow large. Therefore, the number K of delay values, that is the size of our backward time horizon, and the total number of derived attributes D have to be chosen properly from the large number of possible embedding strategies. A good choice of such derived attributes and their parameters is non-trivial and has to be determined by careful experiments and suitable assumptions on the behaviour of the market.

In general, the transformation into feature space, i.e. the space of the embedding, for a given point in time t_j is an operator $T : \mathbb{R}^{R \cdot K} \to \mathbb{R}^D$

$$\begin{aligned}\underline{x}(t_j) = T\big(&f_1(t_j),\ \ldots, f_1(t_j - (K-1)\tau), f_2(t_j), \ldots,\\ &\ldots, f_2(t_j - (K-1)\tau), \ldots, f_R(t_j), \ldots, f_R(t_j - (K-1)\tau)\big)\end{aligned}$$

with the feature vector $\underline{x}(t_j) = (x_1, \ldots, x_D) \in \mathbb{R}^D$, where the single features $x_d, d = 1, \ldots, D$, are any of the derived values mentioned.

As the response variable in the machine learning process we employ the normalized difference between the exchange rate f_1 at the current time t_j and at some time $t_j + \hat{k}\tau$ in the future, i.e.

$$y(t_j) = \frac{f_1(t_j + \hat{k}\tau) - f_1(t_j)}{f_1(t_j)}.$$

This will give a regression problem later on. If one is only interested in the trend, the sign of $y(t_j)$ can be used as the response variable which will result in a classification problem.

This transformation of the transaction data into a D-dimensional feature vector can be applied at $J - (K-1) - \hat{k}$ different time points t_j over the whole data series, since at the beginning and end of the given time series data one has to allow for the time frame of the delay embedding and prediction, respectively. Altogether, the application of such an embedding transformation and the evaluation of the associated forecast values over the whole time series results in a data set of the form

$$S = \{(\underline{x}_m, y_m) \in \mathbb{R}^D \times \mathbb{R}\}_{m=1}^{J-(K-1)-\hat{k}}, \tag{1}$$

with $\underline{x}_m = \underline{x}(t_{m+K-1})$ and $y_m = y(t_{m+K-1})$.

This dataset can now be used by any machine learning algorithm, such as neural networks, multivariate adaptive regression splines or support vector machines, to construct a function $u : \Omega \subset \mathbb{R}^D \to \mathbb{R}$ which describes the relationship between the features $\underline{x}$, i.e. the market situation, and the response y, i.e. the trend. This

relationship can then be evaluated at a future time t by using the same operator T to transform its corresponding transaction data into a D-dimensional feature vector $\underline{x}$ which describes this new market situation. Since we assume that the market behaves similarly in similar situations, the evaluation of the reconstructed continuous function u in such a new market situation $\underline{x}$ is supposed to yield a good prediction $u(\underline{x})$.

2.3 Regularized Least Squares Regression

In the following we formulate the scattered data approximation problem in D-dimensional space by means of a regularization network approach [8, 15]. As stated above, we assume that the relation between $\underline{x}$ and y in the data set (1) can be described by an unknown function $u : \Omega \subset \mathbb{R}^D \to \mathbb{R}$ which belongs to some space V of functions defined over $\mathbb{R}^D$. The aim is now to recover the function u from the given data S, of some size M, with e.g. $M := J - (K-1) - \hat{k}$, as good as possible. A simple least squares fit of the data would surely result in an ill-posed problem. To obtain a well-posed, uniquely solvable problem, we use regularization theory and impose additional smoothness constraints on the solution of the approximation problem. In our approach this results in the variational problem

$$\min_{u \in V} R(u)$$

with

$$R(u) = \frac{1}{M} \sum_{m=1}^{M} (u(\underline{x}_m) - y_m)^2 + \lambda \|Gu\|_{L_2}^2. \tag{2}$$

Here, the mean squared error enforces closeness of u to the data, the regularization term defined by the operator G enforces the smoothness of u, and the regularization parameter λ balances these two terms. Other error measurements can also be suitable. Further details can be found in [8, 12, 33]. Note that there is a close relation to reproducing kernel Hilbert spaces and kernel methods where a kernel is associated to the regularization operator G, see also [30, 35].

3 Sparse Grid Discretization

In order to compute a numerical solution of (2), we restrict the problem to a finite dimensional subspace $V_N \subset V$ of dimension $\dim V_N = N$. Common data mining methods such as radial basis approaches or support vector machines work with global ansatz functions associated to data points which leads to $N = M$. These

methods allow to deal with very high-dimensional feature spaces, but typically scale at least quadratically or even cubically with the number of data points and, thus, cannot be applied to the huge data sets prevalent in foreign exchange rate prediction. Instead, we use grid based local basis functions, i.e. finite elements, in the feature space, similarly to the numerical treatment of partial differential equations. With such a basis $\{\varphi_n\}_{n=1}^N$ of the function space V_N we can approximately represent the regressor u as

$$u_N(\underline{x}) = \sum_{n=1}^{N} \alpha_n \varphi_n(\underline{x}). \tag{3}$$

Note that the restriction to a suitably chosen finite-dimensional subspace involves some additional regularization (regularization by discretization [20]) which depends on the choice of V_N. In the following, we simply choose $G = \nabla$ as the smoothing operator. Although this does not result in a well-posed problem in an infinite dimensional function space its use is reasonable in the discrete function space $V_N, N < \infty$, see [12, 13].

Now we plug (3) into (2). After differentiation with respect to the coefficients α_j, the necessary condition for a minimum of $R(u_N)$ gives the linear system of equations [12]

$$(\lambda \mathcal{C} + \mathcal{B} \cdot \mathcal{B}^T)\underline{\alpha} = \mathcal{B}\underline{y}. \tag{4}$$

Here $\mathcal{C}$ is a square $N \times N$ matrix with entries $\mathcal{C}_{n,n'} = M \cdot (\nabla\varphi_n, \nabla\varphi_{n'})_{L_2}$ for $n, n' = 1, \ldots, N$, and $\mathcal{B}$ is a rectangular $N \times M$ matrix with entries $\mathcal{B}_{n,m} = \varphi_n(\underline{x}_m), m = 1, \ldots, M, n = 1, \ldots, N$. The vector $\underline{y}$ contains the response labels $y_m, m = 1, \ldots, M$. The unknown vector $\underline{\alpha}$ contains the degrees of freedom α_n and has length N. A solution of this linear system then gives the vector $\underline{\alpha}$ which spans the approximation $u_N(\underline{x})$ with (3).

3.1 Sparse Grid Combination Technique

Up to now we have not yet been specific what finite-dimensional subspace V_N and what type of basis functions $\{\varphi_n\}_{n=1}^N$ we want to choose. If uniform grids were used here, we would immediately encounter the curse of dimensionality and could not treat higher dimensional problems. Instead we employ sparse grid subspaces as introduced in [3, 36] to discretize and solve the regularization problem (2), see also [12]. This discretization approach is based on a sparse tensor product decomposition of the underlying function space. In the following we describe the relevant basic ideas, for details see [3, 9, 12, 36].

To be precise, we apply sparse grids in form of the combination technique [17]. There, we discretize and solve the problem on a suitable sequence of small and in general anisotropic grids $\Omega_{\underline{l}}$ of level $\underline{l} = (l_1, \ldots, l_D)$, which have different but uniform mesh sizes $h_d = 2^{-l_d}, d = 1, \ldots, D$, in each coordinate direction. The points of a given grid $\Omega_{\underline{l}}$ are numbered using the multi-index $\underline{i} = (i_1, \ldots, i_D)$ with

$i_d \in \{0, \ldots, 2^{l_d}\}$ for $d = 1, \ldots, D$. For ease of presentation, we assume the domain $\Omega = [0, 1]^D$ here and in the following, which can be always achieved by a proper rescaling of the data.

A finite element approach with piecewise multilinear functions

$$\phi_{\underline{l},\underline{i}}(\underline{x}) := \prod_{d=1}^{D} \phi_{l_d,i_d}(x_d), \quad i_d = 0, \ldots, 2^{l_d}, \tag{5}$$

on each grid $\Omega_{\underline{l}}$, where the one-dimensional basis functions $\phi_{l_d,i_d}(x_d)$ are the so-called hat functions

$$\phi_{l_d,i_d}(x_d) = \begin{cases} 1 - |\frac{x_d}{h_{l_d}} - i_d|, & x_d \in [(i_d - 1)h_{l_d}, (i_d + 1)h_{l_d}] \\ 0, & \text{otherwise,} \end{cases}$$

results in the discrete function space $V_{\underline{l}} := \text{span}\{\phi_{\underline{l},\underline{i}}, i_d = 0, \ldots, 2^{l_d}, d = 1, \ldots, D\}$ on grid $\Omega_{\underline{l}}$. A function $u_{\underline{l}} \in V_{\underline{l}}$ is then represented as

$$u_{\underline{l}}(\underline{x}) = \sum_{i_1=0}^{2^{l_1}} \ldots \sum_{i_D=0}^{2^{l_D}} \alpha_{\underline{l},\underline{i}} \phi_{\underline{l},\underline{i}}(\underline{x}).$$

Each multilinear function $\phi_{\underline{l},\underline{i}}(\underline{x})$ equals one at the grid point $\underline{i}$ and is zero at all other points of grid $\Omega_{\underline{l}}$. Its support, i.e. the domain where the function is non-zero, is $\otimes_{d=1}^{D}[(i_d - 1)h_{l_d}, (i_d + 1)h_{l_d}]$.

To obtain a solution in the sparse grid space V_L^s of level L the combination technique considers all grids $\Omega_{\underline{l}}$ with

$$l_1 + \ldots + l_D = L + (D - 1) - q, \quad q = 0, \ldots, D - 1, \quad l_q > 0, \tag{6}$$

see also Fig. 1 for an example in two dimensions. One gets an associated system of linear equations (4) for each of the involved grids $\Omega_{\underline{l}}$, which we currently solve by a diagonally preconditioned conjugate gradient algorithm.

The combination technique [17] now linearly combines the resulting discrete solutions $u_{\underline{l}}(\underline{x})$ from the grids $\Omega_{\underline{l}}$ according to the formula

$$u_L^c(\underline{x}) := \sum_{q=0}^{D-1} (-1)^q \binom{D-1}{q} \sum_{|\underline{l}|_1 = L + (D-1) - q} u_{\underline{l}}(\underline{x}). \tag{7}$$

The resulting function u_L^c lives in the sparse grid space V_L^s which has dimension $N = \dim V_L^s = \mathcal{O}(h_L^{-1}(\log(h_L^{-1}))^{D-1})$, see [3]. It therefore depends on the dimension D to a much smaller degree than a function on the corresponding uniform grid $\Omega_{(L,\ldots,L)}$ whose number of degrees of freedom is $\mathcal{O}(h_L^{-D})$. Note that for the approximation of a function u by a sparse grid function $u_L^c \in V_L^s$ the error relation

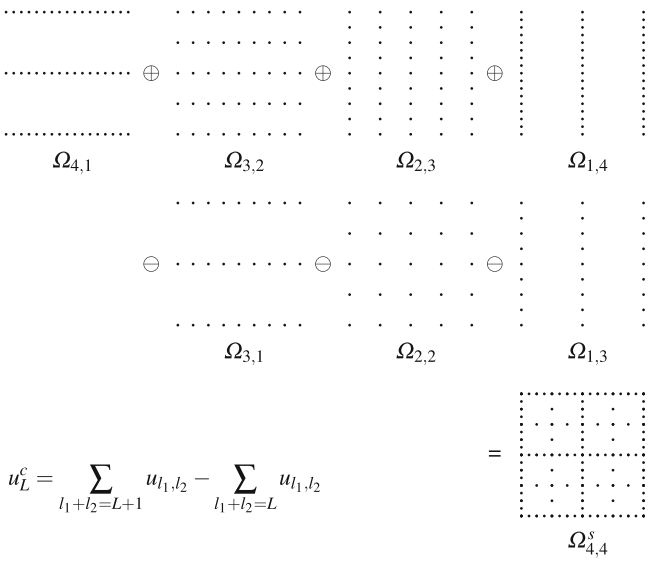


Fig. 1 Grids employed by the combination technique of level $L = 4$ in two dimensions

$$||u - u_L^c||_{L_2} = \mathcal{O}(h_L^2 \log(h_L^{-1})^{D-1})$$

holds, provided that u fulfills certain smoothness requirements which involve bounded second mixed derivatives [3]. The combination technique can be further generalized [9, 19] to allow problem dependent coefficients.

Note that we never explicitly assemble the function u_L^c but instead keep the solutions $u_{\underline{l}}$ which arise in the combination technique (6).

If we now want to evaluate the solution at a newly given data point $\underline{\tilde{x}}$ by

$$\tilde{y} := u_L^c(\underline{\tilde{x}}),$$

we just form the combination of the associated point values $u_{\underline{l}}(\underline{\tilde{x}})$ according to (7). The cost of such an evaluation is of the order $\mathcal{O}(L^{D-1})$.

4 Numerical Results

We now present results for the prediction of intraday foreign exchange rates with our sparse grid combination technique. Our aim is to forecast the EUR/USD exchange rate. First, we use just the EUR/USD exchange rate time series as input and employ a delay embedding of this single time series. Here we compare the performance with that of a traditional trading strategy using only EUR/USD information. We then also take the other exchange rates into account and show the corresponding results.

Table 1 Total and missing number of ticks, number of gaps, and maximum and average gap length of the input data

Exchange rate	Total ticks	Missing ticks	Number of gaps	Max. gap length	Avg. gap length
EUR/USD	701,280	168,952	6,403	879	26
USD/CHF	701,280	171,015	6,192	920	27
USD/JPY	701,280	184,264	4,144	911	44
GBP/USD	701,280	185,442	5,278	912	35

Furthermore, we present a strategy which involves trading on strong signals only to cope with transaction costs. Based on that, we finally present a trading strategy which in addition reduces the amount of invested capital. Moreover, we compare these approaches and demonstrate their properties in numerical experiments.

4.1 Experimental Data

The data were obtained from Olsen Data, a commercial data provider. In the following, we employ the exchange rates from 01.08.2001 to 28.07.2005 between EUR/USD (denoted by €), GBP/USD (£), USD/JPY (¥) and USD/CHF (Fr.). To represent a specific currency pairing we will use the above symbols instead of f_r in the following. For this data set the data provider mapped the recorded raw intraday tick data by piecewise constant interpolation to values $f_r(t_j)$ at equidistant points in time which are $\tau = 3\,\text{min}$ apart. No data is generated if in the time interval $[t_j - \tau, t_j]$ no raw tick is present. Due to this, the data set contains a multitude of gaps, which can be large when only sparse trading takes place, for example over weekends and holidays. The properties of this input data concerning these gaps is shown in Table 1. Here, the total number of ticks in the time frame would be 701,280 for each currency pair, but between 168,000 and 186,000 ticks are missing due to the above reasons. The number of gaps varies between about 4,000 and 6,000 while the gap length varies between 1 and about 900 with an average of about 30. These characteristics are similar for the four currency pairs.

Note that the trading volumes are not constant during the day. The main trading starts each day in the East-Asian markets with Tokyo and Sydney as centers, then the European market with London and Frankfurt dominates, while the main trading activity takes place during the overlap of the European business hours and the later starting American market with New York as the hub [18, 22].

For the following experiments with the sparse grid regression approach the associated input data set S is obtained from the given tick data. Note that the embedding operator T at a time t_j depends on a certain number of delayed data positions between $t_j - (K-1)\tau$ and t_j. Typically not all time positions in the backward time horizon are employed for a given T. Nevertheless, the feature vector at time t_j can only be computed if the data at the positions necessary for T are present, although small data gaps in between these required points are allowed.

Note here that we employ the common practice of restricting the values of outliers to a suitable chosen maximum value. Afterwards we linearly map the derived delay embedded features into $[0, 1]^D$.

In all our experiments we attempt to forecast the change in the EUR/USD exchange rate. The aim of our regression approach is to predict the relative rate difference $y(t_j) = (€(t_j + \hat{k}\tau) - €(t_j))/€(t_j)$ at $\hat{k}$ steps into the future (*future step*) in comparison to the current time. Such a forecast is often also called (trading) signal.

For the experiments we separate the available data into training data (90 %) and test data (10 %). This split is done on the time axis, otherwise a bias from the time frame during which the prediction performance is evaluated might be introduced. On the training data we perform three-fold cross-validation (again splitting in time) to find good values for the level parameter L from (7) and the regularization parameter λ from (2) of our regression approach. To this end, the training data set is split into three equal parts. Two parts are used in turn as the learning set and the quality of the regressor (see the following section) is evaluated on the remaining part for varying L and λ. The pair of values of L and λ which performs best in the average of all three splittings is then taken as the optimum and is used for the forecast and final evaluation on the 10 % remaining newest test data.

4.2 Quality Assessment

To judge the quality of the predictions by our sparse grid combination technique for a given number M of data we use the so-called realized potential

$$rp := cp/mcp$$

as the main measurement. Here cp is the cumulative profit

$$cp := \sum_{m=1}^{M} \frac{\operatorname{sign}(u_L^c(\underline{x}_m)) \cdot (f_1(t_m + \hat{k}\tau) - f_1(t_m))}{f_1(t_m)},$$

i.e. the sum of the actual gain or loss in the exchange rate realized by trading at the M time steps according to the forecast of the method, while mcp is the maximum possible cumulative profit

$$mcp := \sum_{m=1}^{M} \frac{|f_1(t_m + \hat{k}\tau) - f_1(t_m)|}{f_1(t_m)},$$

i.e. the gain when the exchange rate would have been predicted correctly for each trade. For example $M = J - (K - 1) - \hat{k}$ if we considered the whole training data from Sect. 2.2. Note that these measurements also take the amplitude of the

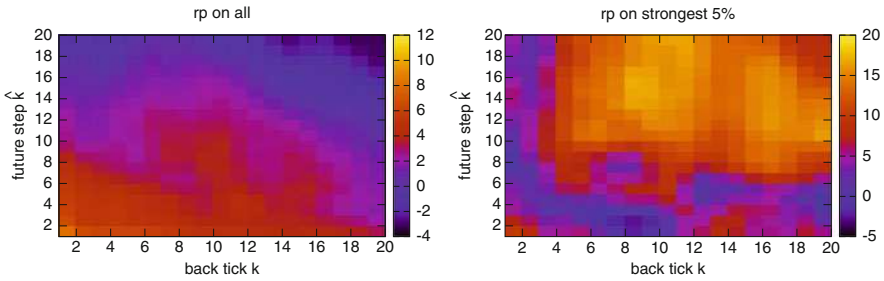


Fig. 2 Realized potential rp for the currency pair EUR/USD for all predictions (*left*) and for the 5 % ticks with the strongest predictions (*right*), $L = 4$ and $\lambda = 0.0001$

potential gain or loss into account. According to practitioners, a forecasting tool which achieves a realized potential rp of 20 % starts to become useful.

Furthermore, we give the prediction accuracy pa, often also called hit rate or correctness rate,

$$pa := \frac{\#\{u_L^c(\underline{x}_m) \cdot (f_1(t_m + \hat{k}\tau) - f_1(t_m)) > 0\}_{m=1}^{M}}{\#\{u_L^c(\underline{x}_m) \cdot (f_1(t_m + \hat{k}\tau) - f_1(t_m)) \neq 0\}_{m=1}^{M}}$$

which denotes the percentage of correctly predicted forecasts. Prediction accuracies of more than 55 % are often reported as worthwhile results for investors [1, 34]. So far, all these measurements do not yet directly take transaction costs into account. We will address this aspect later in Sect. 4.5 in more detail.

4.3 Forecasting Using a Single Currency Pair

In a first set of experiments we aim to forecast the EUR/USD exchange rate from the EUR/USD exchange data. We begin with using one feature, the normalized discrete first derivative

$$\tilde{€}'_k = \frac{€(t_j) - €(t_j - k\tau)}{k\tau €(t_j - k\tau)}.$$

Here, *back tick k* is a parameter to be determined as is $\hat{k}$, the time horizon for the forecast into the future. The results of experiments for the prediction of the EUR/USD exchange rate from the first derivative for several values of k and $\hat{k}$ are shown in Fig. 2, where the performance is measured by the realized potential rp. To evaluate the behaviour we consider all data and those data for which stronger signals are predicted. For the latter we here consider for simplification the 5 % ticks for which we obtain the strongest predictions, although in practise not a relative and a posterior threshold, but an absolute and a priori would need to be considered. We observe the best results, in particular taking the performance on the stronger signals

Table 2 Three-fold cross-validation results for the forecast of EUR/USD at $\hat{k} = 15$ ticks into the future and the feature $\tilde{\in}'_9$ for varying refinement level L and regularization parameter λ

1 feature	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{\in}'_9$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$
Level 2	1.89	1.72	50.7	2.10	1.91	50.6	2.40	2.18	50.6	2.40	2.18	50.6
Level 3	3.02	2.76	51.5	2.44	2.22	51.0	2.26	2.05	50.8	2.40	2.18	50.6
Level 4	**3.37**	**3.07**	**51.9**	3.08	2.81	51.6	2.33	2.12	51.0	2.40	2.18	50.6

Table 3 Three-fold cross-validation results for the forecast of EUR/USD at $\hat{k} = 15$ ticks into the future and the features $\tilde{\in}'_9$ and $\tilde{\in}'_4$ for varying refinement level L and regularization parameter λ

2 features	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{\in}'_9, \tilde{\in}'_4$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$	cp	$rp\%$	$pa\%$
Level 2	3.81	3.48	51.9	3.53	3.22	51.6	2.42	2.2	50.6	2.41	2.19	50.6
Level 3	**4.69**	**4.29**	**52.6**	4.57	4.17	52.4	3.56	3.25	51.8	2.25	2.05	50.8
Level 4	4.58	4.18	52.5	4.61	4.21	52.5	4.29	3.91	52.4	2.39	2.17	51.0

into account, for $k = 9$ and $\hat{k} = 15$ which we will use from now on. Since we consider a single currency pair we obtain just a one-dimensional problem here. The combination technique then falls back to conventional discretization.

In Table 2 we give the results of the three-fold cross-validation on the training data for several λ and L. We observe the highest rp for $\lambda = 0.0001$ and $L = 4$. Using these parameters we now learn on all training data. The evaluation on the remaining 10 % test data then results in $cp = 0.741$, $rp = 2.29\,\%$, and $pa = 51.5\,\%$ on 51,056 trades. Of course, such small values for rp and pa are far from being practically relevant. Therefore we investigate in the following different strategies to improve performance. We start by adding an additional feature.

To this end, we consider a two-dimensional regression problem where we take—besides $\tilde{\in}'_9$—the normalized first derivative $\tilde{\in}'_4$ as the second attribute. We choose the value $k = 4$ for the back tick since the combination with the first derivative $\tilde{\in}'_9$ can be interpreted as an approximation to a normalized second derivative

$$\tilde{\in}''_k = \frac{\in(t_j) - 2\in(t_j - k\tau) + \in(t_j - 2k\tau)}{(k\tau)^2 \in(t_j - k\tau)}$$

with $k = 4$. The use of two first derivatives $\tilde{\in}'_9$ and $\tilde{\in}'_4$ captures more information in the data than just the second derivative would.

The results from the three-fold cross-validation on the training data are shown in Table 3. Again we pick the best parameters and thus use $\lambda = 0.0001$ and $L = 3$ for the prediction on the 10 % remaining test data. The additional attribute $\tilde{\in}'_4$ results in a significant improvement of the performance: We achieve $cp = 1.084$, $rp = 3.36\,\%$, and $pa = 52.1\,\%$ on 50,862 trades, see also Table 4 for the comparison with the former experiment using only one feature. In particular we observe that rp grows by about 50 %. Furthermore, we observe that the profit is to

Table 4 Forecast of the EUR/USD at $\hat{k} = 15$ ticks into the future on the 10 % remaining test data using first derivatives of the EUR/USD exchange rate

	Features	Data points	L	λ	Trades	$pa\%$	cp	mcp	$rp\%$
				All ticks					
1 feature	$\widetilde{\in}'_9$	510,553	4	0.0001	51,056	51.5	0.741	32.37	2.29
2 features	$\widetilde{\in}'_9, \widetilde{\in}'_4$	508,616	3	0.0001	50,862	52.1	1.084	32.28	3.36
				Signal $> 1.0_{-4}$					
1 feature	$\widetilde{\in}'_9$	510,553	4	0.0001	460	56.7	0.070	0.650	10.8
2 features	$\widetilde{\in}'_9, \widetilde{\in}'_4$	508,616	3	0.0001	916	58.6	0.291	1.206	24.2

a significant degree in the stronger signals. If we only take predictions into account which indicate an absolute change[1] larger than 1.0_{-4}, we trade on 916 signals and achieve $cp = 0.291$, $rp = 24.2\,\%$ and $pa = 58.6\,\%$, see Table 4. Thus, trading on 1.8 % of the signals generates 26.8 % of the overall profit. In real life applications the threshould in regard to the strong signals would be considered a parameter as well, for simplifications we stick in the following to the value of 1.0_{-4}. Note again that a rp-value of more than 20 % and a pa-value of more than 55 % is often considered practically relevant. Therefore trading on the stronger signals may result in a profitable strategy. Nevertheless, the use of just two features is surely not yet sufficient.

Before we add more features we need to put the performance of our approach into context. To this end, we compare with results achieved by the moving average-oscillator, a widely used technical trading rule [2]. Here buy and sell signals are generated by two moving averages, a long-period average x_l and a short-period average x_s. They are computed according to

$$x_{\{s,l\}}(t_j) = \frac{1}{w_{\{s,l\}}} \sum_{i=0}^{w_{\{s,l\}}-1} \in(t_{j-i}),$$

where the length of the associated time intervals is denoted by w_l and w_s, respectively. To handle small gaps in the data, we allow up to 5 % of the tick data to be missing in a time interval when computing an average, which we scale accordingly in such a case. Furthermore we neglect data positions in our experiments where no information at time t_j, or $t_j + \tau\hat{k}$ is present.

In its simplest form this strategy is expressed as buying (or selling) when the short-period moving average rises above the long-period moving average by an amount larger than a prescribed *band*-parameter b, i.e.

$$x_s(t_j) > b \cdot x_l(t_j)$$

[1] Observe that a change of 1.0_{-4} in our target attribute is roughly the size of a pip (the smallest unit of the quoted price) for EUR/USD.

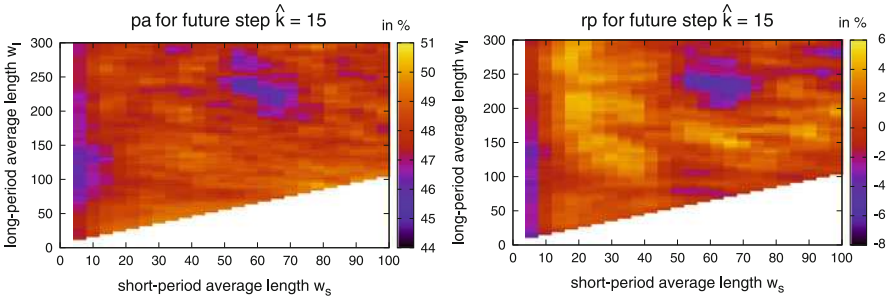


Fig. 3 The prediction accuracy and the realized potential on the training data for the fixed-length moving average trading strategy at $\hat{k} = 15$ ticks into the future and varying lengths of the moving averages

(or falls below it, i.e. $x_s(t_j) < (2 - b) \cdot x_l(t_j)$). This approach is called variable length moving average. The band-parameter b regulates the trading frequency.

This conventional technical trading strategy is typically used for predictions on much longer time frames and did not achieve any profitable results in our experiments. Therefore we considered a different moving average strategy which performed better. Here, a buy signal is generated as above at time t_j when $x_s(t_j) > b \cdot x_l(t_j)$, but a trade only takes place if $x_s(t_{j-1}) \leq b \cdot x_l(t_{j-1})$ holds as well. Such a position is kept for a number $\hat{k}$ of time steps and is then closed. In the same way, sell signals are only acted upon if both conditions (with reversed inequality sign) are fulfilled. Here, several positions might be held at a given time. This rule is called fixed-length moving average (FMA) and stresses that returns should be stable for a certain time period following a crossover of the long- and short-period averages [2].

In Fig. 3 we give the results for EUR/USD of the fixed-length moving average technical rule on the training data. Here, we vary the intervals for both the long-period and the short-period average while using a fixed time horizon in the future of $\hat{k} = 15$. We use the prediction at 15 time steps into the future for two reasons: First we want to be able to compare the results with that of our other experiments which employ the same time horizon, and, second, this value turned out to be a very good choice for the FMA trading rule. As the parameters which achieve the highest rp on the training data we found $w_s = 20$, $w_l = 216$ and $b = 1.000001$.

With these values we obtain $cp = 0.026$, $rp = 9.47\,\%$, and $pa = 47.4\,\%$ on the remaining 10 % test data using a total of 414 trades. Although the rp with FMA is higher in comparison to the results of our approach when trading on all signals in the test data (compare with the first two rows of Table 4), much less trading takes place here. This small amount of trading is the reason for the quite tiny cp for FMA which is almost 40 times smaller. In addition the prediction accuracy for the signals where trading takes place is actually below 50 %. But if we compare the results of FMA with our approach which acts only on the stronger signals $> 1.0_{-4}$ we outperform the FMA strategy on all accounts (compare with the last two rows of Table 4).

Table 5 Three-fold cross-validation results for the forecast of EUR/USD at $\hat{k} = 15$ ticks into the future and features derived from different exchange rates for varying refinement level L and regularization parameter λ. Results for just $\tilde{\text{€}}'_9$ are given in Table 2

2 features	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{\text{€}}'_9, \tilde{\text{Fr.}}'_9$	*cp*	*rp%*	*pa%*	*cp*	*rp%*	*pa%*	*cp*	*rp%*	*pa%*	*cp*	*rp%*	*pa%*
Level 2	4.96	4.56	52.2	4.63	4.25	51.9	2.76	2.53	50.7	2.47	2.27	50.7
Level 3	**4.98**	**4.58**	**52.3**	4.76	4.38	52.2	3.40	3.12	51.2	2.47	2.27	50.7
Level 4	4.57	4.21	52.0	4.89	4.49	52.1	4.02	3.69	51.6	2.51	2.30	50.7
3 features	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{\text{€}}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9$	*cp*	*rp%*	*pa%*	*cp*	*rp%*	*pa%*	*cp*	*rp%*	*pa%*	*cp*	*rp%*	*pa%*
Level 2	4.84	4.49	52.2	4.63	4.29	52.0	3.23	2.98	50.9	2.56	2.36	50.7
Level 3	4.71	4.38	52.3	**4.90**	**4.56**	**52.3**	4.21	3.91	51.8	2.68	2.47	50.7
Level 4	4.28	3.97	52.2	4.84	4.49	52.1	4.60	4.27	51.9	2.74	2.52	50.9
4 features	$\lambda = 0.0001$			$\lambda = 0.001$			$\lambda = 0.01$			$\lambda = 0.1$		
$\tilde{\text{€}}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	*cp*	*rp%*	*pa%*	*cp*	*rp%*	*pa%*	*cp*	*rp%*	*pa%*	*cp*	*rp%*	*pa%*
Level 2	4.26	3.96	52.1	4.40	4.08	51.9	3.24	3.01	51.1	2.61	2.41	50.6
Level 3	4.42	4.11	52.2	**4.61**	**4.29**	**52.2**	4.21	3.92	51.9	2.50	2.32	50.9
Level 4	3.69	3.43	51.9	4.30	4.00	52.1	4.50	4.18	52.0	3.18	2.95	51.3

4.4 Forecasting Using Multiple Currency Pairs

Now we are interested in the improvement of the prediction of the EUR/USD exchange rate if we also take the other currency pairs £, ¥, and Fr. into account. This results in a higher-dimensional regression problem. We employ first derivatives using the same backticks as before for the different currency pairs.[2] Note that the number of input data points decreases slightly when we add further exchange rate pairs since some features cannot be computed any longer due to overlapping gaps in the input data.

For now we only consider the first derivatives for $k = 9$ to observe the impact due to the use of additional currency pairs. According to the best rp we select which of the three candidates $\tilde{\text{Fr.}}'_9, \tilde{£}'_9, \tilde{¥}'_9$ is successively added. For example $\tilde{\text{Fr.}}'_9$ in addition to $\tilde{\text{€}}'_9$ gave the best result using two currency pairs to predict EUR/USD. We then add $\tilde{£}'_9$ before using $\tilde{¥}'_9$. As before, we select the best parameters L and λ for each number of features according to the rp achieved with three-fold cross-validation on the training data, see Table 5. Note that the values of L and λ with the best performance do not vary much in these experiments. This indicates the stability of our parameter selection process.

Using the combination with the best performance in the three-fold cross-validation we then learn on all training data and evaluate on the before unseen test

[2]Different back ticks might result in a better performance, but we restricted our experiments to equal back ticks for reasons of simplicity.

Table 6 Forecast of EUR/USD at $\hat{k} = 15$ ticks into the future on the 10 % remaining test data using one first derivative from multiple currency pairs. Results are for trading on all signals and on signals $> 1.0_{-4}$

	Currencies	Data points	L	λ	Trades	$pa\%$	cp	mcp	$rp\%$
				All ticks					
1 feature	$\tilde{€}'_9$	510,553	4	0.0001	51,056	51.5	0.741	32.37	2.29
2 features	$\tilde{€}'_9, \tilde{\text{Fr.}}'_9$	503,939	3	0.0001	50,394	51.8	1.380	32.08	4.30
3 features	$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9$	495,654	3	0.001	49,566	51.6	1.469	31.76	4.62
4 features	$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	494,238	3	0.001	49,424	51.7	1.478	31.70	4.66
				Signal $> 1.0_{-4}$					
1 feature	$\tilde{€}'_9$	510,553	4	0.0001	460	56.7	0.070	0.650	10.8
2 features	$\tilde{€}'_9, \tilde{\text{Fr.}}'_9$	503,939	3	0.0001	2,318	54.0	0.469	2.504	18.8
3 features	$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9$	495,654	3	0.001	2,379	54.3	0.516	2.566	20.1
4 features	$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	494,238	3	0.001	3,559	53.8	0.614	3.484	17.6

data. The results on the training data are given in Table 6, both for the case of all data and again for the case of absolute values of the signals larger than 1.0_{-4}. Note that the performance on the training data in Table 5 suggests to employ the first two or three attributes. In any case, the use of information from multiple currencies results in a significant improvement of the performance in comparison to just using one attribute derived from the exchange rate to be predicted. The results on the test data given in Table 6 confirm that the fourth attribute $\tilde{¥}'_9$ does not achieve much of an improvement, whereas the additional features $\tilde{\text{Fr.}}'_9, \tilde{£}'_9$ significantly improve both cp and rp. Trading on signals larger than 1.0_{-4} now obtains a pa of up to 56.7 % and, more importantly, $rp = 20.1$ % using three attributes. This clearly shows the potential of our approach. Altogether, we see the gain in performance which can be achieved by a delay embedding of tick data of several currencies into a higher dimensional regression problem while using a first derivative for each exchange rate.

In a second round of experiments we use two first derivatives with back ticks $k = 9$ and $k = 4$ for each exchange rate. We add step-by-step the different currencies in the order of the above experiment from Table 6. To be precise, we use $\tilde{\text{Fr.}}'_9$ before $\tilde{\text{Fr.}}'_4$, but both before $\tilde{£}'_9, \tilde{£}'_4$, etc.[3] We thus obtain a higher dimensional regression problem. Again we look for good values for λ and L via three-fold cross-validation.

In Table 7 we give the results which were achieved on the test data. Note that the numbers obtained on the training data suggest to use the four features $\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4$ only; nevertheless we show the test results with the two additional features $\tilde{£}'_9, \tilde{£}'_4$ as well. Again, the use of information from multiple currencies gives an improvement of the performance in comparison to the use of just the attributes which were derived from the EUR/USD exchange rate. In particular cp grows from one to several currencies. With four features based on two first derivatives for

[3] Note that a different order might result in a different performance.

Table 7 Forecast of EUR/USD at $\hat{k} = 15$ ticks into the future using multiple currency pairs and derivatives on the 10 % remaining test data. Results are for trading on all signals and on signals $> 1.0_{-4}$

	Currencies	Data points	L	λ	Trades	$pa\%$	cp	mcp	$rp\%$
			All ticks						
1 feature	$\widetilde{\in}_9'$	510,553	4	0.0001	51,056	51.5	0.741	32.37	2.29
2 features	$\widetilde{\in}_9', \widetilde{\in}_4'$	508,616	3	0.0001	50,862	52.1	1.084	32.28	3.36
3 features	$\widetilde{\in}_9', \widetilde{\in}_4', \widetilde{\text{Fr.}}_9'$	503,017	2	0.0001	50,300	52.1	1.315	32.03	4.10
4 features	$\widetilde{\in}_9', \widetilde{\in}_4', \widetilde{\text{Fr.}}_9', \widetilde{\text{Fr.}}_4'$	502,243	2	0.001	50,220	52.4	1.536	31.98	4.80
5 features	$\widetilde{\in}_9', \widetilde{\in}_4', \widetilde{\text{Fr.}}_9', \widetilde{\text{Fr.}}_4', \widetilde{£}_9'$	494,975	2	0.001	49,497	52.1	1.556	31.73	4.90
6 features	$\widetilde{\in}_9', \widetilde{\in}_4', \widetilde{\text{Fr.}}_9', \widetilde{\text{Fr.}}_4', \widetilde{£}_9', \widetilde{£}_4'$	492,965	2	0.001	49,296	52.1	1.538	31.60	4.87
			Signal $> 1.0_{-4}$						
1 feature	$\widetilde{\in}_9'$	510,553	4	0.0001	460	56.7	0.070	0.650	10.8
2 features	$\widetilde{\in}_9', \widetilde{\in}_4'$	508,616	3	0.0001	916	58.6	0.291	1.206	24.2
3 features	$\widetilde{\in}_9', \widetilde{\in}_4', \widetilde{\text{Fr.}}_9'$	503,017	2	0.0001	1,811	58.9	0.467	2.048	22.8
4 features	$\widetilde{\in}_9', \widetilde{\in}_4', \widetilde{\text{Fr.}}_9', \widetilde{\text{Fr.}}_4'$	502,243	2	0.001	1,557	59.6	0.447	1.785	25.0
5 features	$\widetilde{\in}_9', \widetilde{\in}_4', \widetilde{\text{Fr.}}_9', \widetilde{\text{Fr.}}_4', \widetilde{£}_9'$	494,975	2	0.001	2,178	58.7	0.523	2.392	21.9
6 features	$\widetilde{\in}_9', \widetilde{\in}_4', \widetilde{\text{Fr.}}_9', \widetilde{\text{Fr.}}_4', \widetilde{£}_9', \widetilde{£}_4'$	492,965	2	0.001	2,711	56.8	0.508	2.796	18.2

each currency pair we now achieve a somewhat better performance for all trading signals than before using several first derivatives, compare Tables 6 and 7. We obtain $rp = 4.80$ for four attributes in comparison to $rp = 4.62$ with three attributes. The results on the stronger signals are also improved, we now achieve $rp = 25.0\,\%$ in comparison to $rp = 20.1\,\%$.

4.5 Towards a Practical Trading Strategy

For each market situation $\underline{x}$ present in the test data, the sparse grid regressor $u_L^s(\underline{x})$ yields a value which indicates the predicted increase or decrease of the exchange rate f_1. So far, trading on *all* signals showed some profit. But if one would include transaction costs this approach would no longer be viable, although low transaction costs are nowadays common in the foreign exchange market. Most brokers charge no commissions or fees whatsoever and the width of the bid/ask spread is thus the relevant quantity for the transaction costs. We assume here for simplicity that the spread is the same whether the trade involves a small or large amount of currency. It is therefore sufficient to consider the profit per trade independent of the amount of currency. Consequently, the average profit per trade needs to be at least above the average spread to result in a profitable strategy. This spread is typically five pips or less for EUR/USD and can nowadays even go down to one pip during high trading with some brokers. Note that in our case one pip is roughly equivalent to a change of 8.5_{-5} of our normalized target attribute for the time interval of the test data

Table 8 cp per trade the forecast of EUR/USD at $\hat{k} = 15$ ticks into the future using different attribute selections on the 10 % remaining test data

Currencies	Strategy	cp	Trades	cp per trade
$\tilde{€}'_9$	All ticks	0.741	51,056	1.4_{-5}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9$	All ticks	1.380	50,394	2.7_{-5}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9$	All ticks	1.469	49,566	3.0_{-5}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	All ticks	1.478	49,424	3.0_{-5}
$\tilde{€}'_9, \tilde{€}'_4$	All ticks	1.084	50,862	2.1_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9$	All ticks	1.315	50,300	2.6_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4$	All ticks	1.536	50,220	3.1_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4, \tilde{£}'_9$	All ticks	1.556	49,497	3.1_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4, \tilde{£}'_9, \tilde{£}'_4$	All ticks	1.538	49,296	3.1_{-5}
$\tilde{€}'_9$	Signal $> 1.0_{-4}$	0.070	460	1.5_{-4}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9$	Signal $> 1.0_{-4}$	0.469	2,318	2.0_{-4}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9$	Signal $> 1.0_{-4}$	0.516	2,379	2.2_{-4}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	Signal $> 1.0_{-4}$	0.614	3,559	1.7_{-4}
$\tilde{€}'_9, \tilde{€}'_4$	Signal $> 1.0_{-4}$	0.291	916	3.0_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9$	Signal $> 1.0_{-4}$	0.467	1,811	2.6_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4$	Signal $> 1.0_{-4}$	0.447	1,557	2.9_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4, \tilde{£}'_9$	Signal $> 1.0_{-4}$	0.523	2,178	2.4_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4, \tilde{£}'_9, \tilde{£}'_4$	Signal $> 1.0_{-4}$	0.508	2,711	1.9_{-4}
€	FMA	0.026	414	6.3_{-5}

with an EUR/USD exchange rate of about 1.2. If the average cp per trade is larger than this value one has a potentially profitable trading strategy. In Table 8 we give this value for the different experiments of the previous section. We see that trading on all signals results in values which are below this threshold. The same can be observed for FMA.[4] However, trading on the strongest signals results in a profitable strategy in the experiments with more attribute combinations. For example, the use of $\tilde{€}'_9, \tilde{€}'_4$ results in 3.0_{-4} cp per trade, $\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4$ gives 2.9_{-4} cp per trade and $\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9$ achieves 2.2_{-4} cp per trade. But note that with this strategy one might need to have more than one position open at a given time, which means that more capital is involved. This number of open positions can vary between zero and $\hat{k}$. It is caused by the possibility of opening a position at all times between t_j and $t_j + \hat{k}\tau$, when the first position opened at time t_j is closed again. We observed in our experiments $\hat{k}$ as the maximum number of open positions even when only trading on the stronger signals. This also indicates that a strong signal is present for a longer time period.

[4] Furthermore only relatively few trades take place with FMA which makes this a strategy with a higher variance in the performance.

Table 9 cp per trade for the forecast of EUR/USD at $\hat{k} = 15$ ticks into the future using the trading strategy with opening and closing thresholds on the 10 % remaining test data

Currencies	Strategy	cp	Trades	cp per trade
$\tilde{€}'_9$	Opening $= 0$, closing $= 0$	0.416	9,465	4.4_{-5}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9$	Opening $= 0$, closing $= 0$	0.859	12,897	6.7_{-5}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9$	Opening $= 0$, closing $= 0$	0.824	11,735	7.0_{-5}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	Opening $= 0$, closing $= 0$	0.791	12,029	6.6_{-5}
$\tilde{€}'_9, \tilde{€}'_4$	Opening $= 0$, closing $= 0$	0.563	12,045	4.7_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9$	Opening $= 0$, closing $= 0$	0.931	13,331	7.0_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4$	Opening $= 0$, closing $= 0$	1.013	14,178	7.1_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4, \tilde{£}'_9$	Opening $= 0$, closing $= 0$	1.042	13,936	7.5_{-5}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4, \tilde{£}'_9, \tilde{£}'_4$	Opening $= 0$, closing $= 0$	1.022	14,493	7.1_{-5}
$\tilde{€}'_9$	Opening $= 1.0_{-4}$, closing $= 0.5 \cdot 10{-}4$	0.018	108	1.7_{-4}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9$	Opening $= 1.0_{-4}$, closing $= 0.5 \cdot 10{-}4$	0.117	718	1.6_{-4}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9$	Opening $= 1.0_{-4}$, closing $= 0.5 \cdot 10{-}4$	0.133	748	1.7_{-4}
$\tilde{€}'_9, \tilde{\text{Fr.}}'_9, \tilde{£}'_9, \tilde{¥}'_9$	Opening $= 1.0_{-4}$, closing $= 0.5 \cdot 10{-}4$	0.169	1,213	1.4_{-4}
$\tilde{€}'_9, \tilde{€}'_4$	Opening $= 1.0_{-4}$, closing $= 0.5 \cdot 10{-}4$	0.076	248	3.1_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9$	Opening $= 1.0_{-4}$, closing $= 0.5 \cdot 10{-}4$	0.166	667	2.5_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4$	Opening $= 1.0_{-4}$, closing $= 0.5 \cdot 10{-}4$	0.173	593	2.9_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4, \tilde{£}'_9$	Opening $= 1.0_{-4}$, closing $= 0.5 \cdot 10{-}4$	0.176	839	2.1_{-4}
$\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4, \tilde{£}'_9, \tilde{£}'_4$	Opening $= 1.0_{-4}$, closing $= 0.5 \cdot 10{-}4$	0.186	1,177	1.6_{-4}

To avoid the need for a larger amount of capital we also implement a tradeable strategy where at most one position is open at a given time. Here one opens a position if the buy/sell signal at a time t_j for a prediction at $\hat{k}$ time steps into the future is larger—in absolute values—than a pre-defined *opening threshold*, and no other position is open. The position is closed when a prediction in the opposite direction occurs at some time t_e in the time interval $[t_j, t_j + \tau\hat{k}]$ and the absolute value of that prediction is greater than a prescribed *closing threshold*. At the prediction time $t_j + \tau\hat{k}$ the position is closed, unless a trading signal in the same direction as that of the original prediction is present which is larger than the *opening threshold*. The latter condition avoids an additional, but unnecessary trade. Furthermore, the closing at the forecast time $t_j + \tau\hat{k}$ avoids an open position in situations with no trading activity and where no signals can be generated.

When both of the above thresholds are zero the proposed new strategy is acting on all ticks, but at most one position is open at any given time. Besides the reduction in invested capital this strategy also improves the performance with respect to the cp per trade, see top half of Table 9. In comparison to trading on all ticks this strategy improves the results by more than a factor of 2 while only considering, but not acting on all ticks. Altogether, this strategy is getting close to the profitable threshold of one pip, i.e. 8.5_{-5} in our scaling, but it is still not yet creating a true profit.

However, as observed before, a large part of the profit is generated by acting only on the strong signals. We now set for our new strategy the opening threshold to 1.0_{-4} and the closing threshold to $0.5 \cdot 10_{-4}$. This adaption of our strategy achieves results which are comparable to the trading on the strong signals only. Since at most one position is open, less capital is involved than in the case of trading on all strong signals. In the bottom half of Table 9 we give the corresponding results. We see that the cp per trade is now always above the threshold 8.5_{-5}. The highest cp per trade is 3.1_{-4}, where 248 trades take place while using $\tilde{€}'_9, \tilde{€}'_4$. For the attributes $\tilde{€}'_9, \tilde{€}'_4, \tilde{\text{Fr.}}'_9, \tilde{\text{Fr.}}'_4$ we achieve 2.9_{-4} cp per trade while acting 593 times. This might be preferable due to the larger number of trades which should lead to more stable results. Thus, we finally obtained a profitable strategy, which promises a net gain of more than one pip per trade if the spread is less than two pips.

5 Conclusions

We presented a machine learning approach based on delay embedding and regression with the sparse grid combination technique to forecast the intraday foreign exchange rates of the EUR/USD currency pair. It improved the results to take not only attributes derived from the EUR/USD rate but from further exchange rates such as the USD/JPY and/or GBP/USD rate into account. In some situations a realized potential, i.e. achieved percentage of the maximum possible cumulative profit, of more than 20 % was achieved. We also developed a practical trading strategy using an opening and closing threshold which obtained an average profit per trade larger than three pips. If the spread is on average below three pips this results in profitable trading. Thus, our approach seems to be able to learn the effect of technical trading tools which are commonly used in the intraday foreign exchange market. It also indicates that FX rates have an underlying process which is not purely Markovian, but seems to have additional structure and memory which we believe is caused by technical trading in the market.

Our methodology can be further refined especially in the choice of attributes and parameters. For example, we considered the same time frame for the first derivatives of all the involved currency pairs, i.e. $k = 9$ and $k = 4$. Using different time frames for the different exchange rates might result in a further improvement of the performance. Other intraday information such as the variance of the exchange rates or the current spread can also be incorporated. Furthermore, we did not yet take the different interest rates into account, but their inclusion into the forecasting process can nevertheless be helpful. The time of day could also be an useful attribute since the activity in the market changes during the day [22]. The use of a dimension adaptive combination technique [10, 14], where the partial functions employed depend only on a subset of all features and are adaptively chosen during the computational procedure, becomes relevant once more than, say, five attributes are used.

In our experiments we used data from 2001 to 2005 to forecast for 5 months in the year 2005. Therefore, our observations are only based on this snapshot in time of the foreign exchange market. For a snapshot from an other time interval one would most likely use different features and parameters and one would achieve somewhat changed results. Furthermore, it has to be seen if today's market behaviour, which may be different especially after the recent financial crisis, can still be forecast with such an approach, or if the way the technical trading takes place has changed fundamentally. In any case, for a viable trading system, a learning approach is necessary which relearns automatically and regularly over time, since trading rules are typically valid only for a certain period.

Note finally that our approach is not limited to the FX application. In finance it may be employed for the prediction of the behaviour of stocks or interest rates as well, here again the effect of technical traders makes this approach sensible, since the underlying process is then not purely Markovian anymore. It also can be applied to more general time series problems with a large amount of data which arise in many applications in biology, medicine, physics, econometrics and computer science.

Acknowledgements We thank Bastian Bohn and Alexander Hullmann for their assistance with the numerical experiments.

References

1. D. J. E. Baestaens, W. M. van den Bergh, and H. Vaudrey. Market inefficiencies, technical trading and neural networks. In C. Dunis, editor, *Forecasting Financial Markets*, pages 245–260. Wiley, 1996.
2. W. Brock, J. Lakonishok, and B. LeBaron. Simple technical trading rules and the stochastic properties of stock returns. *The Journal of Finance*, 47(5):1731–1764, 1992.
3. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.
4. Y.-W. Cheung and C. Y.-P. Wong. Foreign exchange traders in Hong Kong, Tokyo, and Singapore: A survey study. In T. Bos and T. A. Fetherston, editors, *Advances in Pacific Basin Financial Markets*, volume 5, pages 111–134. JAI, 1999.
5. R. Curcio, C. Goodhart, D. Guillaume, and R. Payne. Do technical trading rules generate profits? Conclusions from the intra-day foreign exchange market. *Int. J. Fin. Econ.*, 2(4):267–280, 1997.
6. M. A. H. Dempster, T. W. Payne, Y. S. Romahi, and G. W. P. Thompson. Computational learning techniques for intraday FX trading using popular technical indicators. *IEEE Trans. Neural Networks*, 12(4):744–754, 2001.
7. M. Engel. Time series analysis. Part III Essay, University of Cambridge, 1991.
8. T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.
9. J. Garcke. Regression with the optimised combination technique. In W. Cohen and A. Moore, editors, *Proceedings of the 23rd ICML '06*, pages 321–328, 2006.
10. J. Garcke. A dimension adaptive sparse grid combination technique for machine learning. In W. Read, J. W. Larson, and A. J. Roberts, editors, *Proceedings of the 13th Biennial Computational Techniques and Applications Conference, CTAC-2006*, volume 48 of *ANZIAM J.*, pages C725–C740, 2007.

11. J. Garcke and M. Griebel. Classification with sparse grids using simplicial basis functions. *Intelligent Data Analysis*, 6(6):483–502, 2002.
12. J. Garcke, M. Griebel, and M. Thess. Data mining with sparse grids. *Computing*, 67(3):225–253, 2001.
13. J. Garcke and M. Hegland. Fitting multidimensional data using gradient penalties and the sparse grid combination technique. *Computing*, 84(1–2):1–25, April 2009.
14. T. Gerstner and M. Griebel. Dimension–Adaptive Tensor–Product Quadrature. *Computing*, 71(1):65–87, 2003.
15. F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–265, 1995.
16. P. Grassberger and I. Procaccia. Characterization of strange attractors. *Phys. Rev. Lett*, 50:346–349, 1983.
17. M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens, editors, *Iterative Methods in Linear Algebra*, pages 263–281. IMACS, Elsevier, North Holland, 1992.
18. D. M. Guillaume, M. M. Dacorogna, R. R. Davé, U. A. Müller, R. B. Olsen, and O. V. Pictet. From the bird's eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets. *Finance Stochast.*, 1(2):95–129, 1997.
19. M. Hegland, J. Garcke, and V. Challis. The combination technique and some generalisations. *Linear Algebra and its Applications*, 420(2–3):249–275, 2007.
20. M. Hegland, O. M. Nielsen, and Z. Shen. Multidimensional smoothing using hyperbolic interpolatory wavelets. *Electronic Transactions on Numerical Analysis*, 17:168–180, 2004.
21. I. Horenko. Finite element approach to clustering of multidimensional time series. *SIAM Journal on Scientific Computing*, 32:62–83, 2010.
22. K. Iwatsubo and Y. Kitamura. Intraday evidence of the informational efficiency of the yen/dollar exchange rate. *Applied Financial Economics*, 19(14):1103–1115, 2009.
23. H. Kantz and T. Schreiber. *Nonlinear time series analysis*. Cambridge University Press, 1997.
24. K. Lien. *Day Trading the Currency Market*. Wiley, 2005.
25. Y.-H. Lui and D. Mole. The use of fundamental and technical analyses by foreign exchange dealers: Hong Kong evidence. *J. Int. Money Finance*, 17(3):535–545, 1998.
26. A. L. M. Verleysen, E. de Bodt. Forecasting financial time series through intrinsic dimension estimation and non-linear data projection. In J. Mira and J. V. Sánchez-Andrés, editors, *Engineering Applications of Bio-Inspired Artificial Neural Networks, Volume II*, volume 1607 of *Lecture Notes in Computer Science*, pages 596–605. Springer, 1999.
27. C. J. Neely and P. A. Weller. Intraday technical trading in the foreign exchange market. *J. Int. Money Finance*, 22(2):223–237, 2003.
28. C. Osler. Support for resistance: Technical analysis and intraday exchange rates. *Economic Policy Review*, 6(2):53–68, 2000.
29. T. R. Reid. *The United States of Europe, The New Superpower and the End of the American Supremacy*. Penguin Books, 2004.
30. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.
31. F. Takens. Detecting strange attractors in turbulence. In D. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer, 1981.
32. M. P. Taylor and H. Allen. The use of technical analysis in the foreign exchange market. *J. Int. Money Finance*, 11(3):304–314, 1992.
33. A. N. Tikhonov and V. A. Arsenin. *Solutions of ill-posed problems*. W. H. Winston, Washington D.C., 1977.
34. G. Tsibouris and M. Zeidenberg. Testing the efficient markets hypothesis with gradient descent algorithms. In A.-P. Refenes, editor, *Neural Networks in the Capital Markets*, chapter 8, pages 127–136. Wiley, 1995.
35. G. Wahba. *Spline models for observational data*, volume 59 of *Series in Applied Mathematics*. SIAM, Philadelphia, 1990.

36. C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990*, volume 31 of *Notes on Num. Fluid Mech.* Vieweg-Verlag, 1991.
37. H. G. Zimmermann, R. Neuneier, and R. Grothmann. Multi-agent modeling of multiple FX-markets by neural networks. *IEEE Trans. Neural Networks*, 12(4):735–743, 2001.

Dimension- and Time-Adaptive Multilevel Monte Carlo Methods

Thomas Gerstner and Stefan Heinz

Abstract We use the multilevel Monte Carlo method to estimate option prices in computational finance and combine this method with two adaptive algorithms. In the first algorithm we consider time discretization and sample size as two separate dimensions and use dimension-adaptive refinement to optimize the error with respect to these dimensions in relation to the computational costs. The second algorithm uses locally adaptive timestepping and is constructed especially for non-Lipschitz payoff functions whose weak and strong order of convergence is reduced when the Euler-Maruyama method is used to discretize the underlying SDE. The numerical results show that for barrier and double barrier options the convergence order for smooth payoffs can be recovered in these cases.

1 Introduction

The pricing of exotic options often leads to the problem of calculating high-dimensional integrals. The high dimension typically arises from the number of timesteps in the time discretization of the underlying stochastic process. If those integrals cannot be solved analytically, numerical methods are used to estimate a solution. Monte Carlo methods are a popular way for estimating option prices, but these methods generally suffer from low convergence rates. They can be improved with various variance reduction techniques or importance sampling, which can change the constant, but not the rate of convergence [9].

T. Gerstner (✉) · S. Heinz
Institut für Mathematik, Johann Wolfgang Goethe-Universität, D-60054, Frankfurt am Main, Germany
e-mail: gerstner@math.uni-frankfurt.de; heinz@math.uni-frankfurt.de

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_5,

As a substantial improvement, Giles introduced the multilevel Monte Carlo (MLMC) method [5] to estimate the expectation of $P(S(T))$ where P represents the payoff function of an option and the underlying asset price S is modelled by the stochastic differential equation (SDE)

$$\mathrm{d}S(t) = \mu(S,t)\mathrm{d}t + \sigma(S,t)\mathrm{d}W(t), \quad 0 \le t \le T. \tag{1}$$

The algorithm is similar to the standard Monte Carlo method but uses a multigrid idea to reduce the computational costs. A discretization of this SDE with the Euler method leads to a computational complexity of $O(\epsilon^{-3})$ to achieve a root mean square error (RMSE) of ϵ for the Monte Carlo method while the MLMC method can achieve a complexity of $O(\epsilon^{-2}(\log \epsilon)^2)$. Further studies show that discretization schemes with higher strong order of convergence [6] or simulations using quasi-random numbers [7] lead to further improvements in the computational complexity.

A completely different approach to solve high-dimensional integrals is dimension-adaptive tensor-product quadrature [3] which is based on the sparse grid method [4]. In contrast to Monte Carlo methods, the sample points are not placed randomly but completely deterministic. The dimension-adaptive algorithm thereby decides in which dimensions more grid points need to be placed in order to best reduce the estimation error. It turns out that the multilevel Monte Carlo method and the dimension-adaptive sparse grid method are based on similar ideas. In [11] this is described in a detailed way for elliptic stochastic partial differential equations.

In this paper we at first combine both approaches to create a dimension-adaptive multilevel Monte Carlo method which obtains the same convergence rate in the RMSE as the original MLMC method but attains the given error threshold with lower costs.

As a further improvement, another adaptive algorithm is presented that is quite effective for payoff functions with jumps such as barrier options. It uses the Brownian Bridge method and adaptively refines the Euler path when it comes close to the barrier. The numerical results show that the time-adaptive MLMC method can recover the order of convergence for smooth payoffs with only constant additional computational costs.

The properties of these two adaptive multilevel algorithms will be underlined by numerical results.

2 Multilevel Monte Carlo Method

We consider the scalar stochastic differential equation

$$\mathrm{d}S(t) = \mu(S,t)\mathrm{d}t + \sigma(S,t)\mathrm{d}W(t), \quad 0 \le t \le T \tag{2}$$

with drift function $\mu : \mathbb{R} \times [0, T] \to \mathbb{R}$, volatility $\sigma : \mathbb{R} \times [0, T] \to \mathbb{R}$, starting value $S_0 \in \mathbb{R}_0^+$ and a Wiener process $(W_t)_{t \geq 0}$. Using the Euler method with equidistant timestep h we get the discretization

$$\widehat{S}_{n+1} = \widehat{S}_n + \mu(\widehat{S}_n, t_n)h + \sigma(\widehat{S}_n, t_n)\Delta W_n.$$

with Wiener increments $\Delta W_n = W_{n+1} - W_n$. Given a payoff function $P : \mathbb{R} \to \mathbb{R}$ it is our aim to calculate an estimate of the expectation $E[P(S(T))]$.

The MLMC method approximates P by $\widehat{P}_l$ using a series of discretizations with timesteps $h_l = M^{-l}T, l = 0, 1, \ldots, L$, to construct estimators

$$\widehat{Y}_l = N_l^{-1} \sum_{i=1}^{N_l} \left(\widehat{P}_l^{(i)} - \widehat{P}_{l-1}^{(i)} \right)$$

for levels $l > 0$ and

$$\widehat{Y}_0 = N_0^{-1} \sum_{i=1}^{N_0} \widehat{P}_0^{(i)}$$

for level $l = 0$, such that

$$\widehat{Y} = \sum_{l=0}^{L} \widehat{Y}_l$$

is an estimator for

$$E[\widehat{P}_L] = E[\widehat{P}_0] + \sum_{l=1}^{L} E[\widehat{P}_l - \widehat{P}_{l-1}].$$

If there are positive constants $\alpha \geq \frac{1}{2}, \beta, c_1, c_2, c_3$, such that

(i) $E[\widehat{P}_l - P] \leq c_1 h_l^\alpha$

(ii) $E[\widehat{Y}_l] = \begin{cases} E[\widehat{P}_0], & l = 0 \\ E[\widehat{P}_l - \widehat{P}_{l-1}], & l > 0 \end{cases}$

(iii) $\mathrm{Var}[\widehat{Y}_l] \leq c_2 N_l^{-1} h_l^\beta$ and

(iv) C_l, the computational complexity of $\widehat{Y}_l$, is bounded by

$$C_l \leq c_3 N_l h_l^{-1},$$

with α and β corresponding to the weak and doubled strong convergence order of the Euler method, for the overall computational complexity C follows (see [5]) that

$$C \leq \begin{cases} c_4 \epsilon^{-2}, & \beta > 1, \\ c_4 \epsilon^{-2} (\log \epsilon)^2, & \beta = 1, \\ c_4 \epsilon^{-2-(1-\beta)/\alpha}, & 0 < \beta < 1. \end{cases}$$

Neglecting the log-term this implies

$$RMSE = \begin{cases} O(C^{-1/2}), & \beta \geq 1, \\ O(C^{\frac{-\alpha}{2\alpha+1-\beta}}), & \beta < 1. \end{cases}$$

In comparison, the RMSE of the the standard Monte Carlo method is of order $O(C^{\frac{-\alpha}{2\alpha+1}})$ which corresponds to a convergence rate of 1/3 if $\alpha = 1$, while the MLMC method can obtain a rate of 1/2 if $\beta = 1$.

The MLMC algorithm then works as follows. It estimates for each level l an optimal number N_l of samples to make sure that $V[\widehat{Y}] < \frac{1}{2}\epsilon^2$ and increases l until the estimator for the bias $\widehat{Y}_l^2$ is smaller than $\frac{1}{2}\epsilon^2$. This finally leads to

$$MSE = V[\widehat{Y}] + E[\widehat{Y} - Y]^2 \leq \epsilon^2$$

for $Y = E[P]$.

3 Adaptive Multilevel Monte Carlo Methods

3.1 Dimension-Adaptive Algorithm

The option pricing problem can often be written as an integration problem which usually has to be numerically solved by quadrature methods. If the integrand is non-smooth then these quadrature methods have to be adapted accordingly. Such adaptive quadrature methods in general try to find important areas of the integrand, such that adaptive refinement in those areas results in a highest possible error reduction. These methods allocate points in the given areas and will automatically stop when a given error bound is reached, see [1, 2] for adaptive sparse grid approaches. The dimension-adaptive approach in [3] is especially suited for high-dimensional quadrature problems and considers the different dimensions instead of local subareas for refinement.

Interestingly, dimension-adaptive refinement and the multilevel Monte Carlo method are based on similar ideas. In the following, we combine these two approaches. In contrast to the standard MLMC algorithm, the dimension-adaptive algorithm does not calculate the values of N_l in advance but rather checks for which levels more samples give the largest reduction in the MSE and then doubles the previous amount of samples. If, on the other hand, the bias estimator contributes the largest part to the MSE the level will be increased. This ensures that both errors, the quadrature and the path-discretization error are as equal as possible such that the prescribed accuracy ϵ is best obtained.

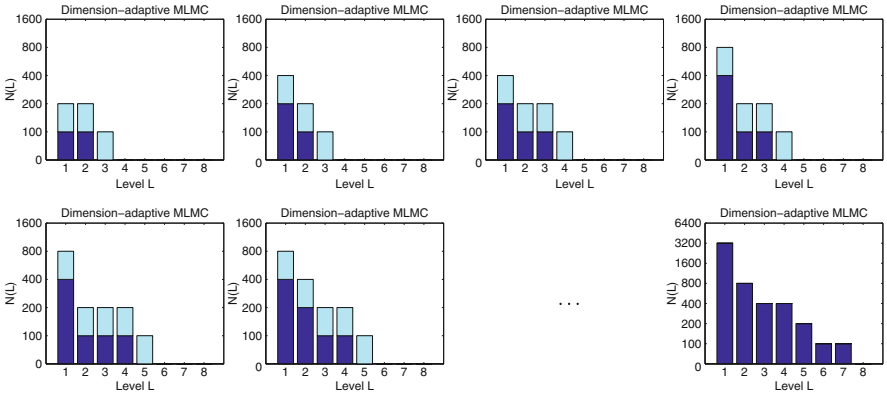


Fig. 1 Plots of the dimension-adaptive algorithm showing the increase in N(L) after each step and the final shape of the samples at each level

The dimension-adaptive MLMC algorithm is based on [8] and can be described as follows:

1. Set $N_0, N_1 = 100$
2. Determine V_l for $l = 0, \ldots, L$ such that $V := \sum_{l=0}^{L} V_l$ estimates the variance and $B := (\widehat{Y}_L/(M^\alpha - 1))^2$ estimates the squared bias with weak convergence order α
3. If $V + B < \epsilon^2$ stop
4. Else
   - If $V > B$ determine the level $l = 0, \ldots, L$, which has the largest variance/work and double N_l
   - If $B > V$ then set $N_{L+1} = 100$ and $L \to L + 1$
5. Go to step 2.

At the start we need samples at level $l = 1$ because only $\widehat{Y}_l$ for $l > 0$ is a good estimator for the bias. Since

$$V[\widehat{Y}] = \sum_{l=0}^{L} N_l^{-1} \widehat{V}_l$$

and $\widehat{V}_l$ is the variance of the samples at level l we set $V_l = N_l^{-1}\widehat{V}_l$. In step 2 we search for the level in which further samples will lead to the largest reduction in the MSE compared to the computational cost. The plots in Fig. 1 explain this in more detail and show how the algorithm is executed stepwise. The actual simulated amount of samples is dark-colored and the possibly increased N_l is light-colored. In each step the algorithm then increases the N_l that gives the highest error reduction

until the optimal shape for a given accuracy is reached in the last plot. Concerning the MSE it is useful to keep in mind that

$$MSE = Var[\widehat{Y}] + E[\widehat{Y} - Y]^2$$

So both, variance and bias, have to be reduced for a good result. The variance is reduced by increasing the amount of samples while the bias decreases due to the increase in the discretization steps which equals the dimension of the integral to solve. This is the connection to the dimension-adaptive tensor product method. In our method there are only two dimensions: the amount of samples at each level $N_l, l = 0, \ldots, L$, and the dimension of the integral d.

The advantage of our algorithm is that in contrast to the standard MLMC method, the contributions of the variances and the squared bias which are summed up to the MSE are distributed in an optimal way by the algorithm and do not necessarily need to be both bounded by $\epsilon^2/2$. As we will see in our numerical results this leads to a MSE that is closer to the targeted accuracy ϵ.

3.2 Time-Adaptive Algorithm

Several types of options, such as binary or barrier options, have jumps in the payoff function which lead to a reduced strong convergence rate when using a discretization method like the Euler-Maruyama method. We intend to solve this problem by adaptively refining the time discretization of the SDE when this refinement leads to a reduced RMSE [12, 13]. A refinement usually takes place when a simulated path is close to a discontinuity of the payoff function. Our numerical results show that it only takes a constant factor of additional costs to recover the convergence order of the smooth case.

In this paper, we especially consider barrier options, where a positive payoff only occurs if the price stays above or below a given barrier B. As a refinement indicator we use the probability of breaking the barrier between two simulated points using a Brownian Bridge to create this point. Thereby, we use the standard Brownian Bridge to construct a path for the Wiener process and then take the mean of forward and backward Euler-Maruyama estimates to construct the midpoint

$$\begin{aligned} S_{i-1/2} &= \frac{1}{2}\left(S_{i-1} + S_{i-1} r (t_{i-1/2} - t_{i-1}) + S_{i-1}\sigma(W_{i-1/2} - W_{i-1})\right) \\ &\quad + \frac{1}{2}\left(\frac{S_i}{1 + r(t_i - t_{i-1/2}) + \sigma(W_i - W_{i-1/2})}\right). \end{aligned} \tag{3}$$

In the extremely rare case that $1 + r(t_i - t_{i-1/2}) + \sigma(W_i - W_{i-1/2}) \leq 0$, a new path is simulated instead of the current one.

We next calculate the probability Ψ that $S_{i-1/2}$ can fall below the given Barrier which is defined by

$$\Psi(S_{i-1}, S_i) := \mathbb{P}\left(S_{i-1/2} < B\right)$$

and can be calculated by solving the inequality $S_{i-1/2} < B$ with respect to $W_{i-1/2}$ using (3) and the Brownian Bridge construction $W_{i-1/2} = \frac{1}{2}(W_{i-1} + W_i) + \frac{\sqrt{t_i - t_{i-1}}}{2} Z$, $Z \sim N(0,1)$. Integrating the standard normal densitiy over the domain in which $S_{i-1/2} < B$ and neglecting the part with the lower probability yields the expression

$$\begin{aligned}\Psi(S_{i-1}, S_i) &= \Phi\left(\frac{-b_2}{2b_1} - \sqrt{\left(\frac{b_2}{2b_1}\right)^2 - \frac{b_3}{b_1}}\right) \\ &\quad + \Phi\left(\frac{-b_2}{2b_1} + \sqrt{\left(\frac{b_2}{2b_1}\right)^2 - \frac{b_3}{b_1}}\right) - \Phi\left(\frac{a_3}{a_4}\right) \\ &\geq \Phi\left(\frac{-b_2}{2b_1} - \sqrt{\left(\frac{b_2}{2b_1}\right)^2 - \frac{b_3}{b_1}}\right)\end{aligned}$$

with $\Delta h_i = \frac{1}{2}(t_i - t_{i-1})$, $\Delta W_i = W_i - W_{i-1}$ and $a_1 = \frac{1}{2} S_{i-1}(1 + \Delta h_i r + \sigma \Delta W_i/2)$, $a_2 = \frac{1}{4} S_{i-1} \sigma \sqrt{2\Delta h_i}$, $a_3 = \frac{2}{S_i}(1 + \Delta h_i r + \sigma \Delta W_i/2)$, $a_4 = (\sigma\sqrt{2\Delta h_i})/S_i$ and $b_1 = -a_2 a_4$, $b_2 = -a_1 a_4 + a_2 a_3 + B a_4$, $b_3 = 1 + a_1 a_3 - B a_3$.

A typical path is presented in Fig. 2. We see that for the starting value $S(0) = 1$ the corresponding Wiener path is refined when the path of S comes close to the barrier $B = 0.95$. In this simulation the path did not cross the barrier. If the barrier is crossed then the payoff is set to zero immediately and no further refinement is made for this path.

In order to apply this technique to the MLMC method we need to add refinement thresholds $w_k, k = 0, \ldots, L$ that control the temporal refinement at each level of the MLMC method. A refinement in the interval $[t_{i-1}, t_i]$ takes place if $\Psi(S_{i-1}, S_i) > \Phi(w_k)$. The MLMC estimator then becomes

$$E[\hat{P}_L^{w_L}] = E[\hat{P}_0^{w_0}] + \sum_{l=1}^{L} E[\hat{P}_l^{w_l} - \hat{P}_{l-1}^{w_l - 1}]$$

where $\hat{P}_l^{w_l}$ is the approximation with respect to the timestep $h_l = M^{-l} T$ and the adaptive path discretization parameter w_l.

The time-adaptive MLMC algorithm finally can be described as follows

1. Set $N_0, N_1 = 100$
2. Determine $V_l^{w_l}$ for $l = 0, \ldots, L$ such that $V := \sum_{l=0}^{L} V_l^{w_l}$

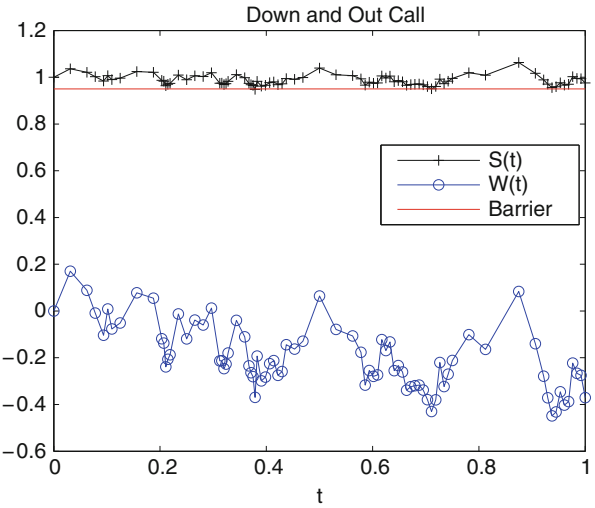


Fig. 2 Realization of an adaptively refined standard Wiener process path and the corresponding realization of the solution of the SDE

3. Define optimal $N_l, l = 0, \ldots, L$ as in the standard algorithm and calculate extra samples if N_l has increased
4. Stop if RMSE$< \epsilon$ and $L \geq 2$
5. Else set $L := L + 1$, $N_L = 100$ and go to step 2.

In step 2 we first simulate a path with M^l equidistant timesteps and then refine the path according to the threshold w_l. For the estimator of $E[\hat{P}_l^{w_l} - \hat{P}_{l-1}^{w_{l-1}}]$ we use the same Wiener path for the coarse level as for the fine one and refine it with threshold w_{l-1} to decrease the variance.

The values for w_l are optimized beforehead such that the additional computational costs are always a constant factor of the costs without refinement for each level. This way, the computational costs are increased only by a constant factor while the convergence rate of the RMSE is increased from order 1/3 to order 1/2.

4 Numerical Results

The following results are based on a geometric Brownian motion for the dynamics of the underlying asset

$$dS = rSdt + \sigma S dW_t, \quad 0 \leq t \leq 1,$$

with $S_0 = 1$, $r = 0.05$ and $\sigma = 0.2$. The SDE is discretized with the Euler-Maruyama method.

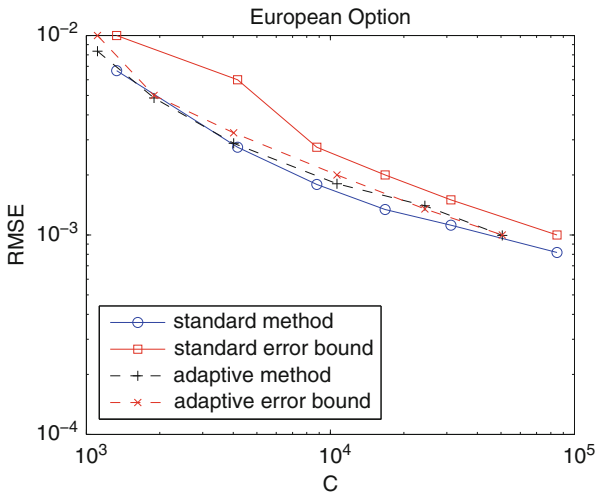


Fig. 3 Comparison of the standard MLMC method and the dimension-adaptive method with the given error bound ϵ for an European Option

For all options we discuss, there exists a closed form solution so we directly calculate the RMSE of 100 simulated prices instead of using the estimator for bias and variance of the algorithm with $M = 2$. Since we are mainly interested in the convergence rates, we show plots that compare the computational cost to calculate the estimator with the accuracy of our calculated prices. The accuracy can be expressed by the input parameter ϵ or by the obtained root mean square error.

4.1 Dimension-Adaptive Algorithm

We first compare the standard MLMC method to our dimension-adaptive algorithm by looking at the convergence rates for an European option.

Assuming a strike price of $K = 1$ and time to maturity $T = 1$, the European option has the discounted payoff function

$$P = e^{-rT}(S(T) - K)^+$$

and approximate price 0.1045 for those parameters. Our plot in Fig. 3 shows the convergence rates of the RMSE of the standard and the dimension-adaptive method. The computed error bounds ϵ are shown as well. We see that both algorithms converge equally well with rate 1/2 as expected. The main difference is that the standard MLMC algorithm reaches the same RMSE as the dimension-adaptive method for larger ϵ which means that it is more accurate than requested. The RMSE

Table 1 Comparison of the MC, MLMC and dimension-adaptive method for an Asian, Lookback and Digital option with expected and computed RMSE rates

	Expected RMSE			Computed RMSE		
Option	MC	MLMC	Adaptive	MC	MLMC	Adaptive
Asian	1/3	1/2	1/2	0.34	0.53	0.54
Lookback	1/4	1/2	1/2	0.24	0.42	0.39
Digital	1/3	2/5	2/5	0.32	0.37	0.37

of the dimension-adaptive algorithm is much closer to the given threshold ϵ and therefore does not use more computational cost than necessary.

Furthermore we simulated the RMSE convergence rate for several types of options and compared the Monte Carlo, the MLMC and the dimension-adaptive method. We see in Table 1 that the MLMC method and the dimension-adaptive method have similar RMSE rates which confirm the theoretical values that are expected, since for the Asian option $\alpha = \beta = 1$, for the Lookback option $\alpha = 1/2$, $\beta = 1$ and for the Digital option $\alpha = 1$, $\beta = 1/2$. The RMSE rate of the Monte Carlo simulation is always lower than the MLMC convergence rate and will not reach it even for a higher α.

4.2 Time-Adaptive Algorithm

We now evaluate the performance of the time-adaptive MLMC algorithm. For both examples we choose $w_l = -l - 1$ for $l = 0, \ldots, L$ as refinement thresholds which correspond to the inverse cumulative normal distribution, so $w_0 = -1$ would equal a 15.9 % probability of crossing the barrier and the simulated path will be refined until the crossing probability is below that value in each interval.

We first consider a down-and-out barrier option with strike $K = 1$, barrier $B = 0.95$ and time to maturity $T = 1$. The discounted payoff is calculated by

$$P = \begin{cases} e^{-r}(S(T) - K)^+ \text{ if } S(t) > B \text{ for all } 0 \le t \le T \\ 0, \text{ otherwise,} \end{cases}$$

and in the discrete version with N steps by

$$P = \begin{cases} e^{-r}(S(t_N) - K)^+ \text{ if } S(t_i) > B \text{ for } i = 0, \ldots, N \\ 0, \text{ otherwise.} \end{cases}$$

The barrier option has price 0.05636 and is compared to the standard MLMC algorithm and conditional sampling combined with MLMC [6, 9] in which the payoff is rewritten as a product of probabilities to break the barrier for N timesteps the following discrete way:

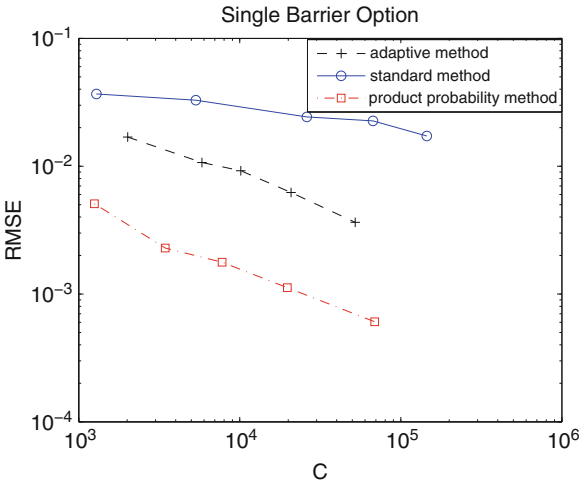


Fig. 4 Comparison of the standard and improved MLMC method with the time-adaptive method for a Single Barrier Option

$$P = e^{-r}(S(t_N) - K)^+ \prod_{i=0}^{N-1} p_i$$

with

$$p_i = 1 - \exp\left(\frac{-2(S_n - B)^+(S_{n+1} - B)^+}{\sigma^2 S_n^2 T/N}\right).$$

This payoff is Lipschitz again so the optimal weak and strong convergence of the Euler method is recovered plus a variance reducing effect takes place. In this case the time-adaptive algorithm does not lead to an improvement. In Fig. 4 we see that both the time-adaptive and the improved method converge with order 1/2 and a smaller convergence constant of the improved method due to the variance reducing effect. They are both better than the standard method that only converges with rate 1/4.

However, not all payoffs can be rewritten as a product of probabilities. The more conditions a payoff function includes the more difficult it is to find a representation of the payoff that is Lipschitz in S. In our next example we look at a double barrier option with discounted payoff

$$P = \begin{cases} e^{-r}(S(T) - K)^+ \text{ if } S(t) > B_l \wedge S(t) < B_u \text{ for all } 0 \le t \le T \\ 0, \text{ otherwise,} \end{cases}$$

and in the discrete version with N steps by

$$P = \begin{cases} e^{-r}(S(t_n) - K)^+ \text{ if } S(t_i) > B_u \wedge S(t_i) < B_u \text{ for } i = 0, \ldots, N \\ 0, \text{ otherwise.} \end{cases}$$

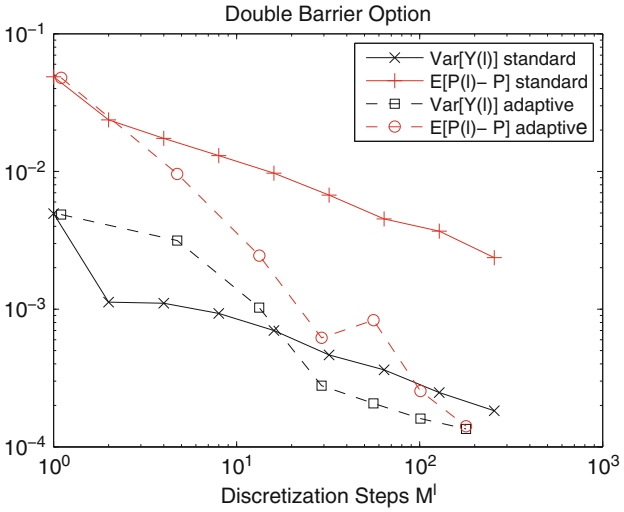


Fig. 5 Comparison of the Alpha and Beta rates of the standard and time-adaptive MLMC method for a Double Barrier Option

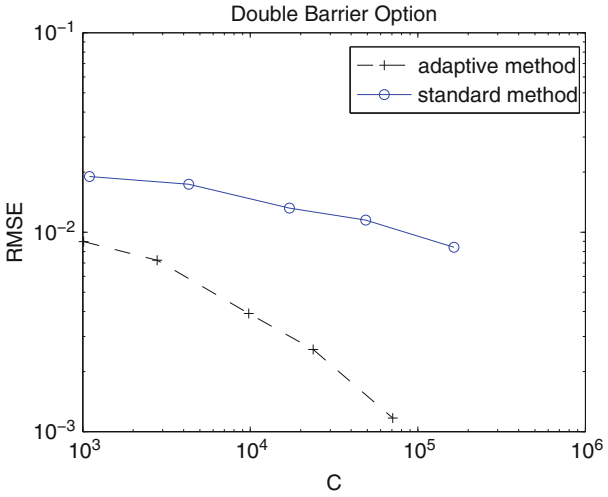


Fig. 6 Comparison of the standard and time-adaptive MLMC method for a Double Barrier Option

For this option there still exists a representation as a product of probabilities that uses an infinite sum which can only be evaluated by cutting it off [10]. For this option the time-adaptive algorithm keeps working as usual with no additional complications as shown in Fig. 6. For $K = 1$, $B_l = 0.85$ and $B_u = 1.25$ the price of the option is 0.01842. We now compare the time-adaptive algorithm to the standard MLMC method and at first estimate $E[\widehat{P}_l - P]$ and $\mathrm{Var}[\widehat{Y}_l]$ with various

numbers of discretization steps for both methods to determine α and β. We see that both values are halved from 1 for the standard European payoff to 1/2. The time-adaptive algorithm recovers the β to order 1 again and slightly improves the α which, however, has no influence on the RMSE rate since now $\beta \geq 1$ (see Fig. 5). This finally leads to an increased convergence rate of the RMSE to the optimal 1/2 as shown in Fig. 6 while the standard method converges with order 1/4 for a higher ϵ but should converge with order 1/3 for smaller ϵ due to the results in Sect. 2.

5 Concluding Remarks

In this article we presented two adaptive algorithms that both can improve the standard multilevel Monte Carlo algorithm. The numerical results for the dimension-adaptive algorithm have shown that it is possible to estimate an option price that is closer to the requested accuracy and therefore needs less computational cost. Nevertheless this technique does not lead to an increased convergence rate. We further developed a time-adaptive algorithm that optimizes the MLMC method in case of non-Lipschitz payoff functions. We chose the example of single and double barrier options to point out that our algorithm refines a simulated path correctly which then leads to a higher weak and strong convergence capped by those convergence orders of the smooth case. As MLMC theory predicts the convergence rate of the RMSE then also recovers to 1/2.

More research must be done at calculating error bounds for the refinement thresholds. We plan on combining the dimension-adaptive and time-adaptive algorithms in such a way that a third dimension will be the refining parameter. We then optimize between those three dimensions in order to equalize the three possible errors that contribute to the RMSE.

Using quasi-random numbers instead of pseudo-random numbers is another possibility to improve the MLMC method. In [6] encouraging numerical results have been presented for a multilevel quasi-Monte Carlo method. We are currently working on a theoretical foundation for these results and a combination of the MLQMC method with the adaptive methods presented in this paper.

References

1. Bonk, Thomas: *A new algorithm for multi-dimensional adaptive numerical quadrature*, In Hackbusch, W.; Wittum, G., editors: *Adaptive Methods: Algorithms, Theory and Applications*, volume 46 of Notes on Numerical Fluid Mechanics, Vieweg, Braunschweig, 1993.
2. Bungartz, Hans-Joachim; Dirnstorfer, Stefan: *Multivariate Quadrature on Adaptive Sparse Grids*, Computing, 71(1): 89–114, 2003.
3. Gerstner, Thomas; Griebel, Michael: *Dimension-Adaptive Tensor-Product Quadrature*, Computing, 71(1): 65–87, 2003.

4. Gerstner, Thomas; Griebel, Michael: *Sparse grids*, Encyclopedia of Quantitative Finance, J. Wiley & Sons, 2009.
5. Giles, Michael B.: *Multilevel Monte Carlo path simulation*, Operations Research, 56(3):607–617, 2008.
6. Giles, Michael B.: *Improved multilevel Monte Carlo convergence using the Milstein scheme*, In Keller, Alexander; Heinrich, Stefan; Niederreiter, Harald, editors: *Monte Carlo and Quasi-Monte Carlo Methods 2006*, Springer, Berlin, 2008.
7. Giles, Michael B.; Higham, Desmond J.; Mao, Xuerong: *Analysing multi-level Monte Carlo for options with non-globally Lipschitz payoff*, Finance & Stochastics, 13(3): 403–413, Springer, 2009
8. Giles, Michael B.; Waterhouse, B.J.: *Multilevel quasi-Monte Carlo path simulation*, Radon Series on Computational and Applied Mathematics, Springer, 2009.
9. Glassermann, Paul: *Monte Carlo Methods in Financial Engineering*, Springer, New York, 2004.
10. Gobet, Emmanuel: *Advanced Monte Carlo methods for barrier and related exotic options*, Mathematical Modelling and Numerical Methods in Finance, 497–530, 2008.
11. Harbrecht, Helmut; Peters, Michael; Siebenmorgen, Markus: *On multilevel quadrature for elliptic stochastic partial differential equations*, In Garcke, J.; Griebel, M., editors: *Sparse Grids and Applications*, LNCSE. Springer, 161–179, 2012.
12. Szepessy, Anders; Tempone, Raul; Zouraris, Georgios E.: *Adaptive weak approximation of stochastic differential equations*, Communications on Pure and Applied Mathematics, 54(10):1169–1214, 2001.
13. Szepessy, A.; Hoel, H; von Schwerin, E.; Tempone, R.: *Implementation and Analysis of an Adaptive Multi Level Monte Carlo Algorithm*, Monte Carlo Methods and Applications, 2010.

An Efficient Sparse Grid Galerkin Approach for the Numerical Valuation of Basket Options Under Kou's Jump-Diffusion Model

Michael Griebel and Alexander Hullmann

Abstract We use a sparse grid approach to discretize a multi-dimensional partial integro-differential equation (PIDE) for the deterministic valuation of European put options on Kou's jump-diffusion processes. We employ a generalized generating system to discretize the respective PIDE by the Galerkin approach and iteratively solve the resulting linear system. Here, we exploit a newly developed recurrence formula, which, together with an implementation of the unidirectional principle for non-local operators, allows us to evaluate the operator application in linear time. Furthermore, we exploit that the condition of the linear system is bounded independently of the number of unknowns. This is due to the use of the Galerkin generating system and the computation of L_2-orthogonal complements. Altogether, we thus obtain a method that is only linear in the number of unknowns of the respective generalized sparse grid discretization. We report on numerical experiments for option pricing with the Kou model in one, two and three dimensions, which demonstrate the optimal complexity of our approach.

1 Introduction

In 1973, Black and Scholes [6] and Merton [22] published their seminal work, which allowed to determine the fair price of an option under a set of certain given assumptions. Here, for some simple options, e.g. European vanillas, analytical formulas exist, but for more complex financial instruments like American put options or arithmetic average Asian options this is no longer the case and numerical approximations must be made. This gave rise to the discipline of computational

M. Griebel · A. Hullmann (✉)
Institute for Numerical Simulation, University of Bonn, Bonn, Germany
e-mail: griebel@ins.uni-bonn.de; hullmann@ins.uni-bonn.de

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_6,

finance, in which most valuation problems are solved by means of Monte Carlo simulations or the numerical solution of partial differential equations.

In recent decades, it has become clear that the assumptions made in the original paper do not hold in practice. This can be seen e.g. from effects like the 'volatility smile', which means that the implied volatility of an option is not independent of the strike price. Thus, observable option prices are not conform with the model assumption that logarithmic returns follow a Brownian motion with constant volatility. Obviously, the normal distribution underestimates the probability of extreme events like the Black Monday in 1987, the crash of Long-Term Capital Management in 1998 or the collapse of Lehman brothers in 2008, just to mention a few. In May 2010, the Dow Jones plunged without obvious reasons by almost 1,000 points, commonly referred to as "Flash Crash". In general, daily returns of six standard deviations can practically be observed in most markets [9], although a market move of that magnitude would theoretically occur only about once in a million years.

By adding jumps to the model of the geometric Brownian motion we can take such effects into account and fix issues like the volatility smile. Even though analytical formulas exist for certain option types on jump processes [20, 23], numerical valuation is nevertheless needed in most practical cases. Then, additional computational difficulties stem from an integral term, which makes the usual Black-Scholes PDE a partial integro-differential equation (PIDE). Due to the non-locality of the integro-operator we obtain linear systems with densely populated matrices, which, treated naively, would result in a substantial additional computational complexity of $\mathcal{O}(N^3)$, with N being the degrees of freedom. Thus, the convolution integral is evaluated by the fast Fourier transform in [1], which reduces the complexity of the system matrix application to $\mathcal{O}(N \log N)$. However, Kou's jump-diffusion model admits an even faster operator application with $\mathcal{O}(N)$ complexity for the finite difference case [29]. In this paper, we introduce a comparable approach for the Galerkin method and exploit it in our numerical solver.

An additional numerical challenge is the pricing of basket options, i.e. options on multiple underlyings. This usually leads to the so-called 'curse of dimensionality', which means that the cost complexity for the approximation to the solution of a problem grows exponentially with the dimension. For example, a d-dimensional mesh with a resolution of h in each direction results in a storage and cost complexity of at least $\mathcal{O}(h^{-d})$. Sparse grid discretizations [3] can circumvent this problem to some extent. They result in a complexity of only $\mathcal{O}(h^{-1} (\log h^{-1})^{d-1})$, which allows for huge savings for higher values of d, whereas – depending on the smoothness assumptions on the function—the convergence rate of the error is unchanged or only affected by a small logarithmic term.

In this paper, we show how generalized sparse grid generating syste ms can be used in the described PIDE setting, i.e. for the pricing of basket options with the Kou model. We use the unidirectional principle [7, 8]—a fast matrix-vector multiplication for sparse grids, which was originally developed for partial differential operators—and generalize it to our non-local operator. In combination with the Galerkin recurrence formula for the Kou model and an optimal

preconditioning based on a H^1-norm equivalence with L_2-orthogonal subspaces, we obtain a method which altogether scales only linearly with respect to the degrees of freedom of our sparse grid discretization.

The remainder of this paper is organized as follows: In Sect. 2 we present Kou's model, its multi-dimensional generalization and we discuss how the pricing of European options on jump-diffusion processes leads to a PIDE problem. Section 3 deals with relevant aspects of the numerical treatment of our PIDE, i.a. optimal preconditioning, the unidirectional principle for non-local operators and the recurrence formula for Kou's model in the Galerkin approach. In Sect. 4 we test our approach on basket put options in one, two and three dimensions. Final remarks in Sect. 5 conclude the paper.

2 Option Pricing with Kou's Model

In this section we describe a one-dimensional model for jump-diffusion processes as presented in [20]. Then, we discuss its generalization to the multi-dimensional setting. Finally, we sketch how a PIDE arises from the option pricing problem on such a jump-diffusion process.

2.1 One-Dimensional Model

Kou's jump-diffusion model [20] assumes that the price of a security S fulfills the stochastic differential equation (SDE)

$$\frac{\mathrm{d}S(t)}{S(t-)} = \mu \mathrm{d}t + \sigma \mathrm{d}W(t) + \mathrm{d}\left(\sum_{j=1}^{N(t)} (V_j - 1)\right), \tag{1}$$

where t denotes time, $W(t)$ is a standard Brownian motion, μ and σ are the usual constants for drift and volatility, $N(t)$ is a Poisson process with rate λ and $\{V_j\}_{j\in\mathbb{N}}$ denotes a sequence of jumps. These jumps are assumed to be independent identically distributed with density

$$h_{p,\eta,\mu}(v) = \begin{cases} (1-p)\eta v^{\eta-1} & \text{for} \quad v < 1, \\ p\mu v^{-\mu-1} & \text{for} \quad v \geq 1, \end{cases}$$

where p and $1-p$ denote the probabilities of jumping upwards and downwards, respectively, while $\eta > 0$ and $\mu > 1$ are parameters that control the jump sizes. The density of $Y_j := \log(V_j)$, $j \in \mathbb{N}$, is then given by

$$\kappa_{p,\eta,\mu}(z) = \begin{cases} (1-p)\eta e^{\eta z} & \text{for} \quad z < 0, \\ p\mu e^{-\mu z} & \text{for} \quad z \geq 0, \end{cases} \tag{2}$$

which is an asymmetric double exponential distribution. Furthermore, in this model all random variables $W(t), N(t), Y_j$ are assumed to be independent. The dynamics of S in the SDE (1) can then be given by

$$S(t) = S(0) \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right) t + \sigma W(t)\right) \cdot \prod_{j=1}^{N(t)} V_j$$

and we have

$$\mathbb{E}(V_j) = \mathbb{E}(\exp(Y_j)) = (1-p)\frac{\eta}{\eta+1} + p\frac{\mu}{\mu-1}.$$

2.2 Multi-dimensional Case and Dependence Modelling

We now consider a d-dimensional price process $\mathbf{S} = (S_1, \ldots, S_d)$ with state space $\mathbb{R}^d$. The components $S_i, i = 1, \ldots, d$, of $\mathbf{S}$ follow the dynamics

$$S_i(t) = S_i(0) \exp\left(\left(\mu_i - \frac{1}{2}\sigma_i^2\right) t + \sigma_i W_i(t)\right) \cdot \prod_{j=1}^{N_i(t)} V_{i,j} \cdot \prod_{j=1}^{\tilde{N}(t)} \tilde{V}_{i,j}, \tag{3}$$

where μ_i and σ_i denote drift and volatility constants, N_i and $\tilde{N}$ are Poisson processes with rates λ_i and $\tilde{\lambda}$, respectively, and $\{\log V_{i,j}\}_{j\in\mathbb{N}}$ and $\{\log \tilde{V}_{i,j}\}_{j\in\mathbb{N}}$ are sequences of jumps of the component i.

Obviously, the Brownian part of the process is decorrelated across dimensions. A priori, this assumption may look unrealistic, it however can easily be achieved by a standard principal component transformation of the covariance matrix, see [26] for example.

In our model (3), every price process S_i has two sources of jumps. The first one leaves other processes $S_{i'}, i' \neq i$ unaffected. It is realized by the Poisson process $N_i(t)$ and i.i.d. double exponential random variables $\{\log V_{i,j}\}_{j\in\mathbb{N}}$ with parameters p_i, η_i, μ_i, see (2). The second one takes care of the correlations of jumps over the dimensions. It consists of a jump process $\tilde{N}(t)$ and the associated random variables $\{\log \tilde{V}_{i,j}\}_{j\in\mathbb{N}}$, which are again assumed to be i.i.d. double exponential random variables with parameters $\tilde{p}_i, \tilde{\eta}_i, \tilde{\mu}_i$. It is noteworthy that even though the jump sizes are independent, dependence of the d price processes $(S_1, \ldots, S_d)$ is created nevertheless, as jumps happen at the same time in *all* components of $\mathbf{S}$. This joint jump term can be understood as a risk factor affecting the whole market and not just a single company or industrial sector.

2.3 Representation of the Multi-dimensional Process as Lévy Process

We now identify our d-dimensional generalization (3) of the Kou model as a Lévy process. To this end, recall the following definition, compare also [9].

Definition 1 (Lévy process). A càdlàg stochastic process $(\mathbf{X}(t))_{0\le t<\infty}$ on the filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{0\le t<\infty}, \mathbb{P})$ with values in $\mathbb{R}^d$ and $\mathbf{X}(0) = \mathbf{0}$ a.s. is called a *Lévy process* if it has the following properties:

1. Independent increments: for every sequence $t_0 < t_1 < \cdots < t_n$, the random variables $\mathbf{X}(t_0), \mathbf{X}(t_1) - \mathbf{X}(t_0), \ldots, \mathbf{X}(t_n) - \mathbf{X}(t_{n-1})$ are independent.
2. Stationary increments: $\mathbf{X}(t) - \mathbf{X}(s)$ has the same distribution as $\mathbf{X}(t-s)$, $0 \le s < t < \infty$.
3. Stochastic continuity: $\lim_{t\to s} \mathbf{X}(t) = \mathbf{X}(s)$, where the limit is taken in probability.

The characteristic exponent $\psi : \mathbb{R}^d \to \mathbb{C}$ of $\mathbf{X}(t)$, which satisfies

$$\mathbb{E}\left(e^{i\langle \xi, \mathbf{X}(t)\rangle}\right) = e^{t\psi(\xi)} \quad \text{for} \quad \xi \in \mathbb{R}^d, t \ge 0,$$

allows for the unique Lévy-Khinchin representation

$$\psi(\xi) = -\frac{1}{2}\langle \xi, \mathbf{Q}\xi\rangle + i\,\langle \gamma, \xi\rangle + \int_{\mathbb{R}^d} \left(e^{i\langle \xi, \mathbf{z}\rangle} - 1 - i\,\langle \xi, \mathbf{z}\rangle \mathbb{1}_{\{|\mathbf{z}|\le 1\}}\right) \nu(\mathrm{d}\mathbf{z}),$$

where $\mathbf{Q} \in \mathbb{R}^{d\times d}$ is the symmetric covariance matrix of the continuous part of $\mathbf{X}$, $\gamma \in \mathbb{R}^d$ is the drift of $\mathbf{X}$ and ν is the Lévy measure, which satisfies

$$\int_{\mathbb{R}^d} \min(1, |\mathbf{z}|^2)\nu(\mathrm{d}\mathbf{z}) < \infty.$$

This condition ensures that the activity for large jumps is finite and possibly infinite for very small jumps only.

For our multi-dimensional Kou model presented in Sect. 2.2, the logarithmic returns follow a Lévy process. We thus can rewrite (3) as

$$S_i(t) = S_i(0)\exp\left(X_i(t)\right) \quad \text{for } i = 1, \ldots, d,$$

with $(\mathbf{X}(t))_{0\le t<\infty}$ being a $\mathbb{R}^d$-valued Lévy process with characteristic triplet $(\mathbf{Q}, \nu, \gamma)$. Here, the elements of covariance matrix $\mathbf{Q} \in \mathbb{R}^{d\times d}$ satisfy

$$q_{ij} = \delta_{ij}\sigma_i^2$$

for all $i, j = 1, \ldots, d$, the drift is

$$\gamma_i = \mu_i - \frac{\sigma_i^2}{2} + \int_{\mathbb{R}^d} z_i \mathbb{1}_{\{|z|\leq 1\}} \nu(\mathrm{d}z)$$

for $i = 1, \ldots, d$, and the Lévy measure ν has finite activity and can be expressed by

$$\nu(\mathrm{d}z) = \sum_{i=1}^{d} \lambda_i \kappa_{p_i,\eta_i,\mu_i}(z_i)\mathrm{d}z_i \otimes \bigotimes_{\substack{i'=1 \\ i'\neq i}}^{d} \delta(\mathrm{d}z_{i'}) + \tilde{\lambda} \bigotimes_{i=1}^{d} \kappa_{\tilde{p}_i,\tilde{\eta}_i,\tilde{\mu}_i}(z_i)\mathrm{d}z_i. \tag{4}$$

In (4), the i-th summand represents the jumps in the i-th component of $\mathbf{X}$ generated by the Poisson process $N_i(t)$ and the associated jump sizes $(\log V_{i,j})_{j\in\mathbb{N}}$ are distributed with density $\kappa_{p_i,\eta_i,\mu_i}$, see (2). The remaining components are unaffected by these jumps, which is expressed by the delta distribution $\bigotimes_{i'\neq i} \delta(\mathrm{d}z_{i'})$. The last term in (4) is due to the Poisson process $\tilde{N}(t)$ with jumps occurring in all components at the same time. The jump sizes $(\log \tilde{V}_{i,j})_{j\in\mathbb{N}}$ in the i-th component are distributed with the probability density $\kappa_{\tilde{p}_i,\tilde{\eta}_i,\tilde{\mu}_i}$.

Remark 1. Even though the jump sizes are all independent, the stochastic processes X_i, and thus S_i, are not. Otherwise, the right-hand side of

$$\mathbb{E}\left(e^{i\langle\xi,\mathbf{X}(t)\rangle}\right) = e^{-t\psi(\xi)}$$

could be written as a product in $\xi_1, \ldots, \xi_d$, which is not possible since ψ does in general *not* decompose into a sum of one-dimensional functions over the dimensions.

Note here that the product $\bigotimes_{i=1}^{d} \kappa_{\tilde{p}_i,\tilde{\eta}_i,\tilde{\mu}_i}(z_i)$ in (4) can be seen as a rank-one approximation to a general finite Lévy measure. Then, this concept can easily be carried over to more complex dependencies by adding further Poisson processes in combination with jump sizes that are again independent of other dimensions. This can be done in the spirit of a low rank approximation, see [5], i.e. as the approximation of a non-separable density function by a small sum of separable functions. Such a generalization would be treatable by the numerical approach proposed in this paper as well.

2.4 Option Pricing

We now want to price a European option with the payoff $g : \mathbb{R}^d \to \mathbb{R}$ depending on the state of the process $\mathbf{S}$ at the time of maturity T. In this paper, we predominantly price put options with strike price K and unit weights, i.e.

$$g(\mathbf{S}(T)) = \max(0, K - \sum_{i=1}^{d} S_i(T)). \tag{5}$$

Note, however, that the same approach can also be used to valuate call options. For the conventional Black–Scholes model, arbitrage considerations show that the fair price of a European option is given by the discounted expected value of the payoff function under the unique risk-neutral measure. Jump-diffusion models lead to incomplete markets and a risk-neutral measure needs to be selected by, e.g., the rational expectations equilibrium, see [21]. We assume that the risk-neutral dynamics of our d price processes is given by

$$S_i(t) = S_i(0) \exp(rt + X_i(t)) \quad \text{for} \quad i = 1, \ldots, d, \tag{6}$$

where $\mathbf{X}$ is a Lévy process with a triplet $(\mathbf{Q}, \nu, \gamma)$, s.t.

$$e^{-tr} S_i(t)$$

are martingales for $i = 1, \ldots, d$. Then, the value of a European option at time t with payoff g can be given by

$$V(t, \mathbf{s}) = \mathbb{E}(e^{-r(T-t)} g(\mathbf{S}(T)) \mid \mathbf{S}(t) = \mathbf{s}). \tag{7}$$

This function V is known to satisfy the PIDE (8) of the following theorem, see [9, 27] for further information.

Theorem 1 (Backward PIDE for European options). *Let* $\mathbf{S}$ *be an exponential Lévy model* (6) *with Lévy triplet* $(\mathbf{Q}, \nu, \gamma)$*, which has a non-vanishing diffusion matrix* $\mathbf{Q}$*, and let* ν *satisfy*

$$\int_{|\mathbf{z}| \geq 1} e^{z_i} \nu(\mathrm{d}\mathbf{z}) < \infty$$

for $i = 1, \ldots, d$*. Then, the function*

$$V \in C^{1,2}\left((0, T) \times \mathbb{R}^d_{>0}\right) \cap C^0\left([0, T] \times \mathbb{R}^d_{\geq 0}\right)$$

given by (7) *is a solution of the backward PIDE for European options*

$$\begin{aligned} &\frac{\partial V}{\partial t}(t, \mathbf{s}) + \frac{1}{2} \sum_{i,j=1}^{d} s_i s_j q_{ij} \frac{\partial^2 V}{\partial s_i \partial s_j}(t, \mathbf{s}) + r \sum_{i=1}^{d} s_i \frac{\partial V}{\partial s_i}(t, \mathbf{s}) - rV(t, \mathbf{s}) \\ &+ \int_{\mathbb{R}^d} \left(V(t, \mathbf{s}e^{\mathbf{z}}) - V(t, \mathbf{s}) - \sum_{i=1}^{d} s_i \left(e^{z_i} - 1\right) \frac{\partial V}{\partial s_i}(t, \mathbf{s}) \right) \nu(\mathrm{d}\mathbf{z}) = 0 \end{aligned} \tag{8}$$

on $(0,T) \times \mathbb{R}^d_{>0}$, *where* $V(t, \mathbf{s}e^{\mathbf{z}}) := V(t, s_1 e^{z_1}, \ldots, s_d e^{z_d})$, *with the terminal condition given by*

$$V(T, \mathbf{s}) = g(\mathbf{s}) \quad \textit{for} \quad \mathbf{s} \in \mathbb{R}^d_{\geq 0}.$$

Under the same constraints as in Theorem 1 and with the assumption of finite activity, i.e. $\lambda := \nu(\mathbb{R}^d) < \infty$, which holds in case of the Kou model (3) with $\lambda = \sum_{i=1}^d \lambda_i + \tilde{\lambda}$, we can rewrite the PIDE as

$$\frac{\partial}{\partial \tau} u(\tau, \mathbf{x}) - \frac{1}{2} \sum_{i,j=1}^{d} q_{ij} \frac{\partial^2}{\partial x_i x_j} u(\tau, \mathbf{x}) + \lambda u(\tau, \mathbf{x}) - \int_{\mathbb{R}^d} u(\tau, \mathbf{x} + \mathbf{z}) \nu(\mathrm{d}\mathbf{z}) = 0 \quad (9)$$

with $e^{-r\tau} u(\tau, \mathbf{x}) = V(t, \mathbf{s})$. To this end, we use the variable transforms $\tau = T - t$ and

$$x_i = \log s_i + \tau (r - \xi_i - \frac{q_{ii}}{2}) \text{ with } \xi_i = \int_{\mathbb{R}^d} (e^{z_i} - 1)\, \nu_i(\mathrm{d}z_i) \quad (10)$$

for $i = 1, \ldots, d$, with ν_i being the marginal distribution of ν in the i-th direction. Then, the terminal condition becomes an initial condition for $\tau = 0$, and reads

$$u(0, \mathbf{x}) = g(\exp(\mathbf{x}))$$

with the exp-function applied componentwise to the vector $\mathbf{x} \in \mathbb{R}^d$.

3 Numerical Treatment

In this section, we discuss the numerical treatment of the transformed PIDE (9). We start with a simple θ-scheme for time discretization and a Galerkin method using generalized sparse grids for space discretization. The resulting set of linear equations is then solved by an optimally preconditioned iterative method, which we discuss in Sect. 3.3. To this end, we present a fast matrix-vector multiplication for the sparse grid generating systems in combination with non-local operators and the Galerkin recurrence formula for the Kou model. All these ingredients are essential to obtain an overall solution method with optimal linear complexity in the degrees of freedom of our discretization.

3.1 Time Discretization and Weak Formulation

First, we localize our space domain $\mathbb{R}^d$ to a rectangular domain

$$\mathcal{D} = (\alpha_1, \beta_1) \times \cdots \times (\alpha_d, \beta_d) \quad (11)$$

and assume zero boundary conditions. This truncation can be understood as an approximation of the price of the option by that of a barrier option, i.e. as an option that has no payoff if the underlying price process has left $\mathcal{D}$ during the time to maturity. It is well-known that the pointwise error introduced by this approximation decreases exponentially with the domain size, see for example [10].

The upper and lower bounds in (11) are chosen by

$$\alpha_i = x_i - \gamma\sqrt{T \cdot \mathrm{var}(X_i(T))}, \tag{12}$$

$$\beta_i = x_i + \gamma\sqrt{T \cdot \mathrm{var}(X_i(T))}, \tag{13}$$

where γ is a proportionality constant relating the domain size and the standard deviation of the stochastic process in that direction. Furthermore, $\mathbf{x} = (x_1 \ldots, x_d)$ denotes a point of interest, which might be, like e.g. in (10), a transformed initial state vector $\mathbf{s}$ for an option we want to price by the evaluation of $e^{-rT}u(T, \mathbf{x}) = V(0, \mathbf{s})$.

We then can write the PIDE (9) with the definition

$$Lu(\tau, \mathbf{x}) := \frac{1}{2}\sum_{i,j=1}^{d} q_{ij}\frac{\partial^2}{\partial x_i x_j}u(\tau, \mathbf{x}) - \lambda u(\tau, \mathbf{x}) + \int_{\mathbb{R}^d} u(\tau, \mathbf{x} + \mathbf{z})\nu(\mathrm{d}\mathbf{z})$$

as

$$\frac{\partial u(\tau, \mathbf{x})}{\partial \tau} - Lu(\tau, \mathbf{x}) = 0 \qquad \text{on} \qquad (0, T) \times \mathcal{D}, \tag{14}$$

with an extension of u by zero outside of $\mathcal{D}$, i.e. $u|_{(0,T)\times\mathbb{R}^d\setminus\mathcal{D}} = 0$, which is relevant to the integral term.

For the time discretization, we subdivide the interval $[0, T]$ into $M+1$ equidistant time-steps

$$t_i = i\Delta t, \qquad i = 0, \ldots, M,$$

with $\Delta t = \frac{T}{M}$ and apply the well-known θ-scheme with $\theta = \frac{1}{2}$.[1] Then, (14) can be expressed for $i = 0, \ldots, M-1$ as a sequence of stationary elliptic PIDEs

$$\frac{u^{(i+1)}(\mathbf{x}) - u^{(i)}(\mathbf{x})}{\Delta t} - L\left(\theta u^{(i+1)}(\mathbf{x}) + (1-\theta)u^{(i)}(\mathbf{x})\right) = 0$$

on $\mathcal{D}$ with $u^{(i)}(\mathbf{x}) \approx u(t_i, \mathbf{x}), i = 0, \ldots, M$.

Next, a weak formulation in space will give us a sequence of $H_0^1(\mathcal{D})$-elliptic problems: Find $u^{(i+1)} \in H_0^1(\mathcal{D})$, s.t.

$$\Delta t^{-1}(u^{(i+1)}, v)_{L_2} + \theta a(u^{(i+1)}, v) = r^{(i)}(v) \quad \forall v \in H_0^1(\mathcal{D}) \tag{15}$$

[1] More sophisticated techniques, e.g. space-time sparse grids [17], are available, but this is beyond the scope of this work.

with

$$a(u,v) = \frac{1}{2}\int_{\mathcal{D}} \sum_{i,j=1}^{d} q_{ij}\frac{\partial u(\mathbf{x})}{\partial x_i}\frac{\partial v(\mathbf{x})}{\partial x_j}\mathrm{d}\mathbf{x} + \lambda(u,v)_{L_2} - \int_{\mathcal{D}}\int_{\mathbb{R}^d} u(\tau,\mathbf{x}+\mathbf{z})\nu(\mathrm{d}\mathbf{z})v(\mathbf{x})\mathrm{d}\mathbf{x}$$

and

$$r^{(i)}(v) = \Delta t^{-1}(u^{(i)},v)_{L_2} - a((1-\theta)u^{(i)},v).$$

Before Eq. (15) can be discretized in space, we need to transform the domain $\mathcal{D}$ to $\Omega := (0,1)^d$, see (11). This is done by a linear affine scaling $\mathcal{T}: \mathcal{D} \to \Omega$

$$\mathcal{T}(\mathbf{x}) = \left(\frac{x_1-\alpha_1}{\beta_1-\alpha_1},\ldots,\frac{x_d-\alpha_d}{\beta_d-\alpha_d}\right). \tag{16}$$

When we apply the domain transformation (16), and take the assumptions made about $(q_{ij})_{i,j=1}^d$ and ν in Sect. 2.2 into account, the bilinear form $a(u,v)$ in the variational equation (15) can be given on Ω by

$$a(u,v) = \sum_{i=1}^{d}\frac{\sigma_i^2}{2(\beta_i-\alpha_i)^2}\int_{\Omega}\frac{\partial u(\mathbf{x})}{\partial x_i}\frac{\partial v(\mathbf{x})}{\partial x_i}\mathrm{d}\mathbf{x} + \lambda\int_{\Omega} u(\mathbf{x})v(\mathbf{x})\mathrm{d}\mathbf{x} \tag{17}$$

$$-\sum_{i=1}^{d}(\beta_i-\alpha_i)\lambda_i\int_{\Omega}\int_{\mathbb{R}^d} u(\mathbf{x}+z_i\mathbf{e}_i)\kappa_{p_i,\eta_i,\mu_i}((\beta_i-\alpha_i)z_i)\mathrm{d}z_i\, v(\mathbf{x})\mathrm{d}\mathbf{x}$$

$$-\tilde{\lambda}\int_{\Omega}\int_{\mathbb{R}^d} u(\mathbf{x}+\mathbf{z})\prod_{i=1}^{d}(\beta_i-\alpha_i)\kappa_{\tilde{p}_i,\tilde{\eta}_i,\tilde{\mu}_i}((\beta_i-\alpha_i)z_i)\mathrm{d}z_i\ v(\mathbf{x})\mathrm{d}\mathbf{x}, \tag{18}$$

where $\mathbf{e}_i$ denotes the i-th unit vector and

$$\lambda := \sum_{i=1}^{d}\lambda_i(\beta_i-\alpha_i) + \tilde{\lambda}\prod_{i=1}^{d}(\beta_i-\alpha_i).$$

3.2 Space Discretization by a Sparse Grid Generating System

What is finally missing is a discretization of (15) with bilinear form (17) in space. To this end, we introduce a sparse grid generating system based on linear splines. First, we consider a one-dimensional multilevel system on the interval $(0,1)$. On

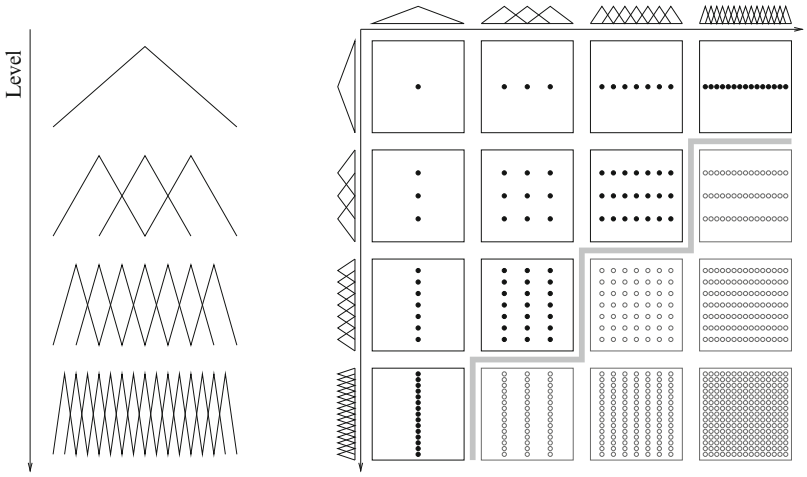


Fig. 1 The first four levels of a one-dimensional multilevel generating system (*left*). Two-dimensional tensorization and the sparse subspace (*right*)

level l, $n_l = 2^l - 1$ hat functions

$$\phi_{l,i}(x) = \max(1 - 2^l \, |x - x_{l,i}| \, , 0)$$

exist, which are centered at the points of an equidistant mesh

$$x_{l,i} = 2^{-l} i$$

for $i = 1, \ldots, n_l$. The left side of Fig. 1 shows all functions on the first four levels. The spaces

$$V_l = \operatorname{span}\{\phi_{l,i} \mid 1 \leq i \leq n_l\}, l \in \mathbb{N}$$

possess the simple inclusion relation $V_l \subset V_{l+1}$. This makes the union of their basis functions $\bigcup_{l=1}^{k} \{\phi_{l,1}, \ldots, \phi_{l,n_l}\}$ a generating system and not a basis. The multi-dimensional case is based on the domain $\Omega = (0,1)^d$. By tensorization, we obtain spaces associated to a multi-index $\mathbf{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$

$$V_{\mathbf{l}} = \bigotimes_{i=1}^{d} V_{l_i}.$$

For a given $\mathbf{l}$, $V_{\mathbf{l}}$ is the space of d-linear functions on a possibly anisotropic full grid space with

$$|V_{\mathbf{l}}| = \prod_{i=1}^{d} (2^{l_i} - 1) = \mathcal{O}(2^{|\mathbf{l}|_1})$$

degrees of freedom. It is spanned by the functions

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) = \phi_{l_1,i_1}(x_1) \cdots \phi_{l_d,i_d}(x_d)$$

with

$$\mathbf{i} \in \chi_{\mathbf{l}} := \{\mathbf{i} = (i_1, \ldots, i_d) \in \mathbb{N}^d : 1 \leq i_j \leq n_j, j = 1, \ldots, d\}.$$

The space

$$\tilde{V} = \sum_{\mathbf{l} \in \mathbb{N}^d} V_{\mathbf{l}}$$

is equal to the underlying Sobolev space $H_0^1(\Omega)$ up to completion with the H^1-norm, see [3]. For our numerical computation, we have to resort to a finite-dimensional subset. To this end, we use an index set $\mathcal{I} \subset \mathbb{N}^d, |\mathcal{I}| < \infty$, which defines the subspaces included in the discretization as

$$V_{\mathcal{I}} = \sum_{\mathbf{l} \in \mathcal{I}} V_{\mathbf{l}}.$$

A proper choice of $\mathcal{I}$ now depends—besides the error we want to achieve—on the smoothness of the function class we want to approximate. For example, the full grid space with index set

$$\mathcal{I}_k^{(\infty)} = \{\mathbf{l} \in \mathbb{N}^d : |\mathbf{l}|_\infty \leq k\}$$

has the approximation property

$$\inf_{v \in V_{\mathcal{I}_k^{(\infty)}}} \|u - v\|_{H^s(\Omega)}^2 \leq c 2^{-2(t-s)k} \|u\|_{H^t(\Omega)}^2$$

with rate[2] $t - s$ and $u \in H_0^t(\Omega)$. Its number of degrees of freedom grows by $\mathcal{O}(2^{kd})$. Thus the accuracy as function of the degrees of freedom deteriorates exponentially with rising d, which resembles the well-known 'curse of dimensionality', cf. [2].

Assuming additional mixed smoothness $u \in H_{0,\text{mix}}^t(\Omega)$, the sparse grid index set

$$\mathcal{I}_k^{(1)} = \{\mathbf{l} \in \mathbb{N}^d : |\mathbf{l}|_1 \leq k + d - 1\} \tag{19}$$

circumvents this problem to some extent. The rate of the best approximation in dependence of k

$$\inf_{v \in V_{\mathcal{I}_k^{(1)}}} \|u - v\|_{H^s(\Omega)}^2 \leq c 2^{-2(t-s)k} \|u\|_{H_{\text{mix}}^t(\Omega)}^2$$

[2] This holds for a range of parameters $0 \leq s < t \leq r$ with r being the order of the spline of the space construction. In our case of linear splines $r = 2$ holds.

is the same[3] as for the full grid space, i.e. $t-s$, but the number of degrees of freedom now only grows by $\mathcal{O}(2^k k^{d-1})$. This is a substantial improvement in comparison to the full grid case. For further details, see [14]. A related approach, adaptive sparse grids based on a linear spline basis, has been employed for option pricing without jumps in [4].

Next, to compensate for the transformation (16), we want our discretization to have a refinement level in each dimension $i = 1, \ldots, d$, which is logarithmically proportional to the size $\beta_i - \alpha_i$ of the respective dimension in $\mathcal{D}$. This then leads to anisotropic sparse grids. First, we need to determine the level shifts

$$\rho_i = \left\lceil \log_2 \frac{\beta_1 - \alpha_1}{\beta_i - \alpha_i} \right\rceil + 1$$

for $i = 1, \ldots, d$, that are necessary to compensate for the former anisotropies. Then, the index set

$$\mathcal{I}_k^{(*)} = \{\mathbf{l} \in \mathbb{N}^d : \frac{l_1 - 1}{\max(k - \rho_1, 1)} + \cdots + \frac{l_d - 1}{\max(k - \rho_d, 1)} \leq 1\} \tag{20}$$

gives us an anisotropic sparse grid on level k, with the classical sparse grid as a special case for $\rho_1 = \cdots = \rho_d = 1$. Anisotropies may reduce the sparse grid cost complexity even further, see [13] for a discussion in this direction.

Remark 2. It is furthermore possible to adapt the index set $\mathcal{I}$ a-posteriori to a given function by means of a proper error estimation and successive refinement procedure. This approach results in adaptively refined sparse grids, see e.g. [11, 12]. For algorithmic reasons, we then need the additional condition

$$\mathbf{l} \in \mathcal{I}, \mathbf{k} \in \mathbb{N}^d, \mathbf{k} \leq \mathbf{l} \Rightarrow \mathbf{k} \in \mathcal{I}, \tag{21}$$

where $\mathbf{k} \leq \mathbf{l}$ is understood componentwise.

Now we present the final equations that are being solved numerically for the parameters and model assumptions specified in Sect. 2.2, where also the domain transformation (16) is being considered. For any valid index set $\mathcal{I} \subset \mathbb{N}^d$, let us denote by $\mathbf{u} = (b_{\mathbf{l},\mathbf{i}})_{\mathbf{l}\in\mathcal{I},\mathbf{i}\in\chi_\mathbf{l}}$ a block vector of $N := \sum_{\mathbf{l}\in\mathcal{I}} |\chi_\mathbf{l}|$ coefficients used to represent functions

$$u(\mathbf{x}) = \sum_{\mathbf{l}\in\mathcal{I}} \sum_{\mathbf{i}\in\chi_\mathbf{l}} u_{\mathbf{l},\mathbf{i}} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) \in V_\mathcal{I}.$$

[3] For $s = 0$ an additional logarithmic term appears in the error estimate.

Then, our variational problem (15) can be discretized as follows: For $i = 0, \ldots, M-1$ find $\mathbf{u}^{(i+1)}$ s.t.

$$\left(\Delta t^{-1}\mathbf{M} + \theta\mathbf{A}\right)\mathbf{u}^{(i+1)} = \left(\Delta t^{-1}\mathbf{M} + (\theta-1)\mathbf{A}\right)\mathbf{u}^{(i)}, \tag{22}$$

with the block-structured mass matrix $\mathbf{M} \in \mathbb{R}^{N\times N}$

$$(\mathbf{M})_{(\mathbf{l},\mathbf{i}),(\mathbf{k},\mathbf{j})} = \int_{\Omega} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x})\phi_{\mathbf{k},\mathbf{j}}(\mathbf{x})d\mathbf{x}$$

and the stiffness matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$

$$(\mathbf{A})_{(\mathbf{l},\mathbf{i}),(\mathbf{k},\mathbf{j})} = a(\phi_{\mathbf{l},\mathbf{i}}, \phi_{\mathbf{k},\mathbf{j}})$$

with the bilinear form $a(\cdot,\cdot)$ from (17). The initial value $\mathbf{u}^{(0)}$ is set to the L_2-projection of the payoff function into the space $V_{\mathcal{I}}$.

Note here that the representation of functions in $V_{\mathcal{I}} = \sum_{\mathbf{l}\in\mathcal{I}} V_{\mathbf{l}}$ is not unique, since all subspaces $V_{\mathbf{l}}, \mathbf{l} \in \mathcal{I}$ appear in our discretization. This means that operator matrices $\mathbf{A}$ and $\mathbf{M}$ for this discretization have a non-trivial kernel and the system matrix cannot be inverted. However, Krylov subspace methods will converge as long as the right-hand side is within the range of the operator matrix, see [18, 19] for further details.

3.3 Preconditioning

We now want to find an optimal preconditioner for our linear system (22). To this end, we need to bound the quotient $\tilde{\lambda}_{\max}(\mathbf{A})/\tilde{\lambda}_{\min}(\mathbf{A})$ independently of $\dim V_{\mathcal{I}}$, where

$$\tilde{\lambda}_{\min}(\mathbf{A}) = \min_{\mathbf{u}\perp\ker(\mathbf{A})} \frac{\langle \mathbf{A}\mathbf{u}, \mathbf{u}\rangle}{\|\mathbf{u}\|_2^2}$$

and

$$\tilde{\lambda}_{\max}(\mathbf{A}) = \max_{\mathbf{u}\perp\ker(\mathbf{A})} \frac{\langle \mathbf{A}\mathbf{u}, \mathbf{u}\rangle}{\|\mathbf{u}\|_2^2}$$

denote the minimum and the maximum of the Rayleigh-quotient restricted to the subspace $\{\mathbf{u} : \mathbf{u} \perp \ker(\mathbf{A})\}$.

The preconditioning of generating systems on regular sparse grid spaces with index set (19) is discussed in [15, 16] within the framework of multilevel subspace splittings. A simple diagonal scaling results in condition numbers that are bounded by $\mathcal{O}(k^{d-2})$, which is a substantial improvement, but does not give an optimal method yet.

To obtain optimal condition numbers of $\mathcal{O}(1)$ we proceed as follows: We rely on the stable subspace splitting

$$\{H^1; \|\cdot\|_{H^1}^2\} = \sum_{\mathbf{l}} \{W_{\mathbf{l}}; 2^{2|\mathbf{l}|_\infty} \|\cdot\|_{L_2}^2\}, \tag{23}$$

where $\mathbf{l} \in \mathbb{N}^d$ denotes a d-dimensional multi-index and $W_{\mathbf{l}}$ is the respective L_2-orthogonal complement space, see [14]. As the nodal basis on level $\mathbf{l}$ is a stable splitting for $V_{\mathbf{l}}$ with a condition number independent of $\mathbf{l}$ and $W_{\mathbf{l}} \subset V_{\mathbf{l}}$, we can derive an optimal preconditioner by splitting a function into its orthogonal complements and then representing them in our generating system subspaces $V_{\mathbf{l}}$.

To explain this in more detail, let us first describe the case of a one-dimensional discretization up to level k, i.e. $\mathcal{I} = \{(l) : 1 \le l \le k\}$ and $d = 1$. Let $Q_l : V_k \to V_l$ be the L_2-orthogonal projection to V_l and $Q_0 = 0$. Because of the nestedness

$$V_l \subset V_{l'} \quad \text{for} \quad l \le l',$$

the telescopic expansion

$$u_k = \sum_{l=1}^{k} (Q_l - Q_{l-1}) u_k, \qquad u_k \in V_k$$

is an orthogonal decomposition with $(Q_l - Q_{l-1})u_k \in W_l$.

The multi-dimensional case can be obtained by tensorization arguments. Then, for $u_{\mathcal{I}} \in V_{\mathcal{I}}$ with an index set $\mathcal{I} \subset \mathbb{N}^d$ satisfying (21), the expression

$$w_{\mathbf{l}} = \sum_{\mathbf{m} \in \{0,1\}^d} (-1)^{|\mathbf{m}|_1} Q_{l_1 - m_1, \ldots, l_d - m_d} u_{\mathcal{I}} \tag{24}$$

denotes the orthogonal complement of $u_{\mathcal{I}}$ in the space $W_{\mathbf{l}}, \mathbf{l} \in \mathcal{I}$.

The algorithmic implementation is straightforward. We first have to compute the projections by

$$(Q_{\mathbf{l}} u_{\mathcal{I}}, v_{\mathbf{l}})_{L_2} = (u_{\mathcal{I}}, v_{\mathbf{l}})_{L_2} \quad \text{for all} \quad v_{\mathbf{l}} \in V_{\mathbf{l}}, \quad \mathbf{l} \in \mathcal{I}. \tag{25}$$

To this end, all right-hand sides for $\mathbf{l} \in \mathcal{I}$ are extracted from the result of one single mass matrix multiplication applied to $u_{\mathcal{I}}$. The operator matrix on the left-hand side of (25) is just a d-fold tensor-product of one-dimensional mass matrices, which themselves are tridiagonal, and thus is easily invertible in $\mathcal{O}(2^{|\mathbf{l}|_1})$ operations for a fixed $\mathbf{l}$. Secondly, after having calculated all discrete functions $Q_{\mathbf{l}} u_{\mathcal{I}}$, we have to sum them up according to (24). This can also be done in $\mathcal{O}(2^{|\mathbf{l}|_1})$ operations,

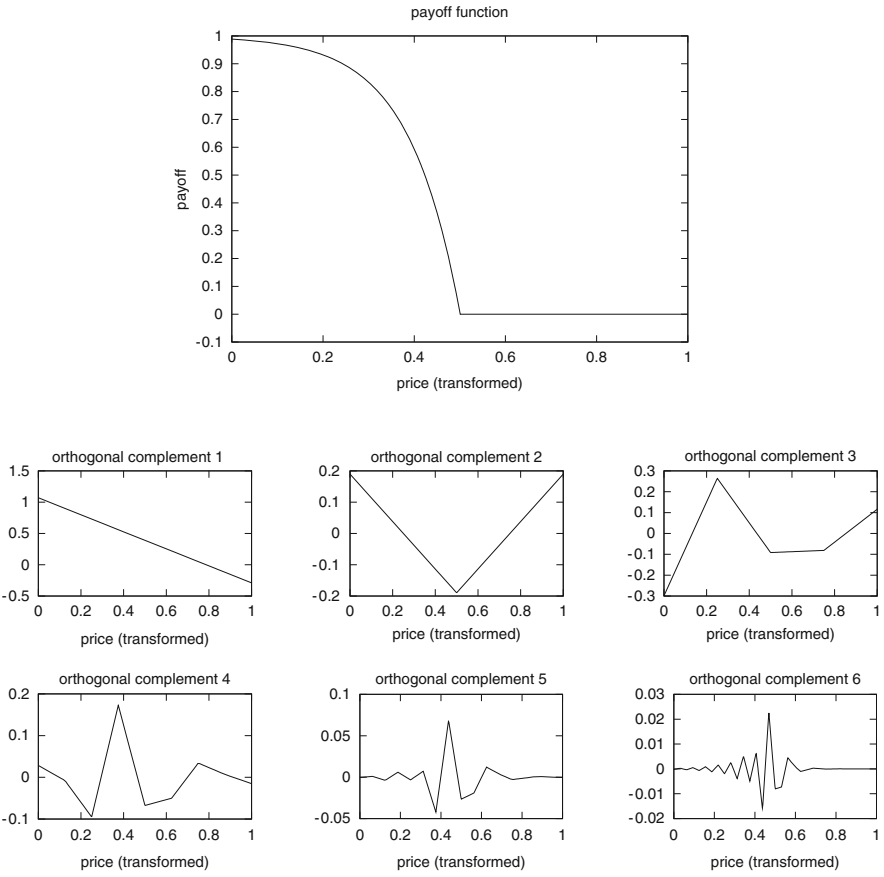


Fig. 2 The payoff function of a one-dimensional put option in the log-space (*top*). The first six orthogonal complements of this function (*middle* and *bottom*)

which is again linear in the degrees of freedom. After scaling with the **l**-dependent weights in the norm equivalence (23) we then obtain an optimal preconditioner with a condition number bounded independently of the space $V_{\mathcal{I}}$. Note that this is achieved without explicitly discretizing the subspaces $W_{\mathbf{l}}$ and the overall system matrix with prewavelets or similar, more complicated basis functions, but merely by sticking to the more simple and natural generating system. A one-dimensional example[4] is given in Fig. 2. We observe that the oscillations on finer scales occur mainly at the position of the non-differentiable kink.

[4]The discretization in this example includes boundary functions otherwise not used for computation.

3.4 Operator Application

Within such an iterative solver the matrix-vector multiplication must be frequently processed. Here, a complexity issue may occur due to the non-trivial interactions between the different subspaces involved in the generating system approach and, additionally, due to the integro-operator in the stiffness matrix. We now describe how a general linear tensor product operator[5]

$$A^{(1)} \otimes A^{(2)} \otimes \cdots \otimes A^{(d)} \tag{26}$$

can be applied to a sparse grid approximation $u_{\mathcal{I}} = [u_{\mathbf{l}}]_{\mathbf{l}\in\mathcal{I}}$ of u using a number of operations, which is of optimal order, e.g. of $\mathcal{O}(2^k k^{d-1})$ for regular sparse grids (19). Consequently, all sums of tensor-product operators can then also be applied with optimal cost complexity.

If the index set $\mathcal{I}$ of our sparse grid would allow a representation $\mathcal{I} = I_1 \times \cdots \times I_d$ with $I_i \subset \mathbb{N}, i = 1, \ldots, d$, we could easily apply the tensor-product operator (26) dimension by dimension. But since the index $\mathcal{I}$ can in general not be represented as a simple cross product, we need a sophisticated algorithm known as the unidirectional principle to maintain optimal complexity. Our aim is to calculate

$$\mathbf{v}_{\mathbf{l}} = \sum_{\mathbf{l}'\in\mathcal{I}} \left(\mathbf{A}^{(1)}_{l_1,l_1'} \otimes \mathbf{A}^{(2)}_{l_2,l_2'} \otimes \cdots \otimes \mathbf{A}^{(d)}_{l_d,l_d'} \right) \mathbf{u}_{\mathbf{l}'}$$

for all $\mathbf{l} \in \mathcal{I}$, where $\mathbf{A}^{(j)}_{l_j,l_j'}$, $j = 1, \ldots, d$, denotes the operator matrix of $A^{(j)}$ in the weak formulation for discretization level l_j' and test function level l_j. First, let us investigate the one-dimensional case.[6] Up to the level k, we have

$$\begin{aligned} \mathbf{v}_l &= \sum_{l'=1}^{k} \mathbf{A}_{l,l'} \mathbf{u}_{l'} \\ &= \sum_{l'=1}^{l} \mathbf{A}_{l,l'} \mathbf{u}_{l'} + \sum_{l'=l+1}^{k} \mathbf{A}_{l,l'} \mathbf{u}_{l'}. \end{aligned} \tag{27}$$

Both sums in (27) need to be treated separately. To this end, we introduce the matrices $\mathbf{I}_{l,l'}$ which serve as restriction operators for $l < l'$ and as prolongation operators for $l > l'$. The matrices $\mathbf{A}_{l,l'}$ can then be written as

[5] Note that the numbers $(1), \ldots, (d)$ in the exponents are no powers but indices, indicating that different operations may be carried out in different dimensions.

[6] For notational convenience we drop the dimension index $^{(j)}$ in $\mathbf{A}^{(j)}_{l_j,l_j'}$ in the following calculations.

$$\mathbf{A}_{l,l'} = \begin{cases} \mathbf{A}_{l,l}\mathbf{I}_{l,l'} & \text{for} \quad l \geq l', \\ \mathbf{I}_{l,l'}\mathbf{A}_{l',l'} & \text{for} \quad l < l'. \end{cases}$$

In the following two algorithms, the prolongations and restrictions are of central importance to efficiently transport intermediate results.

3.4.1 BottomUp Algorithm

The BottomUp algorithm calculates all sums

$$\mathbf{v}_l = \sum_{l'=l+1}^{k} \mathbf{A}_{l,l'}\mathbf{u}_{l'} \tag{28}$$

for $l = 1, \ldots, k$ in linear time. If we express these operations (28) by means of a matrix including all levels, we obtain an upper diagonal form

$$\begin{pmatrix} 0 & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} & \ldots & \mathbf{A}_{k,k} \\ & 0 & \mathbf{A}_{2,3} & \ldots & \mathbf{A}_{2,k} \\ & & \ddots & \ddots & \vdots \\ & & & 0 & \mathbf{A}_{k-1,k} \\ & & & & 0 \end{pmatrix} = \begin{pmatrix} 0 & \mathbf{I}_{1,2}\mathbf{A}_{2,2} & \mathbf{I}_{1,3}\mathbf{A}_{3,3} & \ldots & \mathbf{I}_{1,k}\mathbf{A}_{k,k} \\ & 0 & \mathbf{I}_{2,3}\mathbf{A}_{3,3} & \ldots & \mathbf{I}_{2,k}\mathbf{A}_{k,k} \\ & & \ddots & \ddots & \vdots \\ & & & 0 & \mathbf{I}_{k-1,k}\mathbf{A}_{k,k} \\ & & & & 0 \end{pmatrix}.$$

Obviously, the matrix can be expressed using the restrictions $\mathbf{I}_{l,l'}, l < l'$, and the isotropic matrices $\mathbf{A}_{l,l}$. This gives rise to a recursive formulation for $l = k-1, \ldots, 1$

$$\begin{aligned} \mathbf{v}_l &= \sum_{l'=l+2}^{k} \mathbf{A}_{l,l'}u_{l'} + \mathbf{A}_{l,l+1}\mathbf{u}_{l+1} \\ &= \mathbf{I}_{l,l+1}\left(\sum_{l'=(l+1)+1}^{k} \mathbf{A}_{l+1,l'}\mathbf{u}_{l'} + \mathbf{A}_{l+1,l+1}\mathbf{u}_{l+1} \right) \\ &= \mathbf{I}_{l,l+1}\left(\mathbf{v}_{l+1} + \mathbf{A}_{l+1,l+1}\mathbf{u}_{l+1}\right). \end{aligned} \tag{29}$$

Clearly, all $\mathbf{v}_l$ for $1 \leq l \leq k$ can be precalculated in linear time provided that the restrictions $\mathbf{I}_{l,l+1}$ and the application of the matrices $\mathbf{A}_{l,l}$ work in linear time.

3.4.2 TopDown Algorithm

The TopDown algorithm uses a similar recursive relation to calculate the left-hand sides of

$$\mathbf{v}_l = \sum_{l'=1}^{l} \mathbf{A}_{l,l'} \mathbf{u}_{l'} \tag{30}$$

for $1 \leq l \leq k$. The Eqs. (30) can be rewritten as a matrix-vector multiplication with matrix

$$\begin{pmatrix} \mathbf{A}_{1,1} & & & & \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & & & \\ \mathbf{A}_{3,1} & \mathbf{A}_{3,2} & \mathbf{A}_{3,3} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \mathbf{A}_{k,1} & \mathbf{A}_{k,2} & \mathbf{A}_{k,3} & \dots & \mathbf{A}_{k,k} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{1,1} & & & & \\ \mathbf{A}_{2,2}\mathbf{I}_{2,1} & \mathbf{A}_{2,2} & & & \\ \mathbf{A}_{3,3}\mathbf{I}_{3,1} & \mathbf{A}_{3,3}\mathbf{I}_{3,2} & \mathbf{A}_{3,3} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \mathbf{A}_{k,k}\mathbf{I}_{k,1} & \mathbf{A}_{k,k}\mathbf{I}_{k,2} & \mathbf{A}_{k,k}\mathbf{I}_{k,3} & \dots & \mathbf{A}_{k,k} \end{pmatrix},$$

which can solely be expressed using the prolongations $\mathbf{I}_{l,l'}, l > l'$, and the isotropic operator matrices $\mathbf{A}_{l,l}$. We can precalculate vectors $\mathbf{w}_l$ for $l = 2, \dots, k$ using the recursive relationship

$$\begin{aligned} \mathbf{w}_l &:= \sum_{l'=1}^{l} \mathbf{I}_{l,l'} \mathbf{u}_{l'} \\ &= \sum_{l'=1}^{l-1} \mathbf{I}_{l,l-1} \mathbf{I}_{l-1,l'} \mathbf{u}_{l'} + \mathbf{u}_l \\ &= \mathbf{I}_{l,l-1} \mathbf{w}_{l-1} + \mathbf{u}_l \end{aligned}$$

starting with $\mathbf{w}_1 := \mathbf{u}_1$. Now (30) can be expressed by

$$\mathbf{v}_l = \sum_{l'=1}^{l} \mathbf{A}_{l,l} \mathbf{I}_{l,l'} \mathbf{u}_{l'} = \mathbf{A}_{l,l} \sum_{l'=1}^{l} \mathbf{I}_{l,l'} \mathbf{u}_{l'} = \mathbf{A}_{l,l} \mathbf{w}_l. \tag{31}$$

As long as the prolongations $\mathbf{I}_{l,l-1}$ and the application of the matrices $\mathbf{A}_{l,l}$ work in linear time, the TopDown algorithm is of optimal order.

3.4.3 Multi-dimensional Case

The multi-dimensional case can be reduced to the application of the one-dimensional algorithms by a splitting and rearrangement of the sum

$$v_{\mathbf{l}} = \sum_{\mathbf{l}' \in \mathcal{I}} \mathbf{A}^{(1)}_{l_1,l'_1} \otimes \mathbf{A}^{(2)}_{l_2,l'_2} \otimes \cdots \otimes \mathbf{A}^{(d)}_{l_d,l'_d} u_{\mathbf{l}'},$$

which then leads to the equation

$$\mathbf{v}_{\mathbf{l}} = \sum_{\substack{l'_1 \le l_1 \text{ with} \\ (l'_1,l_2,\ldots,l_d)\in\mathcal{I}}} (\mathbf{A}^{(1)}_{l_1,l'_1} \otimes \mathbf{I} \otimes \cdots \otimes \mathbf{I}) \cdot \tag{32}$$

$$\sum_{\substack{(l'_2,\ldots,l'_d) \text{ with} \\ (l'_1,l'_2,\ldots,l'_d)\in\mathcal{I}}} \left(\mathbf{I} \otimes \mathbf{A}^{(2)}_{l_2,l'_2} \otimes \cdots \otimes \mathbf{A}^{(d)}_{l_d,l'_d}\right) \mathbf{u}_{(l'_1,l'_2,\ldots,l'_d)} \tag{33}$$

$$+ \sum_{\substack{(l'_2,\ldots,l'_d) \text{ with} \\ (l_1,l'_2,\ldots,l'_d)\in\mathcal{I}}} \left(\mathbf{I} \otimes \mathbf{A}^{(2)}_{l_2,l'_2} \otimes \cdots \otimes \mathbf{A}^{(d)}_{l_d,l'_d}\right) \cdot \tag{34}$$

$$\sum_{\substack{l'_1 > l_1 \text{ with} \\ (l'_1,l'_2,\ldots,l'_d)\in\mathcal{I}}} (\mathbf{A}^{(1)}_{l_1,l'_1} \otimes \mathbf{I} \otimes \cdots \otimes \mathbf{I}) \mathbf{u}_{(l'_1,l'_2,\ldots,l'_d)}. \tag{35}$$

Now, (32) resembles the application of the one-dimensional TopDown algorithm, and (35) resembles the application of the one-dimensional BottomUp algorithm. The sums (33) and (34) are the result of a recursive application of the multi-dimensional algorithm with the first dimension left unchanged. The condition (21) ensures that the storage of intermediate results requires space and time just in the same order as for $u_{\mathcal{I}}$ and $v_{\mathcal{I}}$. Specifically, we know that

$$(l_1,\ldots,l_d) \in \mathcal{I}, l'_1 \le l_1 \Rightarrow (l'_1,l_2,\ldots,l_d) \in \mathcal{I}$$

in (32) and

$$(l'_1,\ldots,l'_d) \in \mathcal{I}, l'_1 > l_1 \Rightarrow (l_1,l'_2,\ldots,l'_d) \in \mathcal{I}$$

in (34), which means that intermediate results can be represented as generalized sparse grid functions on the same index set $\mathcal{I}$.

The origins of this algorithm trace back to [7, 8] with focus on partial differential operators. Now, with the generic concept presented in this paper, it is no longer necessary to specifically tailor the TopDown/BottomUp algorithms to the operator in use. Moreover, it can be employed with non-local operators like integro-differential operators. Note here that the cost complexity of the algorithm is only linear with respect to the degrees of freedom if the cost complexity of the univariate operator application is linear. But this is possible in the special case of the Kou model, which we will show in the next subsection. Note furthermore that, independently of this work, a similar abstraction of the unidirectional principle for multilevel discretizations has been presented in [30].

3.5 Galerkin Recurrence Formula for the Kou Model

In Sect. 3.4, we have assumed that the matrices $\mathbf{A}^{(1)}_{l_1,l'_1}, \ldots, \mathbf{A}^{(d)}_{l_d,l'_d}$ can be applied in linear time. This is possible for differential operators, as their matrices are inherently sparse for nodal basis functions. However, integral operators do not have this nice property. In [29], a recurrence formula is used to apply the integral operator of the Kou model for the finite difference case in linear time. We now derive a similar result for the Galerkin method.

In the following, let $\{\phi_i\}_{i=1}^{n_l}$ be a set of one-dimensional linear spline basis functions on level l with mesh-width $h = 2^{-l}$ and the relation

$$\phi_i(x) = \phi_{i+1}(x+h), \quad i = 1, \ldots, n_l - 1. \tag{36}$$

In the Galerkin approach for the integro-operator of the Kou model, we need to calculate for all $i = 1, \ldots, n_l$ the expression[7]

$$v_i = \int_{[0,1]} \int_{\mathbb{R}} u(x+z)\kappa_{p,\eta,\mu}(z)\mathrm{d}z\phi_i(x)\mathrm{d}x.$$

Here, $\kappa_{p,\eta,\mu}$ is the Kou density function from (2), and $u : [0,1] \to \mathbb{R}$ denotes a function that can be represented by

$$u(x) = \sum_{j=1}^{n_l} u_j \phi_j(x).$$

This translates to the matrix-vector-product

$$\mathbf{v} = \mathbf{A}\mathbf{u}$$

with $\mathbf{v} = (v_1, \ldots, v_{n_l})$, $\mathbf{u} = (u_1, \ldots, u_{n_l})$ and

$$\begin{aligned}(\mathbf{A})_{i,j} &= \int_{[0,1]} \int_{\mathbb{R}} \phi_j(x+z)\kappa_{p,\eta,\mu}(z)\mathrm{d}z\phi_i(x)\mathrm{d}x \\ &= \int_{[0,1]} \int_{\mathbb{R}} \phi_j(z)\kappa_{p,\eta,\mu}(z-x)\phi_i(x)\mathrm{d}z\mathrm{d}x.\end{aligned} \tag{37}$$

The matrix $\mathbf{A}$ is dense but has Toeplitz structure: Applying (36) to both, ϕ_i and ϕ_j in (37) gives $(\mathbf{A})_{i,j} = (\mathbf{A})_{i+1,j+1}$. This property of the matrix would allow us to execute the matrix-vector multiplication in $\mathcal{O}(n_l \log n_l)$ instead of $\mathcal{O}(n_l^2)$.

[7] Here and in the following, we have omitted time related indices and the dependence on other dimensions.

A different approach that takes also the structure of $\kappa_{p,\eta,\mu}$ into account achieves even a linear runtime complexity. To this end, let us assume that $i, j \in \mathbb{N}, n_l \geq j \geq i+2$. Then we know that the interior of the supports of ϕ_i and ϕ_j is disjoint and that $z > x$ for $\phi_j(z) \neq 0$ and $\phi_i(x) \neq 0$. This allows us to obtain

$$\begin{aligned}(\mathbf{A})_{i,j} &= \int_{[0,1]} \int_{\mathbb{R}} \phi_j(z) k(z-x) \phi_i(x) \mathrm{d}z \, \mathrm{d}x \\ &= \int_{[0,1]} \int_{\mathbb{R}} \phi_j(z) p\mu e^{-\mu(z-x)} \phi_{i+1}(x+h) \mathrm{d}z \, \mathrm{d}x \\ &= \int_{[h,1+h]} \int_{\mathbb{R}} \phi_j(z) p\mu e^{-\mu(z-x+h)} \phi_{i+1}(x) \mathrm{d}z \, \mathrm{d}x \\ &= e^{-h\mu} (\mathbf{A})_{i+1,j}.\end{aligned}$$

A similar argument gives $(\mathbf{A})_{i,j} = e^{-h\eta} (\mathbf{A})_{i-1,j}$ for $1 \leq j \leq i-2$. Now we can introduce the splitting

$$v_i = v_i^{\leftarrow} + v_i^{\circ} + v_i^{\rightarrow}$$

with

$$v_i^{\leftarrow} = \sum_{j=1}^{i-2} (\mathbf{A})_{i,j} u_j, \quad v_i^{\circ} = \sum_{j=i-1}^{i+1} (\mathbf{A})_{i,j} u_j, \quad v_i^{\rightarrow} = \sum_{j=i+2}^{n_l} (\mathbf{A})_{i,j} u_j.$$

With the recursive relationships

$$\begin{aligned}v_i^{\leftarrow} &= \sum_{j=1}^{i-3} (\mathbf{A})_{i,j} u_j + (\mathbf{A})_{i,i-2} u_{i-2} \\ &= e^{-h\eta} \sum_{j=1}^{(i-1)-2} (\mathbf{A})_{i-1,j} u_j + (\mathbf{A})_{i,i-2} u_{i-2} \\ &= e^{-h\eta} v_{i-1}^{\leftarrow} + (\mathbf{A})_{i,i-2} u_{i-2}\end{aligned}$$

for $i = 4, \ldots, n_l$ and

$$\begin{aligned}v_i^{\rightarrow} &= \sum_{j=i+3}^{n_l} (\mathbf{A})_{i,j} u_j + (\mathbf{A})_{i,i+2} u_{i+2} \\ &= e^{-h\mu} \sum_{j=(i+1)+2}^{n_l} (\mathbf{A})_{i+1,j} u_j + (\mathbf{A})_{i,i+2} u_{i+2} \\ &= e^{-h\mu} v_{i+1}^{\rightarrow} + (\mathbf{A})_{i,i+2} u_{i+2}\end{aligned}$$

for $i = 1, \ldots, n_l - 3$, all $v_i^{\leftarrow}$, v_i° and $v_i^{\rightarrow}$ can be precalculated in linear time. Altogether, we can compute the matrix-vector product $\mathbf{v} = \mathbf{A}\mathbf{u}$ in $\mathcal{O}(n_l)$ complexity even though the matrix $\mathbf{A}$ is dense.

So far, we have described the application of the integro-operator of the Kou model to a one-dimensional linear spline discretization on level l only. This approach now easily carries over to the multi-dimensional generating system case by the unidirectional principle with the dimension-recursive form of the algorithm (32)–(35), which exploits a given tensor product structure and requires only one-dimensional non-hierarchical applications of the Kou integral-operator in (29) and (31). This altogether allows the application of the operator matrices in the discretized equation (22) in just linear time.

4 Numerical Experiments

In the following numerical experiments we use the described PIDE solver for the pricing of European basket put options. The focus of our studies is on space discretization, that means we have chosen the domain of computation large enough and the time steps small enough such that the main error now stems from the space discretization only. We are especially interested in the spatial convergence rates of our method. To this end, we look at the L_2-error of the solution at $\tau = T$ and at the relative error of the option price at a predefined point of evaluation. Assuming a relationship

$$e \leq cN^{-\alpha}$$

with e being the error and N the total number of degrees of freedom, we can estimate the convergence rate α by computing

$$\alpha \approx -\frac{\log e_2 - \log e_1}{\log N_2 - \log N_1}$$

for two successive discretization spaces with N_1 and N_2 degrees of freedom and associated errors e_1 and e_2, respectively. As all computational steps can be carried out in linear complexity, this is a relevant measure for the efficiency of our method.

4.1 European Put Option

We now price a European put option on a single asset, i.e. $d = 1$, with the parameters

$$T = 0.2,\ K = 1.0,\ r = 0.00,\ S = 1.0,\ \sigma = 0.2,$$

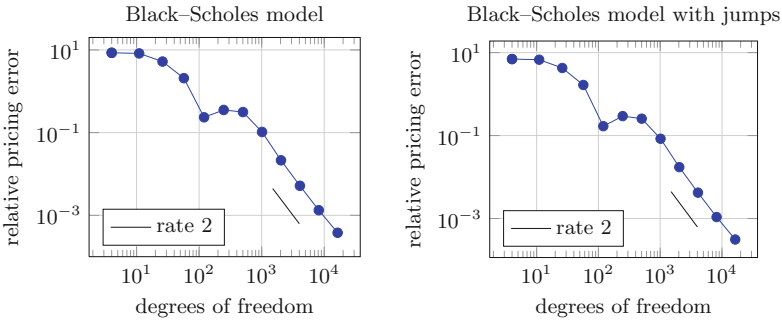


Fig. 3 The relative pricing error at $S_1 = 1.0$ versus the degrees of freedom of the discretization for the Black–Scholes model without jumps (*left*) and with Kou jumps (*right*)

and a jump part

$$\lambda = 0.2,\ \eta = 2.0,\ \mu = 3.0 \text{ and } p = 0.5$$

using the Eqs. (22). The domain is chosen as

$$\mathcal{D} = (-35.781, 35.773)$$

before being scaled down to Ω by (16), and the choice of $\Delta t = 2^{-11}T$ results in 2048 equidistant time steps. The reference price is given by 0.042647805, compare [29]. Without the jump part, the correct Black–Scholes price of the option can be given analytically as 0.035670591. We will use both models to test our method. Figure 3 shows the convergence plots of the relative pricing error with respect to the degrees of freedom[8] with and without the jump part. Here and in Table 1 we see that we essentially get the same rate of 2 as for the simple Black–Scholes PDE. The last measured rate decreases slightly to about 1.8, which can be explained by the time discretization and domain truncation error starting to kick in at that level.

Figure 4 shows that the integro-term just adds a small constant factor to the runtime. Moreover, we see that the runtime is asymptotically proportional to the degrees of freedom as claimed in Sect. 3.5.

4.2 Two-Dimensional Example

We now define two underlyings by

$$r = 0.00 \quad \text{and} \quad \begin{cases} S_1 = 1.0,\ \sigma_1 = 0.2, \\ S_2 = 1.0,\ \sigma_2 = 0.2. \end{cases}$$

[8] In one dimension, full and sparse grids are of course the same.

Table 1 Relative errors and rates α for the one-dimensional option pricing problem with and without jumps. The degrees of freedom (DOFs) refer to the size of the multilevel generating system

		Without jumps		With jumps	
Level	DOFs	Pricing error	Rate α	Pricing error	Rate α
2	4	$8.53 \cdot 10^{0}$	N/A	$6.98 \cdot 10^{0}$	N/A
3	11	$8.23 \cdot 10^{0}$	0.03	$6.75 \cdot 10^{0}$	0.03
4	26	$5.23 \cdot 10^{0}$	0.53	$4.26 \cdot 10^{0}$	0.54
5	57	$2.07 \cdot 10^{0}$	1.18	$1.66 \cdot 10^{0}$	1.20
6	120	$2.35 \cdot 10^{-1}$	2.93	$1.68 \cdot 10^{-1}$	3.07
7	247	$3.54 \cdot 10^{-1}$	−0.57	$2.94 \cdot 10^{-1}$	−0.77
8	502	$3.14 \cdot 10^{-1}$	0.17	$2.55 \cdot 10^{-1}$	0.20
9	1,013	$1.04 \cdot 10^{-1}$	1.58	$8.38 \cdot 10^{-2}$	1.59
10	2,036	$2.14 \cdot 10^{-2}$	2.26	$1.73 \cdot 10^{-2}$	2.26
11	4,083	$5.18 \cdot 10^{-3}$	2.04	$4.21 \cdot 10^{-3}$	2.04
12	8,178	$1.33 \cdot 10^{-3}$	1.96	$1.09 \cdot 10^{-3}$	1.95
13	16,369	$3.79 \cdot 10^{-4}$	1.81	$3.12 \cdot 10^{-4}$	1.80

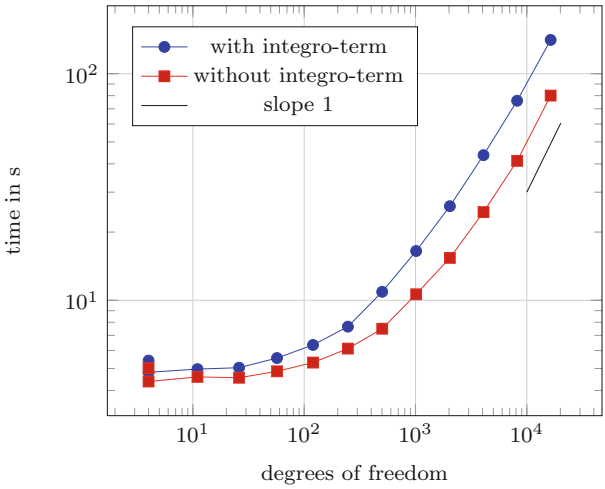


Fig. 4 The run time versus the degrees of freedom. The slope of 1 in this log-log plot relates to the formula $3 \cdot 10^{-3} x$, which indicates that the runtime grows linearly with the degrees of freedom

The first jump term

$$\lambda = 0.3 \quad \begin{cases} \eta_1 = 8.0, \ \mu_1 = 8.0, \ p_1 = 0.0, \\ \eta_2 = 8.0, \ \mu_2 = 8.0, \ p_2 = 0.0, \end{cases} \tag{38}$$

introduces negative jumps in both underlyings, while the second jump term

$$\lambda = 0.2, \ \eta_1 = 10.0, \ \mu = 10.0 \text{ and } p_1 = 0.5 \tag{39}$$

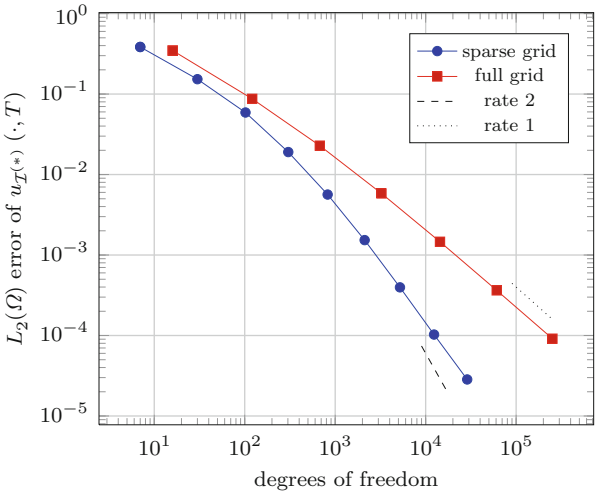


Fig. 5 $L_2(\Omega)$ error of the solution for the geometric call option (40) at $\tau = T$ with a full grid level 10 as reference solution

only affects the first dimension. First, we consider the geometric call option with payoff

$$g(\mathbf{x}) = \max(0, e^{\frac{x_1}{2}+\frac{x_2}{2}} - K) \tag{40}$$

on the domain $\mathcal{D} = (\frac{K}{2}, \frac{3K}{2})^2$ for $K = 1$ as suggested in [24] for the Black-Scholes model. Note here that with this particular choice the non-differentiable kink is not included in the domain of computation, which is favorable for discretizations that rely on additional smoothness constraints like sparse grids. We include both jump terms and calculate the solution approximation $u_{\mathcal{I}^{(*)}}$ based on our PIDE solver approach with anisotropic sparse and full grids, compare (20). In this and in all following experiments, we choose $\Delta t = 2^{-8}T$, which results in 256 equidistant time steps. In Fig. 5 the L_2-error at $T = 0.2$ is measured against a full grid solution on level 10 after the linear transformation (16) to the unit square $\Omega = (0, 1)^2$. As we can see, we achieve roughly a rate of 2 with respect to the degrees of freedom used for the sparse grids, while the full grid rate deteriorates to $\frac{2}{d} = 1$ for $d = 2$.

Now we get back to our usual basket payoff

$$g(\mathbf{x}) = \max(0, K - \sum_{i=1}^{d} e^{x_i}), \tag{41}$$

which is equivalent to (5) after the logarithmic transformation of the coordinates (10). Strike and maturity are chosen as $K = 2$ and $T = 0.2$, respectively. We want to price the option on the underlyings with both jump terms and choose the domain

$$\mathcal{D} = (-0.984, 0.989) \times (-0.942, 0.948),$$

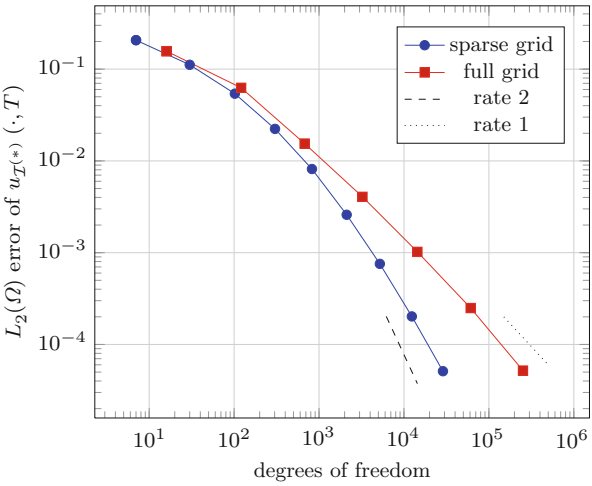


Fig. 6 $L_2(\Omega)$ error of the solution for the basket put option (41) at $\tau = T$ with a full grid level 10 as reference solution

which now includes the non-differentiable kink of the payoff function. In terms of the L_2-error at $T = 0.2$, we achieve asymptotically the rate 2 for the sparse grid approach in contrast to the rate 1 for the full grid, see Fig. 6. This is a noteworthy result, as the payoff function lacks the mixed smoothness regularity, which is typically required by sparse grids. However, it is well-known [28] that the finite element solutions to parabolic problems converge to full order due to the smoothing effect of the solution/propagation operator even when the initial data are nonsmooth.

4.3 Three-Dimensional Option

As an example for a three-dimensional option we set

$$T = 0.2,\ K = 3.0,\ r = 0.00 \quad \text{and} \quad \begin{cases} S_1 = 1.0,\ \sigma_1 = 0.2, \\ S_2 = 1.0,\ \sigma_2 = 0.1, \\ S_3 = 1.0,\ \sigma_1 = 0.05, \end{cases}$$

and choose a jump term

$$\lambda = 0.2 \quad \begin{cases} \eta_1 = 4.0,\ \mu_1 = 5.0,\ p_1 = 0.5, \\ \eta_2 = 10.0,\ \mu_2 = 11.0,\ p_2 = 0.3, \\ \eta_3 = 13.0,\ \mu_3 = 16.0,\ \ p_3 = 0.7. \end{cases}$$

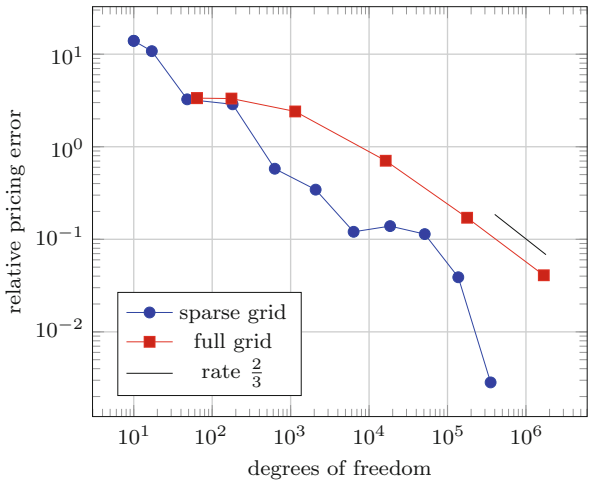


Fig. 7 Relative pricing error at $S_1 = S_2 = S_3 = 1.0$ with respect to the number of unknowns

The option's reference price of 0.04476 has been determined using a Monte Carlo simulation. For our PIDE approach, the parameter choice $\gamma = 30$ results in a domain

$$\mathcal{D} = (-3.302, 3.292) \times (-1.554, 1.555) \times (-0.873, 0.870),$$

which includes the non-differentiable kink. Figure 7 shows the relative pricing error with respect to the degrees of freedom for an anisotropic sparse grid and an anisotropic full grid discretization.

We observe that the convergence rate of the full grid approach is close to $\frac{2}{d}$ like in the former experiments. The sparse grid error convergence is somewhat erratic, but we clearly achieve a higher accuracy with less degrees of freedom.

Note here that the computation of a reference solution for the L_2-error measurement of the solution at maturity would be extremely demanding. We therefore omitted experiments for the L_2-norm and $d = 3$ here. Nevertheless, we expect analogous results as for the case $d = 2$, compare Figs. 5 and 6.

5 Concluding Remarks

We have presented a numerical method for the pricing of multi-dimensional basket options under Kou's jump-diffusion model. It involves a PIDE, i.e. a sum of tensor-product operators, and employs a general sparse grid discretization, which allows us to compute the solutions of moderate-dimensional problems. With the implementation of the unidirectional principle for non-local operators, an optimal preconditioner and a recurrence formula for the Kou model, we achieve linear runtime complexity with respect to the total number of degrees of freedom. The concept can easily be carried over to price more complex option types, e.g. early

exercise options. It can also be generalized to a discretization by space-time sparse grids along the lines of [17, 25].

References

1. L. Andersen and J. Andreasen. Jump-diffusion processes: Volatility smile fitting and numerical methods for option pricing. *Review of Derivatives Research*, 4(3):231–262, 2000.
2. R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.
3. H. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:1–123, 2004.
4. H. Bungartz, A. Heinecke, D. Pflüger, and S. Schraufstetter. Option pricing with a direct adaptive sparse grid approach. *Journal of Computational and Applied Mathematics*, 2011.
5. G. Beylkin and M. Mohlenkamp. Numerical operator calculus in higher dimensions. *Proc. Natl. Acad. Sci. USA*, 99:10246–10251, 2002.
6. F. Black and M. Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.
7. H. Bungartz. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung*. Dissertation, Fakultät für Informatik, Technische Universität München, November 1992.
8. R. Balder and C. Zenger. The solution of multidimensional real Helmholtz equations on sparse grids. *SIAM J. Sci. Comput.*, 17:631–646, May 1996.
9. R. Cont and P. Tankov. *Financial Modelling with Jump Processes*. Chapman & Hall/CRC Financial Mathematics Series, 2004.
10. R. Cont and E. Voltchkova. A finite difference scheme for option pricing in jump diffusion and exponential Lévy models. *SIAM J. Numer. Anal.*, 43:1596–1626, 2005.
11. C. Feuersänger. *Sparse Grid Methods for Higher Dimensional Approximation*. Dissertation, Institut für Numerische Simulation, Universität Bonn, September 2010.
12. T. Gerstner and M. Griebel. Dimension–adaptive tensor–product quadrature. *Computing*, 71(1):65–87, 2003.
13. M. Griebel and H. Harbrecht. On the construction of sparse tensor product spaces. *Mathematics of Computations*, 2012. to appear. Also available as INS Preprint No. 1104, University of Bonn.
14. M. Griebel and S. Knapek. Optimized general sparse grid approximation spaces for operator equations. *Mathematics of Computations*, 78(268):2223–2257, 2009.
15. M. Griebel and P. Oswald. On additive Schwarz preconditioners for sparse grid discretizations. *Numer. Math.*, 66:449–464, 1994.
16. M. Griebel and P. Oswald. Tensor product type subspace splitting and multilevel iterative methods for anisotropic problems. *Adv. Comput. Math.*, 4:171–206, 1995.
17. M. Griebel and D. Oeltz. A sparse grid space-time discretization scheme for parabolic problems. *Computing*, 81(1):1–34, 2007.
18. M. Griebel. *Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen*. Teubner Skripten zur Numerik. Teubner, Stuttgart, 1994.
19. E. Kaasschieter. Preconditioned conjugate gradients for solving singular systems. *Journal of Computational and Applied Mathematics*, 24(1–2):265–275, 1988.
20. S. Kou. A jump-diffusion model for option pricing. *Management Science*, 48(8):1086–1101, 2002.
21. S. Kou. Jump-diffusion models for asset pricing in financial engineering. In John R. Birge and Vadim Linetsky, editors, *Financial Engineering*, volume 15 of *Handbooks in Operations Research and Management Science*, pages 73–116. Elsevier, 2007.
22. R. Merton. Theory of rational option pricing. *Bell Journal of Economics*, 4(1):141–183, 1973.
23. R. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3:125–144, 1976.

24. C. Schwab N. Hilber, S. Kehtari and C. Winter. Wavelet finite element method for option pricing in highdimensional diffusion market models. Research Report 2010-01, Seminar for Applied Mathematics, Swiss Federal Institute of Technology Zurich, 2010.
25. D. Oeltz. *Ein Raum-Zeit Dünngitterverfahren zur Diskretisierung parabolischer Differentialgleichungen*. Dissertation, Institut für Numerische Simulation, Universität Bonn, 2006.
26. C. Reisinger. *Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben*. Dissertation, Ruprecht-Karls-Universität Heidelberg, 2004.
27. N. Reich, C. Schwab, and C. Winter. On Kolmogorov equations for anisotropic multivariate Lévy processes. *Finance and Stochastics*, 14(4):527–567, 2010.
28. V. Thomée. *Galerkin finite element methods for parabolic problems*. Springer series in computational mathematics. Springer, 1997.
29. J. Toivanen. Numerical valuation of European and American options under Kou's jump-diffusion model. *SIAM J. Sci. Comput.*, 30(4):1949–1970, 2008.
30. A. Zeiser. Fast matrix-vector multiplication in the sparse-grid Galerkin method. *J. Sci. Comput.*, 47(3):328–346, 2011.

The Use of Sparse Grid Approximation for the r-Term Tensor Representation

Wolfgang Hackbusch

Abstract The sparse grid approach uses basis functions which are tensor products of univariate basis functions. This gives a connection between sparse grids and the tensor space setting. On the other hand, in the tensor calculus one is interested in suitable representations of tensors (cf. Hackbusch, Tensor spaces and numerical tensor calculus, Monograph in preparation). Then the storage size is in particular described by certain rank parameters. Limited rank requires an approximation of the tensors (functions). In particular, function approximations like polynomial interpolation may lead to such representations. Here, we exploit the sparse grid approximation. We show that the involved rank is not the sparse grid complexity $O(N \log^{d-1} N)$, but $O(N^{(d-1)/d} \log^{d-3} N)$.

1 Tensors, Tensor Representations

Concerning tensors, their representation and approximation we refer to [6]. In Sects. 1–3, we give a brief introduction. The sparse grid approximation in Sect. 4 precedes the main result in Sect. 5.

Let $\mathbf{V} = V_1 \otimes V_2 \otimes \ldots \otimes V_d$ be a tensor space with finite dimensional vector spaces $V_j = \mathbb{R}^{n_j}$ ($n_j \in \mathbb{N}$). Each tensor $\mathbf{v} \in \mathbf{V}$ is uniquely determined by its entries

$$\mathbf{v}[i_1, \ldots, i_d] \in \mathbb{R} \qquad \text{for all } 1 \le i_j \le n_j .$$

W. Hackbusch (✉)
Max-Planck-Institut *Mathematik in den Naturwissenschaften*,
Inselstr. 22, D-04103 Leipzig, Germany
e-mail: wh@mis.mpg.de

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_7,

An *elementary tensor* is the tensor product

$$\mathbf{u} = \bigotimes_{j=1}^{d} u^{(j)}, \quad \text{defined by } \mathbf{u}[i_1, \ldots, i_d] := \prod_{j=1}^{d} u^{(j)}[i_j].$$

By definition of the *algebraic* tensor space, each $\mathbf{v} \in \mathbf{V}$ is a finite linear combination of elementary tensors, i.e.,

$$\mathbf{v} = \sum_{i=1}^{r} \bigotimes_{j=1}^{d} v_i^{(j)}, \quad \text{for some } r \in \mathbb{N}_0,\ v_i^{(j)} \in V_j. \tag{1}$$

The smallest possible integer r in (1) is defined as the *tensor rank* of $\mathbf{v}$, shortly $\operatorname{rank}(\mathbf{v})$. On the other hand, (1) with fixed $r \in \mathbb{N}_0$ is used as a representation scheme for tensors. Set

$$\mathcal{R}_r := \{\mathbf{v} \in \mathbf{V} : \operatorname{rank}(\mathbf{v}) \le r\}.$$

All tensors $\mathbf{v} \in \mathcal{R}_r$ with can be written in the r-term representation $\mathbf{v} = \sum_{i=1}^{r} \bigotimes_{j=1}^{d} v_i^{(j)}$ for some $v_i^{(j)} \in V_j$, requiring a storage of $r \sum_{j=1}^{d} n_j \le rdn$, where $n := \max_j n_j$. Here, r—the number of terms—is called representation rank (the latter is defined by the set $\mathcal{R}_r$, not by $\mathbf{v}$).

Alternatively, we can consider (infinite dimensional) function spaces V_j. Let $I_j \subset \mathbb{R}$ be finite intervals and consider for instance the spaces $V_j = C(I_j)$ of continuous functions of I_j. An elementary tensor $\mathbf{u} = \bigotimes_{j=1}^{d} u^{(j)}$ of univariate functions $u^{(j)} \in V_j$ in the variable x_j is now the multivariate function $\mathbf{u}(x_1, \ldots, x_d) := \prod_{j=1}^{d} u^{(j)}(x_j)$ defined on the Cartesian product $\mathbf{I} := I_1 \times \ldots \times I_d$. The algebraic tensor space $\mathbf{V}_{\mathrm{alg}} = \bigcup_{r \in \mathbb{N}} \mathcal{R}_r$ is dense in $C(\mathbf{I})$, its closure is denoted by $\mathbf{V} = \|\cdot\| \bigotimes_{j=1}^{d} V_j$ and coincides with $C(\mathbf{I})$. For instance, polynomials belong to the algebraic tensor space $\mathbf{V}_{\mathrm{alg}}$ and have finite tensor rank, while for general functions $\mathbf{v}$ not belonging to $\mathbf{V}_{\mathrm{alg}}$ we may define $\operatorname{rank}(\mathbf{v}) = \infty$. All the more, we need approximations as discussed in Sect. 2.

The second classical representation scheme ('Tucker format') uses tensor subspaces. Given a d-tuple $\mathbf{r} = (r_1, \ldots, r_d) \in \mathbb{N}^d$, the set $\mathcal{T}_\mathbf{r}$ is defined by all $\mathbf{v} \in \bigotimes_{j=1}^{d} U_j$, where $U_j \subset V_j$ are arbitrary subspaces with $\dim(U_j) = r_j$. Introducing bases $\{b_i^{(j)} : 1 \le i \le r_j\}$ of U_j, we obtain the equivalent definition

$$\mathcal{T}_\mathbf{r} = \left\{ \mathbf{v} \in \mathbf{V} : \mathbf{v} = \sum_{i_1=1}^{r_1} \cdots \sum_{i_d=1}^{r_d} \mathbf{a}[i_1, \ldots, i_d] \bigotimes_{j=1}^{d} b_{i_j}^{(j)}, \right.$$
$$\left. \text{where } \mathbf{a} \in \bigotimes_{j=1}^{d} \mathbb{R}^{r_j},\ b_{i_j}^{(j)} \in V_j \right\}.$$

The disadvantage of this format, at least for $d \ge 4$, is the fact that it involves a coefficient tensor $\mathbf{a}$ of size $\prod_{j=1}^{d} r_j$, i.e., the storage is still exponential in d.

Remark 1. Less storage is needed if the coefficient tensor $\mathbf{a}$ is sparse. i.e., $\mathbf{a}[\mathbf{i}] \neq 0$ occurs only for $\mathbf{i} \in \mathbf{I}_{\text{sparse}}$, where the cardinality $\#\mathbf{I}_{\text{sparse}}$ is much smaller than $\prod_{j=1}^{d} r_j$. Note that in the latter case, $\mathbf{v} \in \mathcal{T}_{\mathbf{r}}$ can also be viewed as $\mathbf{v} \in \mathcal{R}_r$ with $r := \#\mathbf{I}_{\text{sparse}}$.

2 Tensor Approximations

As seen before, the storage needed for $\mathbf{v} \in \mathcal{R}_r$ is rdn. Therefore, one likes to keep r small. If rank($\mathbf{v}$) is large or even ∞, the obvious remedy is to

$$\text{find } \tilde{\mathbf{v}} \in \mathcal{R}_r \text{ with } \|\mathbf{v} - \tilde{\mathbf{v}}\| = \min\{\|\mathbf{v} - \mathbf{u}\| : \mathbf{u} \in \mathcal{R}_r\} \tag{2}$$

for some norm. However, this problem is in general not solvable, since a minimum need not exist (cf. [4]). Furthermore, a missing minimum implies a numerical instability for any sequence $\tilde{\mathbf{v}}_\varepsilon$ ($\varepsilon > 0$) with $\|\mathbf{v} - \tilde{\mathbf{v}}_\varepsilon\| \to \inf_{\mathbf{u}\in\mathcal{R}_r} \|\mathbf{v} - \mathbf{u}\|$ as $\varepsilon \to 0$ in the way that $\max_i \| \bigotimes_{j=1}^{d} \tilde{v}_{i,\varepsilon}^{(j)} \|$ of $\tilde{\mathbf{v}}_\varepsilon = \sum_{i=1}^{r} \bigotimes_{j=1}^{d} \tilde{v}_{i,\varepsilon}^{(j)}$ does not remain bounded and leads to cancellation effects. This complicates the approximation in $\mathcal{R}_r$ and requires some regularisation (cf. [5]).

If we replace $\mathcal{R}_r$ in (2) by $\mathcal{T}_{\mathbf{r}}$, the minimum exists, but the computational cost is again exponential in d.

3 Function Approximations

Another remedy is to replace $\mathcal{R}_r$ in (2) by a closed subset. Instead of any univariate functions $u^{(j)} \in V_j$, we may consider special ones. Here, we give two examples.

Let $\mathbf{x} = (x_1, \ldots, x_d) \in [1, R]^d$ (possibly with $R = \infty$) and consider $\mathbf{v}(\mathbf{x}) := 1/\|\mathbf{x}\|$ (Euclidean norm). For the construction of a particular $\tilde{\mathbf{v}} \in \mathcal{R}_r$, we approximate the univariate function $\phi(t) = 1/\sqrt{t}$ by exponential sums $\sum_{\nu=1}^{r} \omega_\nu \exp(-\alpha_\nu t)$, where ω_ν, α_ν are optimal with respect to the maximum norm in $[d, dR^2]$ (cf. [1]). Then we substitute $t = \sum_{j=1}^{d} x_j^2$ and obtain $\mathbf{v}(\mathbf{x}) \approx \sum_{\nu=1}^{r} \omega_\nu \prod_{j=1}^{d} \exp(-\alpha_\nu x_j^2)$. The right-hand side belongs to $\mathcal{R}_r$. The error in the maximum norm is the same as for ϕ and has the bound $O(\exp(-cr))$ if $R < \infty$, and $O(\exp(-c\sqrt{r}))$ if $R = \infty$ ($c > 0$ described in [1]).

The usual function approximations are, however, obtained by tensor subspaces yielding $\tilde{\mathbf{v}} \in \mathcal{T}_{\mathbf{r}}$. An example is a polynomial $\tilde{\mathbf{v}}$ of degree $r_j - 1$ with respect to the variable x_j ($1 \le j \le d$). Another example are finite element approximations, provided that the basis functions are tensor products. To be more explicit, consider

univariate (finite element) bases $\{b_i^{(j)} : 1 \le i \le r\}$ for each[1] direction j. Then the multivariate basis is given by

$$\mathbf{b_i} := \bigotimes_{j=1}^{d} b_{i_j}^{(j)} \quad \text{for all } \mathbf{i} \in \mathbf{I} := \{1, \dots, r\}^d. \tag{3}$$

The approximation $\tilde{\mathbf{v}} \in \mathcal{T}_{\mathbf{r}}$ with $\mathbf{r} = (r, \dots, r)$ has the form

$$\tilde{\mathbf{v}} = \sum\nolimits_{\mathbf{i} \in \mathbf{I}} \mathbf{a}[\mathbf{i}]\, \mathbf{b_i} \tag{4}$$

and is of the special form $\tilde{\mathbf{v}} \in \mathbf{U} = \bigotimes_{j=1}^{d} U_j$ with $U_j = \operatorname{span}\{b_i^{(j)} : 1 \le i \le r\}$. Note that $\mathbf{U} \subset \mathcal{T}_{\mathbf{r}}$.

4 Sparse Grid Approximations

In the following we assume a sparse grid approach for the d-dimensional cube $[0, 1]^d$ discretised by piecewise linear elements. This simplification implies identical spaces $V = V_j$. We need a sequence of grids with step sizes $h_\lambda = 2^{-\lambda} h_0$, $\lambda = 1, \dots, \ell$, leading to ansatz spaces

$$V = V_{(\ell)} \supset V_{(\ell-1)} \supset \dots \supset V_{(2)} \supset V_{(1)}, \tag{5}$$

where $V_{(\lambda)}$ corresponds to step size h_λ. While the usual space $\mathbf{V} = \otimes^d V_{(\ell)}$ has dimension $O(2^{-\ell d})$, the sparse grid approach uses the sum of the following tensor spaces:

$$\mathbf{V}_{\mathrm{sg}} = \mathbf{V}_{\mathrm{sg},\ell} = \sum_{\sum_{j=1}^{d} \ell_j = \ell + d - 1} \bigotimes_{j=1}^{d} V_{(\ell_j)}. \tag{6}$$

For the practical implementation, one uses hierarchical bases.[2] $V_{(1)}$ uses the standard hat function basis $\mathfrak{b}_1 := \{(b_i)_{1 \le i \le n_1}$. The basis $\mathfrak{b}_2$ of $V_{(2)}$ is $\mathfrak{b}_1$ enriched by $n_2/2$ hat functions of $V_{(2)}$. The latter additional basis functions are indexed by $n_1 + 1 \le i \le n_1 + n_2/2 = n_2$. In general, the basis $\mathfrak{b}_\lambda$ consists of $\mathfrak{b}_{\lambda-1}$ and additional $n_\lambda/2$ hat functions of $V_{(\lambda)}$. The space $\mathbf{V}_{\mathrm{sg}} \subset \mathbf{V}$ is the span of a much smaller set of basis functions defining implicitly the set $\mathbf{I}_{\mathrm{sparse}}$:

$$\mathbf{V}_{\mathrm{sg}} = \operatorname{span}\{\mathbf{b_i} : \mathbf{i} \in \mathbf{I}_{\mathrm{sparse}}\}. \tag{7}$$

[1] For simplicity, we assume $r_j = r$ for all directions j.

[2] For simplicity, we assume that $\dim(V_{(\lambda)}) = n_\lambda = 2^\lambda n_0$. Depending on the boundary condition, the true number can be $n_\lambda - 1$ or $n_\lambda + 1$.

The well-known statements about the sparse grid approximation are that the interpolation error is almost the same as in the full space case of $\mathbf{V}$, while the complexity is reduced to

$$\sigma_d(N) := \#\mathbf{I}_{\text{sparse}} = O\big(N \log^{d-1}(N)\big) \tag{8}$$

(cf. [2]), where

$$N := n_0^d 2^L, \qquad L := \ell + d - 1.$$

The inequality $\sum_{j=1}^{d} \ell_j \le L = \ell + d - 1$ used for the summation in (6) is equivalent to

$$\prod_{j=1}^{d} n_j \le N := n_0^d 2^L.$$

One concludes that all indices $\mathbf{i}$ involved in (7) satisfy the hyperbolic cross inequality

$$\prod_{j=1}^{d} i_j \le N. \tag{9}$$

We may use (9) as the selection rule, and set $\hat{\mathbf{V}}_{\text{sg}} := \text{span}\{\mathbf{b}_\mathbf{i} : \mathbf{i} \text{ satisfies (9)}\}$. Since $\hat{\mathbf{V}}_{\text{sg}} \supset \mathbf{V}_{\text{sg}}$, this choice satisfies the same error estimates and (8) also holds for

$$\mathbf{I}_{\text{sparse}} := \{\mathbf{i} \in \mathbf{I} : \mathbf{i} \text{ satisfies (9)}\}. \tag{10}$$

5 From Sparse Grid to r-Term Tensor Format

According to Remark 1, the sparse grid approximation gives rise to an r-term representation[3] $\mathbf{v}_{\text{sg}} \in \mathcal{R}_r$ with $r := \#\mathbf{I}_{\text{sparse}}$:

$$\mathbf{v}_{\text{sg}} = \sum\nolimits_{\mathbf{i} \in \mathbf{I}_{\text{sparse}}} \mathbf{a}[\mathbf{i}]\, \mathbf{b}_\mathbf{i}. \tag{11}$$

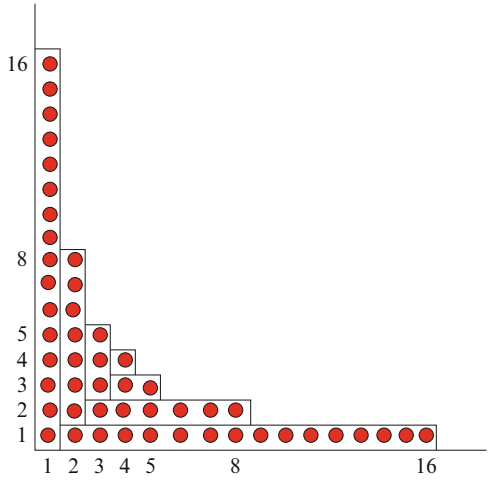
In the following we show that r can be reduced further. For this purpose, we introduce the *fibres* which are generalisations of columns and rows in the matrix case. Given the product $\mathbf{I} = I_1 \times \ldots \times I_d$ of index sets, the j-th fibers $F_{\mathbf{i}'}^{(j)} \subset \mathbf{I}$ are associated to $(d-1)$-tuples $\mathbf{i}' \in \mathbf{I}_{[j]} := I_1 \times \ldots \times I_{j-1} \times I_{j+1} \times \ldots \times I_d$ and consist of

$$F_{\mathbf{i}'}^{(j)} := \{\mathbf{i} \in \mathbf{I} : i_k = i'_k \text{ for } k \ne j \text{ and } i_j \in I_j\} \qquad \text{for all } \mathbf{i}' \in \mathbf{I}_{[j]}.$$

Here, we have the following simple observation.

[3] When we consider (11) as r-term representation, we ignore that $\mathbf{b}_\mathbf{i}$ for different $\mathbf{i}$ may still involve identical b_{i_j}. On the other hand, the factor $\mathbf{a}[\mathbf{i}]$ may be included into the first factor b_{i_1}. Hence, the storage requirement is $\#\mathbf{I}_{\text{sparse}} \cdot d \cdot n_\ell$.

Fig. 1 Partition of $\mathbf{I}_{\rm sparse}$



Remark 2. For any j and $\mathbf{i}' \in \mathbf{I}_{[j]}$ the sum $\sum_{\mathbf{i}\in F^{(j)}_{\mathbf{i}'}} \mathbf{a}[\mathbf{i}]\,\mathbf{b}_{\mathbf{i}}$ can be written as the elementary tensor:

$$b_{i'_1}\otimes\ldots\otimes b_{i'_{j-1}}\otimes\left(\sum\nolimits_{i_j\in I_j}\mathbf{a}[i'_1,\ldots,i'_{j-1},i_j,i'_{j+1},\ldots,i'_d]b_{i_j}\right)\otimes b_{i'_{j+1}}\otimes\ldots\otimes b_{i'_d}.$$

The same statement holds for $F^{(j)}_{\mathbf{i}'}$ replaced by a subset $F\subset F^{(j)}_{\mathbf{i}'}$.

This leads to the following problem.

Problem 1. Given any subset $\mathbf{I}_{\rm sparse}\subset\mathbf{I}$, find a disjoint partition $\mathbf{I}_{\rm sparse}=\bigcup_{p\in P}F_p$ with $F_p\subset F^{(j)}_{\mathbf{i}'}$ for suitable $\mathbf{i}'$, j such that $\#P$ is as small as possible.

By Remark 2, $\mathbf{v}_{\rm sg}$ from (11) can be written as a sum of $\#P$ elementary tensors, i.e., $\mathbf{v}_{\rm sg}\in\mathcal{R}_r$ with $r:=\#P$. If we have a constructive answer to Problem 1, we can also represent $\mathbf{v}_{\rm sg}$ in $\mathcal{R}_r$ constructively.

For the particular definition of $\mathbf{I}_{\rm sparse}$ by (10), we give such a constructive decomposition into fiber pieces. To illustrate the approach, we first consider the case $d=2$ and $i_1i_2\le N=16$ (cf. (9)) in Fig. 1. While in this example $\#\mathbf{I}_{\rm sparse}=50$, only $\#P=7$ partitions are required: $\mathbf{v}_{\rm sg}\in\mathcal{R}_7$. This confirms the later result $\#P\sim\sqrt{\#\mathbf{I}_{\rm sparse}}$ in the case $d=2$.

For general d, the decomposition of $\mathbf{I}_{\rm sparse}$ can be achieved as follows. Let

$$T:=\left\{(t_1,\ldots,t_{d-1})\in\mathbb{N}^{d-1}:\max_{j=1}^{d-1}\{t_j\}\cdot\prod\nolimits_{j=1}^{d-1}t_j\le N\right\}$$

be a set of $(d-1)$-tuples. For each $t := (t_1, \ldots, t_{d-1}) \in T$ and $1 \le k \le d$ define

$$\mathbf{I}_{t,k} := \left\{ (t_1, \ldots, t_{k-1}, i_k, t_k, \ldots, t_{d-1}) \in \mathbf{I}_{\text{sparse}} \quad \text{with } i_k \ge \max_{j=1}^{d-1} \{t_j\} \right\}.$$

$\mathbf{I}_{t,k}$ is part of a fibre in k-th direction. We claim that $\bigcup_{t\in T} \bigcup_{k=1}^{d} \mathbf{I}_{t,k} = \mathbf{I}_{\text{sparse}}$. For a proof take any $\mathbf{i} = (i_1, \ldots, i_d) \in \mathbf{I}_{\text{sparse}}$ and let k and m be indices of the largest and second largest i_j, i.e.,

$$i_k \ge i_m \ge i_j \quad \text{for all } j \in \{1, \ldots, d\} \backslash \{k, m\} \text{ with } k \ne m.$$

Inequality $i_m \le i_k$ and $\mathbf{i} \in \mathbf{I}_{\text{sparse}}$ imply that $i_m \cdot \prod_{j\ne k} i_j \le \prod_{j=1}^{d} i_j \le N$. Therefore the tuple $t := (i_1, \ldots, i_{k-1}, i_{k+1}, \ldots, i_d)$ belongs to T and shows that $\mathbf{i} \in \mathbf{I}_{t,k}$. This proves $\mathbf{I}_{\text{sparse}} \subset \bigcup_{t\in T} \bigcup_{k=1}^{d} \mathbf{I}_{t,k}$, while the direction '$\supset$' is trivial because of $\mathbf{I}_{t,k} \subset \mathbf{I}_{\text{sparse}}$.

We conclude that $\{\mathbf{I}_{t,k} : t \in T, 1 \le k \le d\}$ is a (not necessarily disjoint) decomposition of $\mathbf{I}_{\text{sparse}}$, whose cardinality is denoted by $\tau_d(N)$. Each[4] set $\mathbf{I}_{t,k}$ gives rise to an elementary tensor (cf. Remark 2) and proves $\mathbf{v}_{\text{sg}} \in \mathcal{R}_r$, where[5]

$$r \le \tau_d(N) := d \cdot \#T.$$

It remains to estimate $\#T$. For $t := (t_1, \ldots, t_{d-1}) \in T$ let m be an index with $t_m = \max_{j=1}^{d-1} \{t_j\}$. From $t_m^2 \prod_{j\ne m} t_j \le N$ we conclude that $1 \le t_m \le \sqrt{N}$. In the following, we distinguish the cases $t_m \le N^{1/d}$ and $N^{1/d} < t_m \le N^{1/2}$.

If $t_m \le N^{1/d}$, also $t_j \le N^{1/d}$ holds and all such values satisfy the condition $\max_{j=1}^{d-1}\{t_j\} \cdot \prod_{j=1}^{d-1} t_j \le N$. Hence, the number of $(d-1)$-tuples $t \in T$ with $\max_j \{t_j\} \le N^{1/d}$ is bounded by

$$N^{(d-1)/d}.$$

Now we consider the case $N^{1/d} < t_m \le N^{1/2}$. The remaining components t_j satisfy[6] $\prod_{j\ne m} t_j \le N/t_m^2$. We ignore the condition $t_j \le t_m$ and ask for all $(d-2)$-tuples $(t_j : j \in \{1, \ldots, d-1\} \backslash \{m\})$ with $\prod_{j\ne m} t_j \le N/t_m^2$. Its number is $\sigma_{d-2}(N/t_m^2) = O\big(\frac{N}{t_m^2} \log^{d-3}(\frac{N}{t_m^2})\big)$ (cf. (8)). It remains to bound the sum $\sum_{N^{1/d} < t \le N^{1/2}} \frac{N}{t^2} \log^{d-3}(\frac{N}{t^2})$. Instead, we consider the integral

[4] Since the sets $\mathring{\mathbf{I}}_{t,k}$ may overlap, one must take care that each $\mathbf{v}_{\mathbf{i}}$ is associated to only one $\mathring{\mathbf{I}}_{t,k}$.

[5] $\tau_d(N) < d \cdot \#T$ may occur, since $\mathring{\mathbf{I}}_{t,k} = \mathring{\mathbf{I}}_{t,k'}$ may hold for $k \ne k'$. An example is the decomposition from Fig. 1, where $\tau_2(16) = 7$.

[6] $\prod_{j\ne m}$ abbreviates the product over $\{1, \ldots, d-1\} \backslash \{m\}$.

$$\int_{N^{1/d}}^{N^{1/2}} \frac{N}{x^2} \log^{d-3}\Big(\frac{N}{x^2}\Big)\mathrm{d}x < N\log^{d-3}(N^{(d-2)/d}) \int_{N^{1/d}}^{N^{1/2}} \frac{\mathrm{d}x}{x^2}$$
$$< N\log^{d-3}(N^{(d-2)/d})[N^{-1/d} - N^{-1/2}] < N^{(d-1)/d} \log^{d-3}(N^{(d-2)/d}).$$

Therefore, any tensor $\mathbf{v} = \rho_{\mathrm{sparse}}(\mathbf{I}_{\mathrm{sparse}}, (\mathbf{v}_\mathbf{i})_{\mathbf{i}\in\mathbf{I}}^{\circ}$ can be written as $\mathbf{v} \in \mathcal{R}_r$ with representation rank $r \le O\big(N^{(d-1)/d} \log^{d-3}(N)\big)$. This proves that although $\#\mathbf{I}_{\mathrm{sparse}}$ is not bounded by $O(N)$, the asymptotic rank order is strictly better than $O(N)$.

Proposition 1. *For an index set* $\mathbf{I}_{\mathrm{sparse}} \subset \{\mathbf{i} \in \mathbb{N}^d : \prod_{j=1}^d i_j \le N\}$, *any tensor* $\mathbf{v} = \rho_{\mathrm{sparse}}(\mathbf{I}_{\mathrm{sparse}}, (\mathbf{v}_\mathbf{i})_{\mathbf{i}\in\mathbf{I}}^{\circ})$ *can be explicitly converted into* $\mathbf{v} \in \mathcal{R}_r$ *with a representation rank* $r = \tau_d(N) \le O\big(N^{(d-1)/d} \log^{d-3}(N)\big)$.

The factor $\log^{d-3}$ may be an artifact of the rough estimate. Numerical tests show that the asymptotic behaviour appears rather late. The next table shows $\gamma_d(N) := \log_2(\tau_d(N)/\tau_d(N/2))$ for $N = 2^n$. $\gamma_d(N)$ should converge to $(d-1)/d$. Finally, we compare the quantities $\sigma_d(N)$ (cardinality of the sparse grid) and $\tau_d(N)$ for $d = 3$ and $d = 6$:

n	3	6	9	12	15	18	21	24	27	30	
$\gamma_3(2^n)$	1.0	0.88	0.78	0.73	0.71	0.693	0.684	0.679	0.675	0.672	$\to \frac{2}{3}$
$\gamma_4(2^n)$	1.4	1.15	0.96	0.89	0.85	0.825	0.806	0.794	0.784	0.777	$\to \frac{3}{4}$

N	10	100	1,000	10,000	100,000	1,000,000
$\sigma_3(N)$	53	1,471	29,425	496,623	7,518,850	106,030,594
$\tau_3(N)$	12	102	606	3,265	16,542	81,050
$\sigma_6(N)$	195	14,393	584,325	17, 769,991	439,766,262	–
$\tau_6(N)$	51	2,047	36,018	502,669	5,812,401	59,730,405

6 Conclusion

The basis functions in sparse grid representations are tensor products univariate functions. Therefore, the sparse grid approach may be seen as a particular tensor representation. The number of sparse grid terms is typically $O(N\log^{d-1}N)$, where N is related to the finest grid size. If a sparse grid approximation $\mathbf{v}_{\mathrm{sg}}$ is available, we have shown that an r-term representation of $\mathbf{v}_{\mathrm{sg}}$ is available for $r \sim N^{(d-1)/d} \log^{d-3}(N)$.

We conclude with an hypothetic question. Given any tensor $\mathbf{v} \in \mathbf{V}$, one can compute the higher-order singular value decomposition

$$\mathbf{v} = \sum_{\mathbf{i}\in\mathbf{I}} \mathbf{a}[\mathbf{i}] \bigotimes_{j=1}^{d} b_{i_j}^{(j)},$$

where $b_i^{(j)}$ are the (orthogonal) higher-order singular vectors (cf. [3]).

The connection to the usual singular value decomposition is given by the fact that for any $1 \le j \le d$ there are singular values $\sigma_1^{(j)} \ge \sigma_2^{(j)} \ge \ldots$ such that

$$\sigma_{i_j}^{(j)} = \sqrt{\sum_{i_1,\ldots,i_{j-1},i_{j-1},\ldots,i_d} \mathbf{a}[i_1,\ldots,i_{j-1},i_j,i_{j-1},\ldots,i_d]^2} \qquad \text{for } i_j \in I_j.$$

In this sense, the size of $\mathbf{a}[\mathbf{i}]$ is decreasing, if one of the indices i_j increases. For tensors $\mathbf{v}$ associated to smooth functions, one can expect a fast decrease of the singular values. This does *not* imply that the sparse grid error

$$\sum_{\mathbf{i}\in\mathbf{I} \text{ with } \prod_{j=1}^{d} i_j \le N} \mathbf{a}[\mathbf{i}]^2$$

decays sufficiently fast. However, if this would be true—and numerical calculations for electronic structure problems seem to point into this direction (cf. [7])—the conversion into an approximate r-term representation with $r \lesssim N$ can be performed.

References

1. Braess, D. and Hackbusch, W.: *On the efficient computation of high-dimensional integrals and the approximation by exponential sums*. In: Multiscale, nonlinear and adaptive approximation (R. DeVore, A. Kunoth, eds.), Springer Berlin 2009, pp. 39–74.
2. Bungartz, H.J., Griebel, M.: *Sparse grids*. Acta Numerica **13**, 147–269 (2004)
3. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**, 1253–1278 (2000)
4. De Silva, V., Lim, L.H.: *Tensor rank and the ill-posedness of the best low-rank approximation problem*. SIAM J. Matrix Anal. Appl. **30**, 1084–1127 (2008)
5. M. Espig: *Effiziente Bestapproximation mittels Summen von Elementartensoren in hohen Dimensionen*. Doctoral thesis, University Leipzig, 2008
6. W. Hackbusch: *Tensor spaces and numerical tensor calculus.* Springer Berlin, 2012.
7. Khoromskij, B., Khoromskaia, V.: *Multigrid accelerated tensor approximation of function related multidimensional arrays*. SIAM J. Sci. Comput. **31**, 3002–3026 (2009)

On Multilevel Quadrature for Elliptic Stochastic Partial Differential Equations

Helmut Harbrecht, Michael Peters, and Markus Siebenmorgen

Abstract In this article, we show that the multilevel Monte Carlo method for elliptic stochastic partial differential equations is a sparse grid approximation. By using this interpretation, the method can straightforwardly be generalized to any given quadrature rule for high dimensional integrals like the quasi Monte Carlo method or the polynomial chaos approach. Besides the multilevel quadrature for approximating the solution's expectation, a simple and efficient modification of the approach is proposed to compute the stochastic solution's variance. Numerical results are provided to demonstrate and quantify the approach.

1 Introduction

The present article is dedicated to the numerical solution of elliptic second order boundary value problems with a stochastic diffusion co-efficient. A principal approach to solve such stochastic boundary value problems is the Monte Carlo approach, see e.g. [19] and the references therein. However, it is extremely expensive to generate a large number of suitable samples and to solve a deterministic boundary value problem on each sample. To overcome this obstruction, the multilevel Monte Carlo method (MLMC) has been developed in [1, 10, 11, 15, 16]. From the stochastic point of view, it is a variance reduction technique which considerably decreases the complexity. The idea is to combine the Monte Carlo quadrature of the stochastic variable with a multilevel splitting of the Bochner space which contains the random solution. Then, to compute the solution's statistics, most samples can be performed on coarse spatial discretizations while only a few samples must be performed on fine spatial discretizations. As we will show, this proceeding is a sparse

H. Harbrecht (✉) · M. Peters · M. Siebenmorgen
Mathematisches Institut, Rheinsprung 21, 4051, Basel, Switzerland
e-mail: helmut.harbrecht@unibas.ch; michael.peters@unibas.ch;
markus.siebenmorgen@unibas.ch

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_8,

grid approximation of the expectation and variance. If we replace the Monte Carlo quadrature by another quadrature rule for high dimensional integrals, we obtain for example the multilevel quasi Monte Carlo method (MLQMC) or the multilevel polynomial chaos method (MLPC). A related interpretation of MLMC and MLQMC in financial mathematics as a sparse grid approach is presented in this volume in [8].

2 Sparse Grids

Let

$$V_0^{(i)} \subset V_1^{(i)} \subset \cdots \subset V_j^{(i)} \subset \cdots \subset \mathcal{H}_i, \quad i = 1, 2,$$

denote two sequences of nested, finite dimensional spaces with increasing approximation properties in some abstract spaces $\mathcal{H}_i$. To approximate a given object of the space $\mathcal{H}_1 \times \mathcal{H}_2$ one canonically considers the *full tensor product spaces* $U_j := V_j^{(1)} \otimes V_j^{(2)}$. However, the cost $\dim U_j = \dim V_j^{(1)} \cdot \dim V_j^{(2)}$ are often too expensive. This drawback can be avoided by considering the approximation in so-called *sparse grid* or *sparse tensor product spaces*. To this end, one defines for $j \geq 0$ the complement spaces

$$W_{j+1}^{(i)} = V_{j+1}^{(i)} \ominus V_j^{(i)}, \quad i = 1, 2,$$

and gains the multilevel decompositions

$$V_j^{(i)} = \bigoplus_{i=0}^{j} W_j^{(i)}, \quad W_0^{(i)} := V_0^{(i)}, \quad i = 1, 2. \tag{1}$$

Thus, the sparse grid space is defined by

$$\widehat{U}_j := \sum_{\ell+\ell' \leq j} W_\ell^{(1)} \otimes W_{\ell'}^{(2)} \tag{2}$$

It contains, at most up to a logarithm, only $\max\{\dim V_j^{(1)}, \dim V_j^{(2)}\}$[1] degrees of freedom but offers nearly the same approximation power as U_j provided that the object to be approximated offers some extra smoothness by means of mixed regularity. For further details, see [12].

Sparse grids are used for the approximation of high-dimensional data or functions [14] and for the numerical solution of partial differential equations [22]. Another application issues from the construction of quadrature formulae [7]. For all the details we refer the reader to the survey [2] and the references therein.

[1]This holds under the assumptions that the dimensions of $V_j^{(1)}$ and $V_j^{(2)}$ scale like geometric sequences.

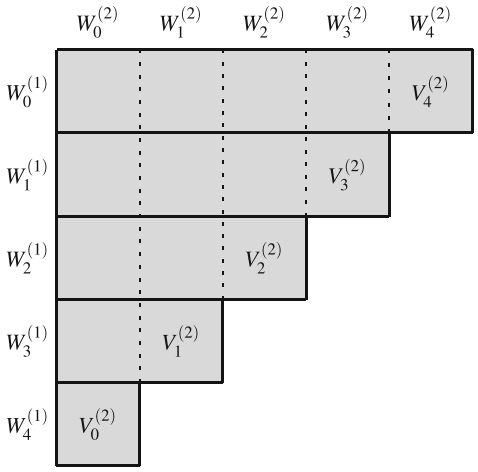


Fig. 1 Different representations of the sparse grid space $\widehat{U}_j$

In the context of stochastic partial differential equations, the sequence $\{V_j^{(1)}\}$ will correspond to a sequence of Bochner spaces which map the high dimensional unit cube $\square$ onto nested finite element spaces. Whereas, the sequence $\{V_j^{(2)}\}$ will correspond to a sequence of quadrature rules which computes the Bochner integral. To arrive at the *multilevel quadrature method* we will make use of the following equivalent representation of the sparse grid space (2). In view of (1), one can rewrite (2) according to

$$\widehat{U}_j = \sum_{\ell=0}^{j} W_\ell^{(1)} \otimes \left(\sum_{\ell'=0}^{j-\ell} W_{\ell'}^{(2)} \right) = \sum_{\ell=0}^{j} W_\ell^{(1)} \otimes V_{j-\ell}^{(2)}$$

(see Fig. 1 for an illustration). This makes the specification of the complement spaces $W_j^{(2)}$ obsolete. Indeed, even the nestedness of the quadrature points can be neglected since it is sufficient that the error of quadrature decreases as the dimension of $V_j^{(2)}$ increases.

The rest of the article is organized as follows. We introduce the elliptic model problem of interest in Sect. 3 and transform it to a parametric diffusion problem in Sect. 4. The next two sections are dedicated to the discretization, namely the quadrature rule for the stochastic variable (Sect. 5) and the finite element method with respect to the physical domain (Sect. 6). The multilevel quadrature for the random solution's expectation is proposed in Sect. 7. The computation of its variance requires some modifications which are carried out in Sect. 8. Finally, in Sect. 9, we provide numerical results to demonstrate and quantify the approach.

3 Problem Setting

In the following, let $D \subset \mathbb{R}^n$ for $n = 2, 3$ be a polygonal or polyhedral domain. Moreover, let (Ω, Σ, P) be a probability space with σ-field $\Sigma \subset 2^\Omega$ and a complete probability measure P, i.e., for all $A \subset B$ and $B \in \Sigma$ with $P[B] = 0$ it follows $A \in \Sigma$ and $P[A] = 0$. We intend to compute the expectation $\mathbb{E}_u := \mathbb{E}(u)$ and the variance $\mathbb{V}_u := \mathbb{V}(u)$ of the random function $u(\omega) \in H_0^1(D)$ which solves the stochastic diffusion problem

$$-\operatorname{div}\big(\alpha(\omega)\nabla u(\omega)\big) = f \text{ in } D. \tag{3}$$

for almost all $\omega \in \Omega$.

To guarantee unique solvability of (3), we assume that $\alpha(\omega)$ is almost surely uniformly elliptic and bounded

$$0 < \alpha_{\min} \leq \alpha(\omega) \leq \alpha_{\max} < \infty \text{ in } D. \tag{4}$$

For sake of simplicity, we further assume that the stochastic diffusion coefficient is given by a finite Karhunen-Loève expansion

$$\alpha(\mathbf{x}, \omega) = \mathbb{E}_\alpha(\mathbf{x}) + \sum_{k=1}^{m} \sqrt{\lambda_k}\varphi_k(\mathbf{x})\psi_k(\omega) \tag{5}$$

with pairwise L^2-orthonormal functions $\varphi_k \in L^\infty(D)$ and stochastically independent random variables $\psi_k(\omega) \in [-1, 1]$. Especially, it is assumed that the random variables admit continuous density functions $\rho_k : [-1, 1] \to \mathbb{R}$ with respect to the Lebesgue measure.

In practice, one generally has to compute the expansion (5) from the given covariance kernel

$$\operatorname{Cov}_\alpha(\mathbf{x}, \mathbf{y}) = \int_\Omega \big\{\alpha(\mathbf{x}, \omega) - \mathbb{E}_\alpha(\mathbf{x})\big\}\big\{\alpha(\mathbf{y}, \omega) - \mathbb{E}_\alpha(\mathbf{y})\big\}\, \mathrm{d}P(\omega).$$

If the expansion contains infinitely many terms, it is appropriately truncated which will induce an additional discretization error. For all the details we refer the reader to [9, 17, 20].

4 Deterministic Reformulation

The assumption that the random variables $\{\psi_k(\omega)\}$ are stochastically independent implies that the respective joint density function and the joint distribution of the random variables are given by

$$\boldsymbol{\rho}(\mathbf{y}) := \prod_{k=1}^{m} \rho_k(y_k) \quad \text{and} \quad \mathrm{d}P_{\boldsymbol{\rho}}(\mathbf{y}) := \boldsymbol{\rho}(\mathbf{y})\mathrm{d}\mathbf{y}.$$

Now we are able to reformulate the stochastic problem (3) by a parametric deterministic problem by replacing the space $L^2_P(\Omega)$ by $L^2_{\boldsymbol{\rho}}(\square)$ where $\square := [-1,1]^m$. Thus, the stochastic space Ω is identified with its image $\square$ with respect to the measurable mapping

$$\boldsymbol{\psi} : \Omega \to \square, \quad \omega \mapsto \boldsymbol{\psi}(\omega) := \big(\psi_1(\omega), \ldots, \psi_m(\omega)\big).$$

Hence, the random variables ψ_k are substituted by coordinates $y_k \in [-1,1]$. With this construction at hand, we define the parameterized and truncated diffusion coefficient $\alpha : D \times \square \to \mathbb{R}$ via

$$\alpha(\mathbf{x},\mathbf{y}) = \mathbb{E}_\alpha(\mathbf{x}) + \sum_{k=1}^{m} \sqrt{\lambda_k}\varphi_k(\mathbf{x})y_k$$

for all $\mathbf{x} \in D$ and $\mathbf{y} = (y_1, y_2, \ldots, y_m) \in \square$. This leads to the following parametric diffusion problem:

$$\begin{aligned} &\text{find } u \in L^2_{\boldsymbol{\rho}}\big(\square; H^1_0(D)\big) \text{ such that} \\ &\quad -\operatorname{div}\big(\alpha(\mathbf{y})\nabla u(\mathbf{y})\big) = f \text{ in } D \text{ for all } \mathbf{y} \in \square. \end{aligned} \tag{6}$$

Here and in the sequel, for a given Banach space X, the Bochner space $L^p_{\boldsymbol{\rho}}(\square; X)$, $1 \le p \le \infty$, consists of all functions $v : \square \to X$ whose norm

$$\|v\|_{L^p_{\boldsymbol{\rho}}(\square;X)} := \begin{cases} \left(\displaystyle\int_{\square} \|v(\mathbf{y})\|_X^p \boldsymbol{\rho}(\mathbf{y})\,\mathrm{d}\mathbf{y}\right)^{1/p}, & p < \infty \\ \operatorname*{ess\,sup}_{\mathbf{y}\in\square} \|v(\mathbf{y})\boldsymbol{\rho}(\mathbf{y})\|_X, & p = \infty \end{cases}$$

is finite. If $p = 2$ and X is a Hilbert space, the Bochner space is isomorphic to the tensor product space $L^2_{\boldsymbol{\rho}}(\square) \otimes X$. Finally, the space $C(\square; X)$ consists of all continuous mappings $v : \square \to X$.

Remark 1. Bochner spaces are the canonical function spaces to define stochastic fields $v : \square \to X$. The expectation is defined via the Bochner integral

$$\mathbb{E}_v(\mathbf{x}) = \int_{\square} v(\mathbf{x},\mathbf{y})\boldsymbol{\rho}(\mathbf{y})\,\mathrm{d}\mathbf{y}$$

and the variance via

$$\mathbb{V}_v(\mathbf{x}) = \int_{\square} v^2(\mathbf{x},\mathbf{y})\boldsymbol{\rho}(\mathbf{y})\,\mathrm{d}\mathbf{y} - \mathbb{E}_v^2(\mathbf{x}).$$

For further details we refer the reader to [3, 5].

By the Lax-Milgram theorem, unique solvability of the parametric diffusion problem (6) in $L^2_{\boldsymbol{\rho}}\big(\square; H^1_0(D)\big)$ follows immediately from the condition (4) on the stochastic diffusion coefficient. In [21], it has further been proven that the solution u of (6) is analytical as mapping $u : \square \to H^1_0(D)$. At least in case of uniformly distributed random variables $\{\psi_k\}$, it is even analytical as mapping $u : \square \to H^1_0(D) \cap H^2(D)$, provided that the functions $\{\varphi_k\}$ in the Karhunen-Loève expansion (5) are in $W^{1,\infty}(D)$, see [4].

5 Quadrature in the Stochastic Variable

Having the solution $u \in L^2_{\boldsymbol{\rho}}\big(\square; H^1_0(D)\big)$ of (6) at hand, then its expectation and variance are given by the high dimensional integrals

$$\mathbb{E}_u(\mathbf{x}) = \int_{\square} u(\mathbf{x},\mathbf{y})\boldsymbol{\rho}(\mathbf{y})\,\mathrm{d}\mathbf{y} \in H^1_0(D) \tag{7}$$

and

$$\mathbb{V}_u(\mathbf{x}) = \mathbb{E}_{u^2}(\mathbf{x}) - \mathbb{E}_u^2(\mathbf{x}) = \int_{\square} u^2(\mathbf{x},\mathbf{y})\boldsymbol{\rho}(\mathbf{y})\,\mathrm{d}\mathbf{y} - \mathbb{E}_u^2(\mathbf{x}) \in W^{1,1}_0(D). \tag{8}$$

To compute these integrals, we shall provide a sequence of quadrature formulae $\{Q_\ell\}$ for the Bochner integral

$$I : L^1_{\boldsymbol{\rho}}(\square; X) \to X, \quad Iv = \int_{\square} v(\cdot,\mathbf{y})\boldsymbol{\rho}(\mathbf{y})\,\mathrm{d}\mathbf{y}$$

where $X \subset L^2(D)$ denotes a Banach space. The quadrature formula

$$Q_\ell : L^1_{\boldsymbol{\rho}}(\square; X) \to X, \quad Q_\ell v = \sum_{i=1}^{N_\ell} \omega_{\ell,i}\, v(\cdot,\boldsymbol{\xi}_{\ell,i})\boldsymbol{\rho}(\boldsymbol{\xi}_{\ell,i}) \tag{9}$$

is supposed to provide the error bound

$$\|(I - Q_\ell)v\|_X \lesssim \varepsilon_\ell \|v\|_{\mathcal{H}(\square;X)} \tag{10}$$

uniformly in $\ell \in \mathbb{N}$, where $\{\varepsilon_\ell\}$ is a null sequence and $\mathcal{H}(\square; X) \subset L^2_{\boldsymbol{\rho}}(\square, X)$ a suitable Bochner space. For our purpose, we assume that the number of points N_ℓ of the quadrature formula Q_ℓ are chosen such that

$$\varepsilon_\ell = 2^{-\ell}. \tag{11}$$

The following particular examples of quadrature rules (9) are considered in our numerical experiments:

- In the mean, the Monte Carlo method satisfies (10) with $\varepsilon_\ell = N_\ell^{-1/2}$ and $\mathcal{H}(\square; X) = L^2_{\boldsymbol{\rho}}(\square; X)$.
- The standard quasi Monte Carlo method leads typically to $\varepsilon_\ell = N_\ell^{-1}(\log N_\ell)^m$ and the Bochner space $\mathcal{H}(\square; X) = W^{1,1}_{\mathrm{mix}}(\square; X)$ of all functions $v : \square \to X$ with finite norm

$$\|v\|_{W^{1,1}_{\mathrm{mix}}(\square;X)} := \sum_{\|\mathbf{q}\|_\infty \le 1} \int_\square \left\| \frac{\partial^{\|\mathbf{q}\|_1}}{\partial y_1^{q_1} \partial y_2^{q_2} \cdots \partial y_m^{q_m}} v(\mathbf{y}) \right\|_X \mathrm{d}\mathbf{y} < \infty, \qquad (12)$$

  see e.g. [18]. Note that this estimate requires that the densities satisfy $\rho_k \in W^{1,\infty}([-1, 1])$.
- If $v : \square \to X$ is analytical with respect to the variable $\mathbf{y}$, then one can apply a tensor product Gaussian quadrature rule, yielding an exponential convergence rate $\varepsilon_\ell = \exp(-bN_\ell^{1/m})$ and $\mathcal{H}(\square; X) = L^\infty(\square; X)$. In fact, the polynomial chaos approach introduced in [6] can be interpreted as a dimension weighted tensor product Gaussian quadrature rule (with weights depending on the numbers $\{\sqrt{\lambda_k}\|\varphi_k\|_{L^\infty(D)}\}$).

6 Finite Element Approximation in the Spatial Variable

In order to apply the quadrature formula (9), we shall calculate the solution $u(\mathbf{y}) \in H^1_0(D)$ of the diffusion problem (6) in certain points $\mathbf{y} \in \square$. To this end, consider a coarse grid triangulation/tetrahedralization $\mathcal{T}_0 = \{\tau_{0,k}\}$ of the domain D. Then, for $\ell \ge 1$, a uniform and shape regular triangulation/tetrahedralization $\mathcal{T}_\ell = \{\tau_{\ell,k}\}$ is recursively obtained by uniformly refining each triangle/tetrahedron $\tau_{\ell-1,k}$ into 2^n triangles/tetrahedrons with diameter $h_\ell \sim 2^{-\ell}$.

Define for $p \ge 1$ the finite element spaces

$$\mathcal{S}^p_\ell(D) := \{v \in C(D) : v|_{\partial D} = 0 \text{ and } v|_\tau \in \mathcal{P}_p \text{ for all } \tau \in \mathcal{T}_\ell\} \subset H^1_0(D),$$

where $\mathcal{P}_p$ denotes the space of all polynomials of total degree $p \ge 0$. Then, the solution $u(\mathbf{y}) \in \mathcal{S}^p_\ell(D)$ of a finite element method in the space $\mathcal{S}^p_\ell(D)$ admits the following approximation properties. [2]

[2] Error estimates in respectively $L^2(D)$ and $L^1(D)$ are derived by straightforward modifications, yielding the convergence rate $4^{-\ell}$. Then, the error analysis of the multilevel quadrature can be performed with respect to these norms, provided that the precision of the quadrature (10) is also chosen as $\varepsilon_\ell = 4^{-\ell}$.

Lemma 1. *Let the domain D be convex and $f \in L^2(D)$. Then, the finite element solution $u_\ell(\mathbf{y}) \in \mathcal{S}_\ell^p(D)$ of the diffusion problem (6) satisfies the error estimates*

$$\|u(\mathbf{y}) - u_\ell(\mathbf{y})\|_{H^1(D)} \lesssim 2^{-\ell} \|u(\mathbf{y})\|_{H^2(D)}, \tag{13}$$

and

$$\|u^2(\mathbf{y}) - u_\ell^2(\mathbf{y})\|_{W^{1,1}(D)} \lesssim 2^{-\ell} \|u(\mathbf{y})\|_{H^2(D)}^2. \tag{14}$$

Here, the constants hidden in (13) and (14) depend on $\alpha_{\min}$ and $\alpha_{\max}$, but not on $\mathbf{y} \in \square$.

Proof. The parametric diffusion problem (6) is H^2-regular since D is convex and $f \in L^2(D)$. Hence, the first error estimate immediately follows from the standard finite element theory. We further find

$$\begin{aligned}
\|u^2(\mathbf{y}) - u_\ell^2(\mathbf{y})\|_{W^{1,1}(D)} &= \big\| \big(u(\mathbf{y}) - u_\ell(\mathbf{y})\big)\big(u(\mathbf{y}) + u_\ell(\mathbf{y})\big) \big\|_{W^{1,1}(D)} \\
&\lesssim \|u(\mathbf{y}) - u_\ell(\mathbf{y})\|_{H^1(D)} \|u(\mathbf{y}) + u_\ell(\mathbf{y})\|_{L^2(D)} \\
&\quad + \|u(\mathbf{y}) - u_\ell(\mathbf{y})\|_{L^2(D)} \|u(\mathbf{y}) + u_\ell(\mathbf{y})\|_{H^1(D)}.
\end{aligned}$$

By using

$$\|u_\ell(\mathbf{y})\|_{H^1(D)} \leq \|u(\mathbf{y})\|_{H^1(D)} + \|u(\mathbf{y}) - u_\ell(\mathbf{y})\|_{H^1(D)} \lesssim (1 + 2^{-\ell}) \|u(\mathbf{y})\|_{H^2(D)}$$

and the corresponding estimate in $L^2(D)$, we arrive at the desired estimate (14). □

Remark 2. If the underlying diffusion problem is more than H^2-regular, for example in the situation of isoparametric finite elements, one obtains higher order error estimates if $p \geq 2$. The subsequent analysis can be modified in this case straightforwardly.

7 Multilevel Quadrature Method for the Expectation

We now have to combine the quadrature method with the multilevel finite element discretization. To this end, we define the ansatz spaces

$$V_j^{(1)} := \big\{ G_j(\mathbf{y}) v(\mathbf{x}, \mathbf{y}) : v \in C\big(\square; H_0^1(D)\big) \text{ and } \mathbf{y} \in \square \big\} \subset L^2_{\boldsymbol{\rho}}\big(\square; \mathcal{S}_j^p(D)\big). \tag{15}$$

Herein, $G_j(\mathbf{y})$ denotes the Galerkin projection

$$G_j(\mathbf{y}) : H_0^1(D) \to \mathcal{S}_j^p(D), \quad v \mapsto v_j,$$

defined via the Galerkin orthogonality

$$\int_D \alpha(\mathbf{x},\mathbf{y})\nabla\big(v(\mathbf{x})-v_j(\mathbf{x})\big)\nabla w_j(\mathbf{x})\,\mathrm{d}\mathbf{x}=0 \text{ for all } w_j\in\mathcal{S}_j^p(D).$$

Note that this bilinear form issues from the weak formulation of (6) in $H_0^1(D)$.[3]

To compute the expectation (7) we shall apply the quadrature rule Q_j to the finite element solution in $\mathcal{S}_j^p(D)$ which yields

$$\mathbb{E}_u(\mathbf{x})\approx Q_j\big(G_j(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)=\sum_{i=0}^{N_j}\omega_{j,i}G_j(\boldsymbol{\xi}_{j,i})u(\mathbf{x},\boldsymbol{\xi}_{j,i})\rho(\boldsymbol{\xi}_{j,i}). \tag{16}$$

This can be interpreted as the *full* tensor approximation of the function $\mathbb{E}_u$ in the product space $V_j^{(1)}\otimes V_j^{(2)}$ where the quadrature rule Q_j serves as "space" $V_j^{(2)}$.

In contrast to this, setting $G_{-1}(\mathbf{y}):=0$ for all $\mathbf{y}\in\square$, the *sparse* tensor product $\sum_{\ell=0}^j W_\ell^{(1)}\otimes V_{j-\ell}^{(2)}$ is performed with the help of the complement spaces

$$W_\ell^{(1)}:=\big\{\big(G_\ell(\mathbf{y})-G_{\ell-1}(\mathbf{y})\big)v(\mathbf{x},\mathbf{y}) : v\in C\big(\square;H_0^1(D)\big) \text{ and } \mathbf{y}\in\square\big\}\subset V_\ell^{(1)}.$$

Namely, we consider the sparse tensor approximation

$$\begin{aligned}\mathbb{E}_u(\mathbf{x})&\approx\sum_{\ell=0}^{j}Q_{j-\ell}\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y})-G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)\\&=\sum_{\ell=0}^{j}\sum_{i=0}^{N_{j-\ell}}\omega_{j-\ell,i}\big(G_\ell(\boldsymbol{\xi}_{j-\ell,i})u(\mathbf{x},\boldsymbol{\xi}_{j-\ell,i})\\&\quad-G_{\ell-1}(\boldsymbol{\xi}_{j-\ell,i})u(\mathbf{x},\boldsymbol{\xi}_{j-\ell,i})\big)\rho(\boldsymbol{\xi}_{j-\ell,i}).\end{aligned} \tag{17}$$

Loosely speaking, the function $u\in L^2_{\boldsymbol{\rho}}\big(\square;H_0^1(D)\big)$ is divided into j slices in accordance with the modulus of its entity. Then, for every slice, the precision of quadrature is properly chosen. We refer to Fig. 2 for a graphical illustration.

Theorem 1. *Let $u\in\mathcal{H}\big(\square;H_0^1(D)\cap H^2(D)\big)$ and let $\{Q_\ell\}$ be a sequence of quadrature rules which satisfy (10) and (11). Then, it holds*

[3] There holds the identity $V_j^{(1)}=P_j\big(L^2_{\boldsymbol{\rho}}\big(\square;H_0^1(D)\big)\big)$ where $P_j:L^2_{\boldsymbol{\rho}}\big(\square;H_0^1(D)\big)\to L^2_{\boldsymbol{\rho}}\big(\square;\mathcal{S}_j^p(D)\big)$ is the Galerkin projection with respect to the bilinear form $A: L^2_{\boldsymbol{\rho}}\big(\square;H_0^1(D)\big)\times L^2_{\boldsymbol{\rho}}\big(\square;H_0^1(D)\big)\to\mathbb{R}$ given by

$$A(v,w):=\int_\square\int_D\alpha(\mathbf{x},\mathbf{y})\nabla_{\mathbf{x}}v(\mathbf{x},\mathbf{y})\nabla_{\mathbf{x}}w(\mathbf{x},\mathbf{y})\boldsymbol{\rho}(\mathbf{y})\,\mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{y}.$$

This bilinear form stems from the weak formulation of the parametric diffusion problem (6) in the Bochner space $L^2_{\boldsymbol{\rho}}\big(\square;H_0^1(D)\big)$. The equivalence of this *weak definition* to the pointwise definition (15) follows immediately from the analyticity of the solutions to (6) in the parameter $\mathbf{y}$.

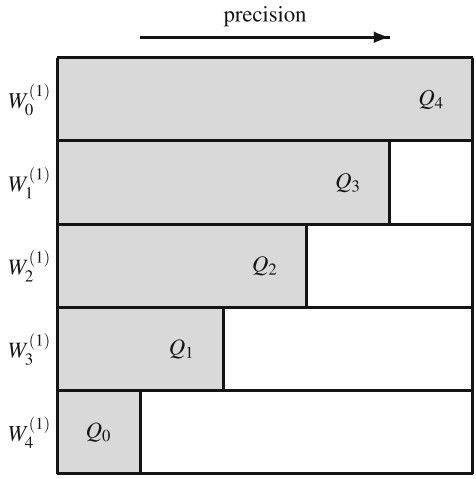


Fig. 2 Visualization of the multilevel quadrature

$$\left\| \mathbb{E}_u(\boldsymbol{x}) - \sum_{\ell=0}^{j} Q_{j-\ell}\big(G_\ell(\boldsymbol{y})u(\boldsymbol{x},\boldsymbol{y}) - G_{\ell-1}(\boldsymbol{y})u(\boldsymbol{x},\boldsymbol{y})\big) \right\|_{H^1(D)} \lesssim 2^{-j}\, j\, \|u\|_{\mathcal{H}(\square;H_2(D))}.$$

Proof. Since $u \in \mathcal{H}\big(\square; H_0^1(D) \cap H^2(D)\big)$, the estimate (13) implies the decay

$$\|G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y}) - G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\|_{\mathcal{H}(\square;H_0^1(D))} \lesssim 2^{-\ell}\|u\|_{\mathcal{H}(\square;H^2(D))}.$$

This, together with (10), (11), and (13), yields

$$\begin{aligned}
&\left\| \mathbb{E}_u(\mathbf{x}) - \sum_{\ell=0}^{j} Q_{j-\ell}\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y}) - G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big) \right\|_{H^1(D)} \\
&\quad\lesssim \left\| \mathbb{E}_u(\mathbf{x}) - I\big(G_j(\mathbf{y})u(\mathbf{x},\mathbf{y})\big) \right\|_{H^1(D)} \\
&\qquad + \sum_{\ell=0}^{j} \left\| (I - Q_{j-\ell})\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y}) - G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big) \right\|_{H^1(D)} \\
&\quad\lesssim 2^{-j}\|u\|_{L^2_\rho(\square;H^2(D))} + \sum_{\ell=0}^{j} 2^{\ell-j}2^{-\ell}\|u\|_{\mathcal{H}(\square;H^2(D))} \\
&\quad\lesssim 2^{-j}\, j\, \|u\|_{\mathcal{H}(\square;H^2(D))}.
\end{aligned}$$

□

Remark 3. The convergence rate of the full tensor approximation (16) is 2^{-j}. Therefore, the convergence rate of the sparse tensor approximation (17) is only

the logarithmic factor j smaller. This factor can be removed, if the precision of quadrature ε_ℓ is chosen as $\ell^{-(1+\delta)}2^{-\ell}$ for some $\delta > 0$ since

$$\sum_{\ell=0}^{j} 2^{\ell-j}2^{-\ell}(j-\ell)^{-(1+\delta)} = 2^{-j}\sum_{\ell=0}^{j}\ell^{-(1+\delta)} \lesssim 2^{-j}.$$

This choice was proposed in [1].

8 Multilevel Quadrature Method for the Second Moment

The determination of the random solution's variance (8) requires the computation of its second moment $\mathbb{E}_{u^2} \in W_0^{1,1}(D)$ which can be performed similarly to the expectation. The full tensor approximation

$$\mathbb{E}_{u^2}(\mathbf{x}) \approx Q_j\big(G_j(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2 = \sum_{i=0}^{N_j}\omega_i\big(G_j(\boldsymbol{\xi}_{j,i})u(\mathbf{x},\boldsymbol{\xi}_{j,i})\big)^2\boldsymbol{\rho}(\boldsymbol{\xi}_{j,i})$$

corresponds to the approximation in the product "space" $\widetilde{V}_j^{(1)} \otimes V_j^{(2)}$ with

$$\widetilde{V}_j^{(1)} := \big\{\big(G_j(\mathbf{y})v(\mathbf{x},\mathbf{y})\big)^2 : v \in C\big(\square; H_0^1(D)\big) \text{ and } \mathbf{y} \in \square\big\} \subset L^2_{\boldsymbol{\rho}}\big(\square; \mathcal{S}_j^{2p}(D)\big).$$

To define the related sparse tensor approximation, we set

$$\widetilde{W}_\ell^{(1)} := \big\{\big(G_\ell(\mathbf{y})v(\mathbf{x},\mathbf{y})\big)^2 - \big(G_{\ell-1}(\mathbf{y})v(\mathbf{x},\mathbf{y})\big)^2 : v \in C\big(\square; H_0^1(D)\big) \text{ and } \mathbf{y} \in \square\big\} \subset \widetilde{V}_j^{(1)}.$$

Then, the approximation of $\mathbb{E}_{u^2}$ in the sparse product "space" $\sum_{\ell=0}^{j}\widetilde{W}_{j-\ell}^{(1)} \otimes V_j^{(2)}$ is given as

$$\begin{aligned}\mathbb{E}_{u^2}(\mathbf{x}) &\approx \sum_{\ell=0}^{j} Q_{j-\ell}\Big(\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2 - \big(G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2\Big)\\ &= \sum_{\ell=0}^{j}\sum_{i=1}^{N_{j-\ell}}\omega_{j-\ell,i}\Big(\big(G_\ell(\boldsymbol{\xi}_{j-\ell,i})u(\mathbf{x},\boldsymbol{\xi}_{j-\ell,i})\big)^2\\ &\quad - \big(G_{\ell-1}(\boldsymbol{\xi}_{j-\ell,i})u(\mathbf{x},\boldsymbol{\xi}_{j-\ell,i})\big)^2\Big)\boldsymbol{\rho}(\boldsymbol{\xi}_{j-\ell,i}).\end{aligned}$$

To estimate the discretization error of this sparse tensor approximation, we will need to further estimate the expression

$$\begin{aligned}&\left\|(I-Q_{j-\ell})\Big(\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2-\big(G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2\Big)\right\|_{W^{1,1}(D))}\\&\quad\lesssim 2^{j-\ell}\left\|\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2-\big(G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2\right\|_{\mathcal{H}(\square;W^{1,1}(D))}.\end{aligned}\tag{18}$$

If $u\in\mathcal{H}\big(\square;H_0^1(D)\cap H^2(D)\big)$, then this is clearly possible by using (14) but requires some further specification of the Bochner space $\mathcal{H}(\square)$. For example, if $\mathcal{H}(\square)=L^2_\rho(\square)$, we arrive at

$$\left\|\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2-\big(G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2\right\|_{L^2_\rho(\square;W^{1,1}(D))}\lesssim 2^{-\ell}\|u\|^2_{L^4_\rho(\square;H^2(D))}.$$

Whereas, if $\mathcal{H}(\square)=W^{1,1}_{\mathrm{mix}}(\square)$, one gets, in view of (12), the estimate

$$\left\|\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2-\big(G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2\right\|_{W^{1,1}_{\mathrm{mix}}(\square;W^{1,1}(D))}\lesssim 2^{-\ell}\|u\|^2_{H^1_{\mathrm{mix}}(\square;H^2(D))},$$

where the norm in $H^1_{\mathrm{mix}}(\square;X)$ is defined in complete analogy to (12). Finally, in case of $\mathcal{H}(\square)=L^\infty_\rho(\square)$ we obtain

$$\left\|\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2-\big(G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2\right\|_{L^\infty_\rho(\square;W^{1,1}(D))}\lesssim 2^{-\ell}\|u\|^2_{L^\infty_\rho(\square;H^2(D))}.$$

These examples cover the three quadrature formulae specified in Sect. 5. Therefore, it is reasonable to make the assumption that a second Bochner space $\mathcal{I}\big(\square;H^2(D)\big)\subset\mathcal{H}\big(\square;H^2(D)\big)$ exists with which we can bound the right hand side in (18).

Theorem 2. *Let $u\in\mathcal{H}\big(\square;H_0^1(D)\cap H^2(D)\big)$ and let $\{Q_\ell\}$ be a sequence of quadrature rules which satisfy (10) and (11). Moreover, assume that*

$$\left\|\big(G_\ell(\mathbf{y})u(\boldsymbol{x},\mathbf{y})\big)^2-\big(G_{\ell-1}(\mathbf{y})u(\boldsymbol{x},\mathbf{y})\big)^2\right\|_{\mathcal{H}(\square;W^{1,1}(D))}\lesssim 2^{-\ell}\|u\|^2_{\mathcal{I}(\square;H^2(D))}.\tag{19}$$

Then, there holds the following error estimate:

$$\begin{aligned}&\left\|\mathbb{E}_{u^2}(\boldsymbol{x})-\sum_{\ell=0}^{j}Q_{j-\ell}\Big(\big(G_\ell(\mathbf{y})u(\boldsymbol{x},\mathbf{y})\big)^2-\big(G_{\ell-1}(\mathbf{y})u(\boldsymbol{x},\mathbf{y})\big)^2\Big)\right\|_{W^{1,1}(D)}\\&\quad\lesssim 2^{-j}\,j\,\big(\|u\|^2_{L^4_\rho(\square;H^2(D))}+\|u\|^2_{\mathcal{I}(\square;H^2(D))}\big).\end{aligned}$$

Proof. Analogously to the proof of the sparse tensor approximation to the expectation, we find

$$\begin{aligned}
&\left\| \mathbb{E}_{u^2}(\mathbf{x}) - \sum_{\ell=0}^{j} Q_{j-\ell}\Big(\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2 - \big(G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2 \Big) \right\|_{W^{1,1}(D)} \\
&\quad \lesssim \left\| \mathbb{E}_{u^2} - I\big(G_j(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2 \right\|_{W^{1,1}(D)} \\
&\qquad + \sum_{\ell=0}^{j} \left\| (I - Q_{j-\ell})\Big(\big(G_\ell(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2 - \big(G_{\ell-1}(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2 \Big) \right\|_{W^{1,1}(D)}.
\end{aligned}$$

Herein, the first term is estimated by

$$\left\| \mathbb{E}_{u^2} - I\big(G_j(\mathbf{y})u(\mathbf{x},\mathbf{y})\big)^2 \right\|_{W^{1,1}(D)} \lesssim 2^{-j} \|u\|^2_{L^4_\rho(\square;H^2(D))}.$$

By bounding the second term by (18) and (19), we derive the desired error estimate. □

9 Numerical Results

In the implementation of our numerical examples we consider the L-shaped domain $D = (-1,1)^2 \setminus [0,1)^2 \subset \mathbb{R}^2$ and choose piecewise linear finite elements on triangles for the discretization (i.e., the spaces $\{\mathcal{S}^1_j(D)\}$). Then, the variance will be a piecewise quadratic finite element function which lives in the spaces $\{\mathcal{S}^2_j(D)\}$. Therefore, the multilevel mesh transfer has to be performed by quadratic prolongations.

It is well known that an approximation of the diffusion coefficient by elementwise constant functions sustains the over-all approximation order achieved by the piecewise linear finite elements on the finest mesh. On the coarser meshes, the diffusion coefficient is successively replaced by its elementwise average with respect to the current triangulation. This does not affect the stiffness matrices but simplifies the computations considerably.

In the numerical experiments, we compare the following particular multilevel quadrature methods:

- The *multilevel Monte Carlo method* where the number of sample points is chosen as $N_\ell = 4^\ell N_0$ with $N_0 = 100$. This choice ensures that the quadrature error estimate (11) holds in the mean.
- The *multilevel quasi Monte Carlo method* using the Halton sequence [18]. It satisfies the error estimate $\varepsilon_\ell \sim N_\ell^{-1} \log^m(N_\ell)$. Since the term $\log^m(N_\ell)$ is unbounded as $m \to \infty$ (which happens in the second numerical example), the number of sample points is chosen as for the multilevel Monte Carlo method.
- *The multilevel polynomial chaos method* which is based on the polynomial chaos method presented in [6]. The polynomial degree and thus the number of quadrature points is coupled to the exact decay of the eigenvalues in the Karhunen-Loève expansion.

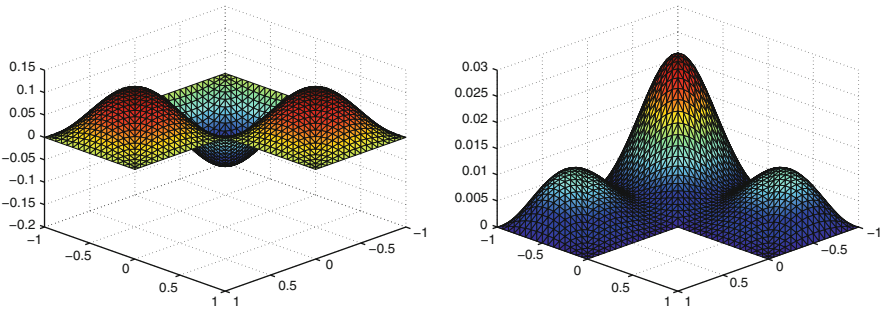


Fig. 3 The expectation and the standard deviation of the solution in case of the load vector $f(\mathbf{x}) = 2\pi^2 \sin(\pi x_1) \sin(\pi x_2)$

9.1 Finite Dimensional Stochastics

The first numerical example treats the stochastic diffusion problem with diffusion coefficient determined by the statistics

$$\begin{aligned} \mathbb{E}_\alpha(\mathbf{x}) &= 6.5 + c_1(\mathbf{x}) + c_2(\mathbf{x}) + c_3(\mathbf{x}), \\ \mathrm{Cov}_\alpha(\mathbf{x}, \mathbf{y}) &= c_1(\mathbf{x})c_1(\mathbf{y}) + c_2(\mathbf{x})c_2(\mathbf{y}) + c_3(\mathbf{x})c_3(\mathbf{y}), \end{aligned} \tag{20}$$

where the coefficient functions are given by

$$c_1(\mathbf{x}) = x_1 x_2, \quad c_2(\mathbf{x}) = x_2(1 - x_1), \quad c_2(\mathbf{x}) = x_1(1 - x_2).$$

By assuming further that the emerging random variables of the Karhunen-Loève expansion are independent and uniformly distributed on $[-1, 1]$, we obtain the uniqueness of the respective diffusion coefficient. Note that the covariance function is intrinsically of rank 3 which results in the stochastic dimension $m = 3$. Hence, this example fits perfectly into the considered setting of this article and is especially well suited for the use of the multilevel quasi Monte Carlo method.

We consider two different load vectors for our computations, namely

$$f(\mathbf{x}) = \exp(0.5x_1 + x_2) \text{ and } f(\mathbf{x}) = 2\pi^2 \sin(\pi x_1) \sin(\pi x_2).$$

Figure 3 shows an illustration of the mean and the standard deviation of the solution which corresponds to the first load vector. Whereas, Figure 4 shows the mean and the standard deviation of the solution which corresponds to the second load vector.

We use the pivoted Cholesky decomposition to calculate the Karhunen-Loève expansion of the diffusion coefficient. The pivoted Cholesky decomposition

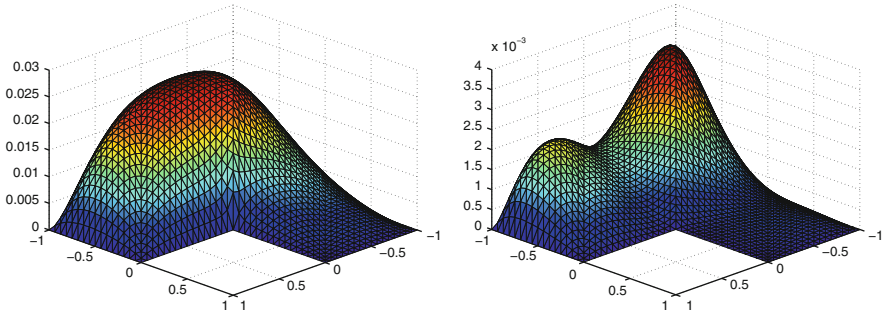


Fig. 4 The expectation and the standard deviation of the solution in case of the load vector $f(\mathbf{x}) = \exp(0.5x_1 + x_2)$

provides a simple method to compute a low-rank approximation to the covariance operator. It converges optimally if the eigenvalues of the covariance operator decay sufficiently fast. In particular, if the rank of the covariance operator is finite, this is captured by the algorithm. The major advantage of this approach is that, at any time, the cut-off error of the Karhunen-Loève expansion is rigorously controlled in terms of the trace norm, cf. [13].

In the following, we denote the discretized expectation and the discretized covariance operator of the diffusion coefficient by $\mathbf{E}_\alpha \in \mathbb{R}^d$ and $\mathbf{A} \in \mathbb{R}^{d\times d}$, respectively. The arising mass matrix is neglected due to the choice of L^2-orthonormalized ansatz and test functions. Now the pivoted Cholesky decomposition yields the approximation

$$\|\mathbf{A} - \mathbf{L}\mathbf{L}^\mathsf{T}\|_{\mathrm{tr}} \leq \varepsilon$$

for some prescribed precision $\varepsilon > 0$ and a low-rank matrix $\mathbf{L} \in \mathbb{R}^{d\times m}$ with $m \ll d$. Since the eigenvalues of $\mathbf{L}\mathbf{L}^\mathsf{T} \in \mathbb{R}^{d\times d}$ coincide with those of $\mathbf{L}^\mathsf{T}\mathbf{L} \in \mathbb{R}^{m\times m}$, only a small eigenvalue problem has to be solved. Here, a reasonable speed-up can be achieved in comparison with other subspace methods for eigenvalue computation. If $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ denote the orthonormal eigenvectors of the small problem, the eigenvectors of the large matrix are given by $\mathbf{L}\mathbf{v}_1, \mathbf{L}\mathbf{v}_2, \ldots, \mathbf{L}\mathbf{v}_m$. Obviously, we have

$$\mathbf{L}\mathbf{L}^\mathsf{T}(\mathbf{L}\mathbf{v}_i) = \mathbf{L}(\mathbf{L}^\mathsf{T}\mathbf{L}\mathbf{v}_i) = \lambda_i(\mathbf{L}\mathbf{v}_i)$$

for all $i = 1, 2, \ldots, m$. Due to

$$(\mathbf{L}\mathbf{v}_i)^\mathsf{T}(\mathbf{L}\mathbf{v}_j) = \mathbf{v}_i^\mathsf{T}\mathbf{L}^\mathsf{T}\mathbf{L}\mathbf{v}_j, = \lambda_i\delta_{i,j}$$

with $\delta_{i,j}$ being the Kronecker symbol, a rescaling of the eigenvectors has to be performed. Consequently, the discretized Karhunen-Loève expansion of the diffusion coefficient is simply given by

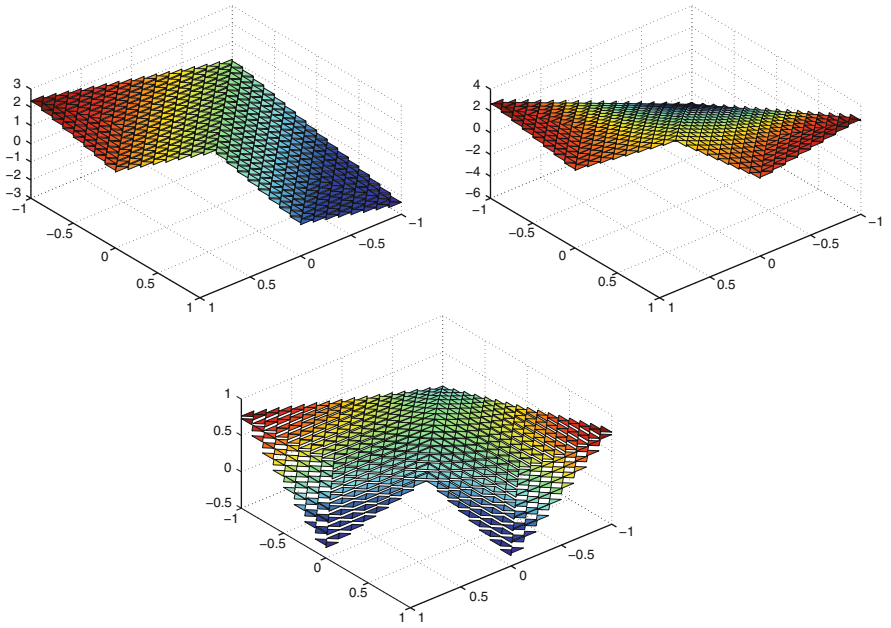


Fig. 5 The scaled Karhunen-Loève modes of the covariance operator given by (19)

$$\alpha(\mathbf{y}) = \mathbf{E}_\alpha + \sum_{k=1}^{m} \mathbf{L}\mathbf{v}_k y_k .$$

Figure 5 shows an illustration of these scaled, elementwise constant Karhunen-Loève modes of the covariance operator.

For the analysis of the convergence rates of the different quadrature rules, we average the solutions of 10 runs of the multilevel Monte Carlo method on level 9. This corresponds to more than three million elements, with about 10^7 samples on the related coarse spatial mesh. As we pointed out in this article, this reference solution is, up to a logarithmic factor, as accurate as the standard Monte Carlo method for the finite element discretization on level 9 with the same number of samples. Likewise, for the following convergence studies, we average the results of 10 runs of the multilevel Monte Carlo method on each particular level.

Figure 6 shows the convergence of the solution's expectation and its second moment in case of the load vector $f(\mathbf{x}) = 2\pi^2 \sin(\pi x_1)\sin(\pi x_2)$. The three different error curves behave quite similar. The dashed line corresponds to the expected convergence rate 2^{-j}. We however observe even a rate about 4^{-j} which is much better than expected. We assume that the over-all error is still limited by the actual accuracy of the finite element approximation.

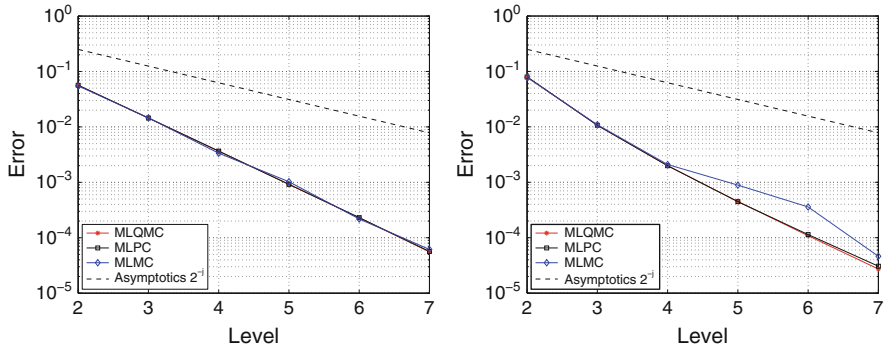


Fig. 6 Relative errors to the expectation and the second moment of the solution in case of $f(\mathbf{x}) = 2\pi^2 \sin(\pi x_1) \sin(\pi x_2)$

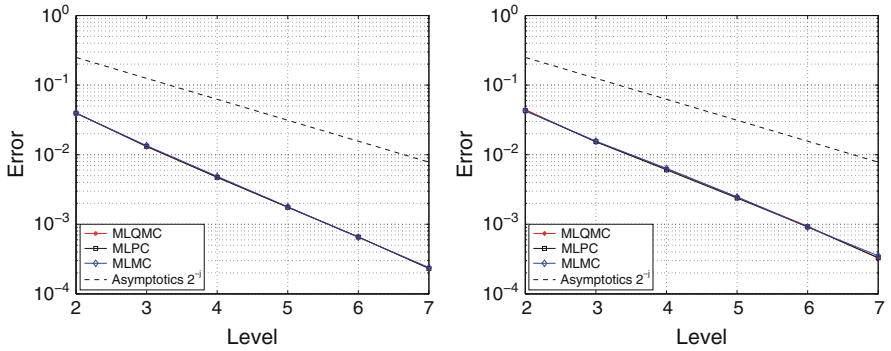


Fig. 7 Relative errors to the expectation and the second moment of the solution in case of $f(\mathbf{x}) = \exp(0.5x_1 + x_2)$

Figure 7 presents the convergence of the considered methods with respect to the load vector $f(\mathbf{x}) = \exp(0.5x_1 + x_2)$. Here, the observed convergence rate is lower than in the first example, but still better than the expected convergence rate of 2^{-j}.

9.2 Infinite Dimensional Stochastics

The second example involves a covariance function of infinite rank. Namely, we consider

$$\mathbb{E}_\alpha(\mathbf{x}) = 10, \quad \mathrm{Cov}_\alpha(\mathbf{x}, \mathbf{y}) = \frac{10}{7} \exp\left(-\frac{100}{49}\|\mathbf{x} - \mathbf{y}\|_2^2\right). \tag{21}$$

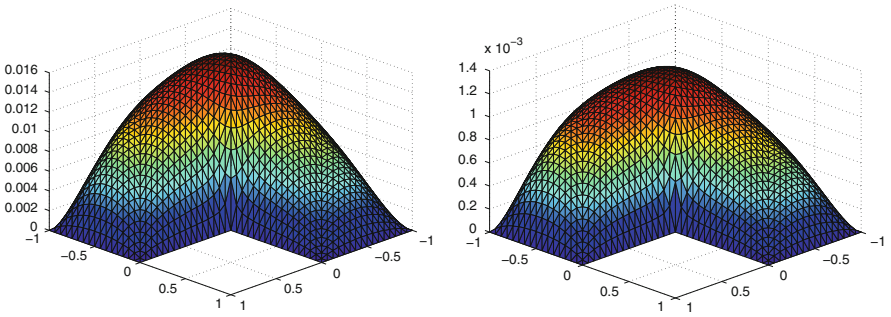


Fig. 8 The expectation and the standard deviation of the solution

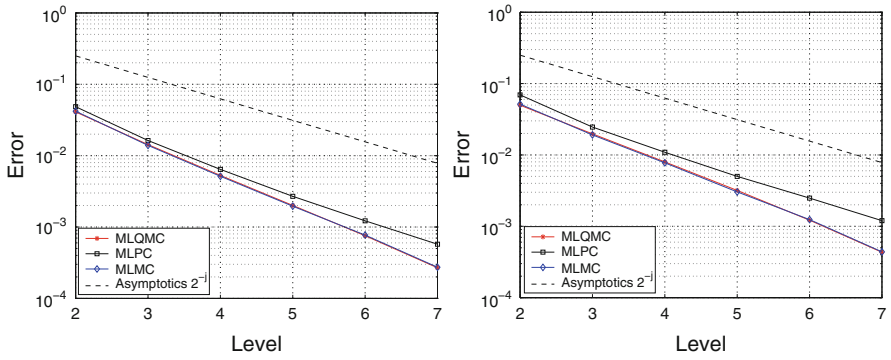


Fig. 9 Relative errors to the expectation and the second moment of the solution

Thus, we have to compute a finite approximation to the corresponding Karhunen-Loève expansion. How the approximation error affects the solution of the respective diffusion problem is investigated in [20]. The load vector under consideration is

$$f(\mathbf{x}) = \exp(0.1x_1 + 0.2x_2).$$

The computations for this example are carried out analogously to the previous example. The only difference is that the Karhunen-Loève expansion needs to be appropriately truncated. For example, it consists of 53 terms for the reference solution and of 39 terms for the calculation on level 7. The mean and the standard deviation of the random solution are found in Figure 8.

Figure 9 shows the convergence behaviour of the different quadrature rules under consideration. Now, the convergence rate fits exactly the expected decay 2^{-j} of the error in the case of the multilevel polynomial chaos method. In the case of the multilevel Monte Carlo method and the multilevel quasi Monte Carlo method the rates are better, although the choice of the number of quadrature points for the multilevel quasi Monte Carlo method is heuristically.

References

1. A. Barth, C. Schwab, and N. Zollinger. Multi-level monte carlo finite element method for elliptic PDEs with stochastic coefficients. *Numer. Math.*, 119(1):123–161, 2011.
2. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numer.*, 13:147–269, 2004.
3. E. W. Cheney and W. A. Light. *Approximation Theory in Tensor Product Spaces*. Springer, Berlin, 1980.
4. A. Cohen, R. DeVore, and C. Schwab. Convergence rates of best N-term Galerkin approximations for a class of elliptic sPDEs. *Found. Comput. Math.*, 10:615–646, 2010.
5. J. Diestel and J. J. Uhl, Jr. *Vector Measures*. American Mathematical Society, Providence, 1979.
6. P. Frauenfelder, C. Schwab, and R. Todor. Finite elements for elliptic problems with stochastic coefficients. *Comput. Methods Appl. Mech. Engrg.*, 194(2–5):205–228, 2005.
7. T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numer. Algorithms*, 18:209–232, 1998.
8. T. Gerstner and S. Heinz. Dimension- and time-adaptive multilevel Monte Carlo methods. In J. Garcke and M. Griebel, editors, *Sparse Grids and Applications*, LNCSE. Springer, 107–120, 2012.
9. R. Ghanem and P. Spanos. *Stochastic finite elements: a spectral approach*. Springer-Verlag, New York, 1991.
10. M. Giles. Multilevel Monte Carlo path simulation. *Operations Research*, 56(3):607–617, 2008.
11. M. Giles and B. Waterhouse. Multilevel quasi-Monte Carlo path simulation. *Radon Series Comp. Appl. Math.*, 8:1–18, 2009.
12. M. Griebel and H. Harbrecht. On the construction of sparse tensor product spaces. *Preprint 497, Berichtsreihe des SFB 611, Universität Bonn*, 2011. (to appear in Math. Comput.).
13. H. Harbrecht, M. Peters, and R. Schneider. On the low-rank approximation by the pivoted Cholesky decomposition. *Appl. Numer. Math.*, 62(4):428–440, 2012.
14. M. Hegland. Adaptive sparse grids. *ANZIAM J.*, 44:C335–C353, 2002.
15. S. Heinrich. The multilevel method of dependent tests. In *Advances in stochastic simulation methods (St. Petersburg, 1998)*, Stat. Ind. Technol., pages 47–61. Birkhäuser Boston, Boston, MA, 2000.
16. S. Heinrich. Multilevel Monte Carlo methods. In *Lect. Notes in Large Scale Scientific Computing*, pages 58–67, London, 2001. Springer-Verlag.
17. M. Loève. *Probability theory. I+II*, volume 45 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 4th edition, 1977.
18. H. Niederreiter. *Random number generation and quasi-Monte Carlo methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.
19. P. Protter. *Stochastic integration and differential equations: a new approach*. Springer, Berlin, 3rd edition, 1995.
20. C. Schwab and R. Todor. Karhunen-Loève approximation of random fields by generalized fast multipole methods. *J. Comput. Phys.*, 217:100–122, 2006.
21. R. Todor and C. Schwab. Convergence rates for sparse chaos approximations of elliptic problems with stochastic coefficients. *IMA J. Numer. Anal.*, 27(2):232–261, 2007.
22. C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations*, volume 31 of *Notes on Numerical Fluid Mechanics*, pages 241–251, Braunschweig/Wiesbaden, 1991. Vieweg-Verlag.

Local and Dimension Adaptive Stochastic Collocation for Uncertainty Quantification

John D. Jakeman and Stephen G. Roberts

Abstract In this paper we present a stochastic collocation method for quantifying uncertainty in models with large numbers of uncertain inputs and non-smooth input-output maps. The proposed algorithm combines the strengths of dimension adaptivity and hierarchical surplus guided local adaptivity to facilitate computationally efficient approximation of models with bifurcations/discontinuties in high-dimensional input spaces. A comparison is made against two existing stochastic collocation methods and found, in the cases tested, to significantly reduce the number of model evaluations needed to construct an accurate surrogate model. The proposed method is then used to quantify uncertainty in a model of flow through porous media with an unknown permeability field. A Karhunen–Loève expansion is used to parameterize the uncertainty and the resulting mean and variance in the speed of the fluid and the time dependent saturation front are computed.

1 Introduction

Knowledge of the sources and effects of uncertainty is needed for effective decision making, policy and planning [20]. Once model uncertainties have been identified and classified, the effects of such uncertainty should be traced through the system thoroughly enough to allow one to evaluate their effects on the intended use of the model. The models can be highly complex, nonlinear and often discontinuous. In

J.D. Jakeman (✉)
Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA
e-mail: jjakeman@purdue.edu

S.G. Roberts
Mathematical Sciences Institute, Australian National University, Canberra ACT 0200, Australia
e-mail: stephen.roberts@anu.edu.au

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_9,

these situations computational methods and software tools are needed to facilitate analysis of model uncertainty. The most appropriate method and tool is dependent on the goals of the uncertainty quantification analysis.

Recently stochastic Galerkin (SG) [2, 10, 25, 27] methods have emerged as efficient and highly competitive means of constructing accurate surrogate models in high dimensional spaces. SG methods are highly suited to dealing with ordinary and partial differential equations and have the ability to deal with steep nonlinear dependence of the solution on random model data [16]. Provided sufficient smoothness conditions are met, SG estimates of uncertainty converge exponentially with the order of the expansion and, for low dimensions, come with a small computational cost. Despite these advantages the intrusive nature of these methods, which requires the modification of often complex deterministic code, can make the application of SG infeasible.

Stochastic collocation (SC) [1, 19, 23, 26] is an alternative that is gaining popularity. SC endeavours to combine the strengths of non-intrusive sampling and SG. As with Monte Carlo methods, SC requires only the solution of a set of decoupled equations, allowing the model to be treated as a black box and run with existing deterministic solvers.

Sparse grid interpolation has emerged as one of the most useful tools to construct the multi-dimensional approximation. Sparse grids have been extensively used for high-dimensional interpolation and quadrature [3,8]. The approach can yield several orders of magnitude reduction in the number of collocation points required to achieve the same level of accuracy as the full tensor grid [22, 26].

Standard sparse grids are isotropic, treating all dimensions equally. Although an advance on full tensor product spaces, such approximations can still be improved. Many problems vary rapidly in only some dimensions, remaining much smoother in other dimensions. Consequently it is advantageous to increase the level of accuracy only in certain non-smooth dimensions, resulting in so-called adaptive or anisotropic grids. In some cases the important dimensions can be determined a priori, but in most cases the collocation points must be chosen during the computational procedure. Examples of dimension-adaptive sparse grid collocation schemes are given by Gana [7] and Nobile [21].

Ma and Zabaras [17, 18] developed an adaptive sparse grid collocation method called Adaptive Sparse Grid Collocation (ASGC). This method uses piecewise multi-linear hierarchical basis functions, based upon the wavelet based representations found in the deterministic sparse grid literature [11, 24]. ASGC achieves the benefits of local adaptivity of the ME-gPC and Wiener–Harr expansions whilst remaining non-intrusive and allowing application to higher-dimensional problems.

The local refinement of the sparse grid is guided by the magnitude of the so-called hierarchical surplus, which is the difference between the true function and the approximation at a new grid point before that point is used in the interpolation. The resulting method automatically concentrates function evaluations in rapidly varying or discontinuous regions.

ASGC is implicitly dimension-adaptive and so the efficiency of this method can be increased by including explicit dimension adaptivity. With this goal, Ma [18] developed ASGC-HDMR which combines ASGC with a dimension-adaptive ANOVA decomposition that iteratively constructs the important sub-dimensional components of the anchored ANOVA decomposition of a function. This extension provides a significant improvement in efficiency.

In this paper we present a stochastic collocation method based upon dimension and locally-adaptive sparse grid algorithm designed for interpolating high-dimensional functions with discontinuities [15]. The underlying dimension adaptation procedure greedily identifies the model variables and variable interactions that contribute most to the variability of the model. The hierarchical surplus at each point within the sparse grid is used as an error indicator to guide local refinement, with the aim of concentrating computational effort within rapidly varying or discontinuous regions. This approach limits the number of points that are invested in 'unimportant' dimensions and regions within the high-dimensional domain. Piecewise polynomial bases are used to take advantage of any smoothness in the model input-output map. The finite support of these basis functions facilitates local adaptation which is not possible with global polynomial basis functions.

The stochastic collocation method is presented in Sect. 2 and the so called h-GSG on which it is based is summarised in Sect. 3. To facilitate a clear discussion the ASGC-HDMR is presented in Sect. 4 and then compared to adaptive generalised sparse grid stochastic collocation methods and the h-GSG stochastic collocation method proposed in this manuscript. The paper is concluded by the application of the proposed method to the quantification of uncertainty in a model of flow through porous media.

2 Problem Formulation

Consider a model M defined on an s-dimensional bounded domain $D \in \mathbb{R}^s$ ($s = 1, 2, 3$)

$$y(\mathbf{x}, t, \boldsymbol{\xi}) = M(\mathbf{x}, t, \boldsymbol{\xi}), \quad \mathbf{x} \in D,\ t \in \mathcal{T},\ \boldsymbol{\xi} \in I_{\boldsymbol{\xi}} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_s)$ are the coordinates in $\mathbb{R}^s$, $s \geq 1$, $\mathcal{T} = (t_0, T]$ are the temporal coordinates in $\mathbb{R}$, and $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_d) \in I_{\boldsymbol{\xi}} \subset \mathbb{R}^d$, $d \geq 1$ is a d-dimensional vector of variables which describe the uncertainty in the model inputs.

We are interested in finding the stochastic solution

$$y(\mathbf{x}, t, \boldsymbol{\xi}) : \bar{D} \times \mathcal{T} \times I_{\boldsymbol{\xi}} \to \mathbb{R}^{n_y}$$

Without loss of generality, hereafter we assume (1) is a scalar system with $n_y = 1$. We also make the fundamental assumption that the problem (1) is well posed in $I_{\boldsymbol{\xi}}$, where $I_{\boldsymbol{\xi}} = I_{\xi_1} \times \cdots \times I_{\xi_d}$ and I_{ξ_n} are the one-dimensional ranges of the variables $\xi_n, n = 1, \ldots, d$.

Traditionally most uncertainty quantification (UQ) studies adopt a probabilistic formulation to quantify the input uncertainty. It is typically assumed that the distribution of the random variables $\boldsymbol{\xi}$ is known, with the most widely adopted approach assuming the marginal distributions of ξ_i are known and all ξ_i are independent from each other. In this paper, however, we wish to also consider the case where the uncertainty is *epistemic*. That is, the distribution functions of ξ_i are not known, primarily due to our lack of understanding and characterization of the physical system governed by the system of Eqs. (1). Under such conditions the focus of UQ analysis is on the discovering the dependence of the solution on the uncertain inputs $\boldsymbol{\xi}$.

Stochastic collocation (SC) [1, 19, 23, 26] can be used to quantify both aleatory and epistemic uncertainty [14]. The stochastic collocation method is equivalent to solving N deterministic problems at a set of nodal points $\Theta_N = \{\boldsymbol{\xi}^{(k)}\}_{k=1}^N$ and constructing an approximation that interpolates the solution y at each node $\boldsymbol{\xi}^{(k)} = (\xi_1^{(k)}, \ldots, \xi_d^{(k)})$.

That is given a set of nodes $\Theta_N = \{\boldsymbol{\xi}^{(k)}\}_{k=1}^N \in I_{\boldsymbol{\xi}}$ we solve

$$y(\boldsymbol{\xi}^{(k)}) = M(\boldsymbol{\xi}^{(k)}) \quad k = 1, \ldots, N \tag{2}$$

to obtain an associated set of solutions $y_N = \{y(\boldsymbol{\xi}^{(k)})\}_{k=1}^N$. This ensemble of solutions is then used to build an approximation that enforces (1) at each node. The problem (2) is decoupled and existing deterministic solvers can be applied to find each realization of the solution.

The choice of N is made in accordance to the computational budget of the user. The exact value of N is set by the method used to construct the approximation of (1), and depends on the computational cost of the evaluating the model, and the desired accuracy of the approximation. Throughout this paper we assume that the evaluation of the model 2 is expensive and significantly outweighs the cost of constructing the approximation. Consequently we would like to develop a collocation method that for a given accuracy, N is smaller than the number of samples required by existing methods to obtain the same accuracy.

3 A Stochastic Collocation Method

While the general strategy of stochastic collocation is straightforward, the options for practical implementation are limited. Multivariate approximation is a challenging area with many open issues. In the following we propose a stochastic collocation method based on the locally and dimension-adaptive sparse grid interpolation method proposed in [15]. We summarize this self-titled h-Adaptive Generalised Sparse Grid (h-GSG) method here.

3.1 Sparse Grids

Let $y : \mathbb{R}^d \to \mathbb{R}$ be an unknown multi-variate function with d uncertain inputs in some function space V. We may not know the closed form of y and only require that we can evaluate y at arbitrary points in $I_{\boldsymbol{\xi}}$ using a numerical code. Here we will restrict attention to consider stochastic collocation problems characterised by variables $\boldsymbol{\xi}$ with finite support normalized to fit in the domain $I_{\boldsymbol{\xi}} = [0, 1]^d$.

Sparse grids [5] approximate functions $y \in V$ via a weighted linear combination of basis functions

$$y_N(\boldsymbol{\xi}) = \sum_{k=1}^{N} v_k \, \Psi_k(\boldsymbol{\xi}) \tag{3}$$

The approximation is constructed on a set of anisotropic grids $\Omega_{\mathbf{l}}$ on the domain $I_{\boldsymbol{\xi}}$ where $\mathbf{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$ is a multi-index denoting the level of refinement of the grid in each dimension d. These rectangular grids have mesh size

$$\mathbf{h_l} = (h_{l_1}, \ldots, h_{l_d}), \quad \text{where} \quad h_l = 2^{-l} \text{ if } l_n > 1, \; h_0 = 0.5$$

and consist of the points

$$\boldsymbol{\xi}_{\mathbf{l},\mathbf{i}} = (\xi_{l_1,i_1}, \ldots, \xi_{l_d,i_d}) = (i_1 h_1, \cdots, i_d h_d), \quad 0 \le i_n \le 2^{l_n} \text{ if } l_n > 1, \; i_n = 1 \text{ if } l_n = 0, \; \forall \, n$$

where $\mathbf{i}$ denotes the location of a given grid point. The one-dimensional grid is shown in Fig. 1a.

The multivariate basis functions Ψ_k are a tensor product of one dimensional basis functions. Adopting the multi-index notation use above we have

$$\Psi_{\mathbf{l},\mathbf{i}}(\boldsymbol{\xi}) = \prod_{n=1}^{d} \psi_{l_n,i_n}(\xi_n) \tag{4}$$

where there is a one-to-one relationship between Ψ_k in (3) and $\Psi_{\mathbf{l},\mathbf{i}}$ and each $\Psi_{\mathbf{l},\mathbf{i}}$ is uniquely associated with a grid point $\boldsymbol{\xi}_{\mathbf{l},\mathbf{i}}$. Many different one-dimensional basis functions $\psi_{l_n,i_n}(\xi_n)$ can be used. One such choice is the multi-linear piecewise basis based upon the one-dimensional formula

$$\psi_{l,i}(\xi) = \begin{cases} 1 & \text{if } l = 0 \\ \max(1 - 2\xi, 0) & \text{if } l = 1 \wedge i = 0 \\ \max(2\xi - 1, 0) & \text{if } l = 1 \wedge i = 2 \\ \psi(2^l \xi - i) & \text{otherwise,} \end{cases} \tag{5}$$

where $\psi(\xi) = \max(1 - |\xi|, 0)$. The one-dimension linear basis is shown in Fig. 1a. Alternative basis functions include the piecewise polynomial basis functions $\Psi^{(\mathbf{p})}_{\mathbf{l},\mathbf{i}}(\boldsymbol{\xi})$ which are the tensor product of d one-dimensional basis functions $\psi^{(p_n)}_{l_n,i_n}(\xi_n)$

$$\Psi_{\mathbf{l},\mathbf{i}}^{(\mathbf{p})}(\boldsymbol{\xi}) = \prod_{n=1}^{d} \psi_{l_n,i_n}^{(p_n)}(\xi_n)$$

where we have introduced the superscript $\mathbf{p} = (p_1, \ldots, p_d)$ to denote the degree of the piecewise polynomial in each dimension. The exact construction of these polynomials can be found in [15].

The multi-dimensional basis formed using this basis in conjunction with (4) span the discrete space

$$V_{\mathbf{l}} = \text{span}\left\{\Psi_{\mathbf{l},\mathbf{i}} : \mathbf{i} \in \mathbf{K}_{\mathbf{l}}\right\} \quad \mathbf{K}_{\mathbf{l}} = \{\mathbf{i} : i_n = 0, \ldots, 2^{l_n} \text{ if } l_n > 0,\ i_n = 1 \text{ if } l_n = 0, \forall\, n\}$$

To facilitate adaptivity we introduce the hierarchical difference spaces

$$W_{\mathbf{l}} = \text{span}\,\{\Psi_{\mathbf{l},\mathbf{i}} \,:\, \mathbf{i} \in \mathbf{I}_{\mathbf{l}}\}$$

where

$$\mathbf{I}_{\mathbf{l}} = \{\mathbf{i} : i_n = 0, \ldots, 2^{l_n},\ i_n \text{ odd if } l_n \neq 1, i_n = 0, 2 \text{ if } l_n = 1,\ \forall\, n\}, \tag{6}$$

Note that this index set is different to the set typically presented in the literature [6]. The traditional index set only deals with points on the interior of $\Omega_{\mathbf{l}}$ and thus assumes the functions being approximated have homogeneous boundary conditions. The index set presented here allows for points on the boundary which facilitates approximation of functions with inhomogeneous boundaries.

The hierachical index set (6) defines the hierarchical basis

$$\{\Psi_{\mathbf{k},\mathbf{i}} : \mathbf{k} \leq \mathbf{l}, \mathbf{i} \in \mathbf{I}_{\mathbf{k}}\}$$

which also spans $V_{\mathbf{i}}$. The one-dimensional hierarchical basis one-dimensional mesh are shown in Fig. 1b.

These hierarchical difference spaces can be used to decompose the input space such that for a functions space V

$$V_{\mathbf{l}} = \bigoplus_{\mathbf{k} \leq \mathbf{l}} W_{\mathbf{l}} \quad \text{and} \quad V = \bigoplus_{k_1=0}^{\infty} \cdots \bigoplus_{k_d=0}^{\infty} W_{\mathbf{k}} = \bigoplus_{\mathbf{k} \in \mathbb{R}^d} W_{\mathbf{k}}.$$

For numerical purposes we must truncate the number of difference spaces used to construct V to some level l. Specifically the classical finite dimensional sparse grid space is defined by

$$V_{l,d}^{(1)} = \bigoplus_{|\mathbf{l}|_1 \leq l} W_{\mathbf{l}}$$

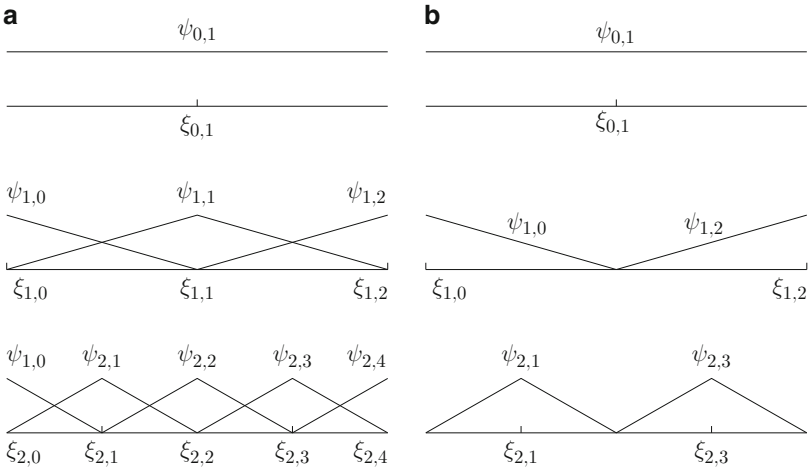


Fig. 1 One-dimensional nodal and hierachical bases for level $l = 0, 1, 2$. (**a**) Nodal basis. (**b**) Hierarchical basis

With such a decomposition any function $y(\boldsymbol{\xi}) \in V$ can be approximated by

$$y_{\mathcal{L}}(\boldsymbol{\xi}) = \sum_{\mathbf{l} \in \mathcal{L}} y_{\mathbf{l}}(\boldsymbol{\xi}), \quad y_{\mathbf{l}}(\boldsymbol{\xi}) = \sum_{\mathbf{i} \in \mathbf{I_l}} v_{\mathbf{l},\mathbf{i}} \, \Psi_{\mathbf{l},\mathbf{i}} \tag{7}$$

where for traditional isotropic sparse grids

$$\mathcal{L} = \{\mathbf{l} : |\mathbf{l}|_1 \leq l\} \tag{8}$$

Note the $v_{\mathbf{i},\mathbf{j}} \in \mathbb{R}$ are the coefficient values of the hierarchical product basis, also known as the hierarchical surplus. They are simply the difference between the function value $y(\boldsymbol{\xi}_{\mathbf{l},\mathbf{i}})$ and $\hat{y}_{\mathcal{L}}(\boldsymbol{\xi}_{\mathbf{l},\mathbf{i}})$ where $\hat{y}_{\mathcal{L}}$ does not contain the the basis function $\Psi_{\mathbf{l},\mathbf{i}}$.

3.2 Moment Estimation

The extension from interpolation to quadrature is straightforward. We can approximate the integral of a function y

$$I[y(\mathbf{x}, t)] = \int_{I_{\boldsymbol{\xi}}} y(\mathbf{x}, t, \boldsymbol{\xi}) \, d\mu(\boldsymbol{\xi})$$

using the hierarchical sparse grid interpolant (7). Utilizing this formulation these integrals can be approximated by

$$I[y_{l,d}(\mathbf{x},t)] = \int_{I_\xi} \sum_{\mathbf{l}\in\mathcal{I}} \sum_{\mathbf{i}\in\mathbf{I_l}} v_{\mathbf{l},\mathbf{i}}(\mathbf{x},t)\,\Psi_{\mathbf{l},\mathbf{i}}(\boldsymbol{\xi})\,d\mu(\boldsymbol{\xi})$$
$$= \sum_{\mathbf{l}\in\mathcal{I}} \sum_{\mathbf{i}\in\mathbf{I_l}} v_{\mathbf{l},\mathbf{i}}(\mathbf{x},t)\,w_{\mathbf{l},\mathbf{i}}$$

where the weights

$$w_{\mathbf{l},\mathbf{i}} = \int_{I_\xi} \Psi_{\mathbf{l},\mathbf{i}}(\boldsymbol{\xi})\,d\mu(\boldsymbol{\xi})$$

can be calculated easily and with no need for extra function evaluations once the interpolant has been constructed. One simply needs to store the volumes of the high-order basis functions. For $d\mu = 1$ these volumes can be calculated analytically.

3.3 Adaptation Procedure

We are not limited to this formulation of the hierarchical level index set $\mathcal{L}$ in (8). The index set $\mathcal{L}$ can be used to control the importance of each variable and the importance of any interaction between a subset of a function's variables. The classical sparse grid construction (8) treats all dimensions equally and assumes that the importance of any interaction between a subset of a function's variables decreases as the number of variables involved in the interaction (interaction order) increases. In practice, often only a small subset of variables and interactions contribute significantly to the variability of the function y. To utilise this property, the h-GSG method constructs $\mathcal{L}$ iteratively and restricts grid points (function evaluations) to dimensions and interactions that contribute significantly to the function variability, according to some predefined measure.

The index sets $\mathbf{I_l}$ control the location of the points $\boldsymbol{\xi}_{\mathbf{l},\mathbf{i}}$ in the input domain I_ξ. The traditional construction in (6) considers all regions of the input space uniformly. Frequently, however, only small regions within the input space possess high variability. Ideally function evaluations should be concentrated in rapidly varying or discontinuous regions. The coefficients v in (7), known as the hierarchical surpluses, provide natural error indicators which can be used to adaptively select points to be included in the sets $\mathbf{I_l}$. In existing methods, typically either the set $\mathcal{L}$ or the sets $\mathbf{I_l}$ is adapted. The h-GSG method overcomes this restriction and constructs the index sets simultaneously, resulting in a method that is both dimension and locally (regionally) adaptive. The adaption procedure is presented in Sect. 3.3.

The dimension adaptivity of the h-GSG method is based upon a generalisation of the traditional isotropic sparse grid index set (8). Specifically the h-GSG method follows the pioneering work of Gerstner and Griebel [9] and Hegland [13] and considers all index sets that satisfy the admissibility criterion

$$\mathbf{l} - \mathbf{e}_j \in \mathcal{I} \text{ for } 1 \le j \le d,\, l_j > 1 \tag{9}$$

This generalisation is extremely effective at determining the hierarchical difference spaces that contribute significantly to the function variability, according to some predefined measure. The dimension-adaptive procedure is a 'greedy' approach which attempts to find the index set $\mathcal{I}$ such that for a given number of points the approximation error is minimized. Admissible grid indices $\mathbf{l}$ are only added to the index set $\mathcal{L}$ if an error estimator $r_{\mathbf{l}} \geq \varepsilon$, where $\varepsilon > 0$ reflects the desired accuracy in the interpolant. Throughout this paper we employ the error measure

$$r_{\mathbf{l}} = \left| \sum_{\mathbf{i} \in \mathbf{I}_{\mathbf{l}}} v_{\mathbf{l},\mathbf{i}} \, w_{\mathbf{l},\mathbf{i}} \right|$$

This limits the creation of sub-components $y_{\mathbf{l}}$ with $r_{\mathbf{l}} << \varepsilon$ which have little effect on the accuracy of the approximation.

When using the traditional approach, each time a grid index $\mathbf{l}$ is added to the sparse grid index set $\mathcal{L}$ the function $y(\boldsymbol{\xi})$ is evaluated at all the points in the set $\mathbf{I}_{\mathbf{l}}$. Such an approach is inefficient if a large proportion of the function variability is concentrated in small regions of the input space. When using equidistant grids, the creation of the grid $\mathbf{l}$ requires approximately two times the number of grid points (and thus function evaluations) than those necessary to construct the index $\mathbf{l} - \mathbf{e}_j$.

To incorporate local adaptivity into the dimension selection the h-GSG method defines the two sets $\mathcal{A}_{\mathbf{l}}$ and $\mathcal{R}_{\mathbf{l}}$ for each grid index $\mathbf{l}$. These sets are referred to as the active point set and redundant point set of the grid index $\mathbf{l}$, respectively . The active point set $\mathcal{A}_{\mathbf{l}}$ contains all admissible points associated with the index $\mathbf{l}$ with an error indicator $\gamma_{\mathbf{l},\mathbf{i}} \geq \varepsilon$. The redundant point set $\mathcal{R}_{\mathbf{l}}$ contains all admissible points with $\gamma_{\mathbf{l},\mathbf{i}} < \varepsilon$. A point is admissible if one of its d possible ancestors exists in the grids associated with the backwards neighbourhood of $\mathbf{l}$. The backwards neighbourhood of an index is the set of d indices $\{\mathbf{l} - \mathbf{e}_j : 1 \leq j \leq d\}$. If, and only if, a grid point is admissible it is created (the function is evaluated) and the error indicator $\gamma_{\mathbf{l},\mathbf{i}}$ calculated. This drastically reduces the number of points generated when a new grid index is created. The form of $\gamma_{\mathbf{l},\mathbf{i}}$ effects the efficiency of h-GSG and is problem dependent.

Figure 2 shows an example of h-adaptivity integrated with the generalised sparse grid algorithm. Here the grid index $\mathbf{l} = (2,2)$ has been deemed admissible by the generalised sparse grid algorithm. Both the backwards neighbours ($\mathbf{l} - \mathbf{e}_1 = (1,2)$ and $\mathbf{l} - \mathbf{e}_1 = (2,1)$) exist in the old index set. The active points in the backwards neighbours are used to determine which points in the active index $\mathbf{l}$ should be evaluated. For any point in the active set of the n-th backwards neighbour $n = 1, \ldots, d$ the children of that point are created in the n-th axial direction. This refinement is carried out for all points in the active point set of the backward neighbour and for all backwards neighbours. The parent-child relationship is shown in Fig. 3.

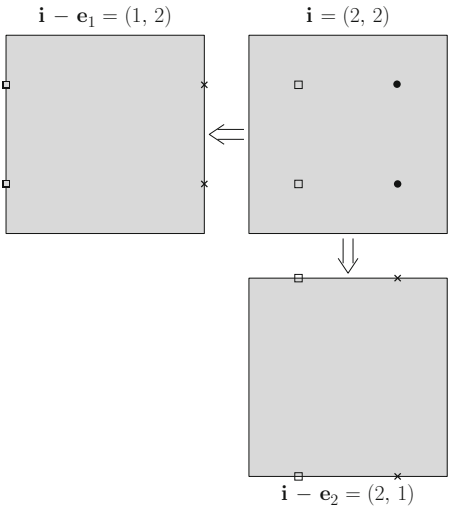


Fig. 2 An example of local adaptation integrated with the generalised sparse grid algorithm. Let us assume that the function only varies significantly in the *left* half of the domain. The *top left* and *bottom right* grids are the backwards neighbours of the grid being created. *Squares* represent points in the active point sets, *crosses* are points in the redundant point sets and *circles* are points associated with $W_{\mathbf{l}}$ that are not created. The active (*square*) points in the backwards neighbours are refined to produce the set of new points that must be added. In this example only two new points (*squares* in grid $\mathbf{l} = (2, 2)$) are added

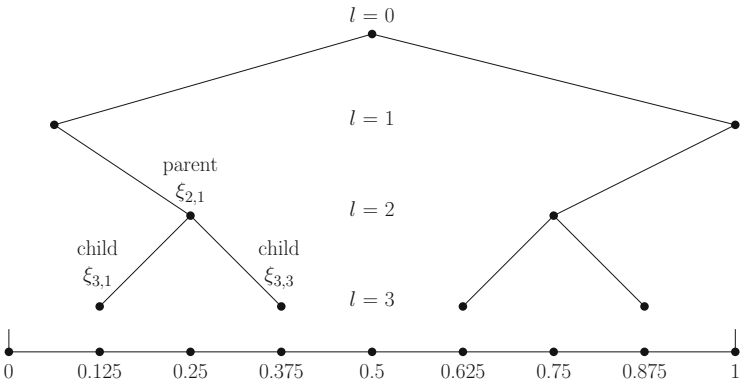


Fig. 3 The dyadic splitting of the one-dimension mesh. Children of a point $\xi_{l,i}$ are the 2 points (in one-dimension, $2d$ in d-dimensions) with level $l + 1$ and connected by one segment in the binary tree

4 An Existing Stochastic Collocation Alternative

The Adaptive Sparse Grid Collocation High Dimensional Model Representation (ASGC-HDMR) method [18] is a recently developed stochastic collocation method that can be used to facilitate UQ analysis of models with possibly hundreds of uncertain input variables. ASGC-HDMR is based upon the ASGC method of Ma and Zabaras [17]. As with the h-GSG method, the ASGC method uses the hierarchical surplus to guide refinement of the sparse grid. If for any basis function $\Psi_{\mathbf{l},\mathbf{i}}$ associated with the grid point $\boldsymbol{\xi}_{\mathbf{l},\mathbf{i}}$ a predefined error indicator $\gamma_{\mathbf{l},\mathbf{i}}$ is above a pre-defined threshold ε, the grid point is flagged for refinement. As with the h-GSG error indicator the form of $\gamma_{\mathbf{l},\mathbf{i}}$ effects the efficiency of the method and is problem dependent.

Refinement involves generating the left and right children of the grid point $\boldsymbol{\xi}_{\mathbf{l},\mathbf{i}}$ in each axial direction. Again see Fig. 3. Due to the binary tree structure of the sparse grid any interior point will have $2d$ children. This refinement strategy is implemented without regard of the effective dimensionality of the problem. Consequently, the efficiency of the method can be improved further by imposing further conditions on when the children of an active point are built. With this goal Ma and Zabaras [18] combined ASGC with a dimension-adaptive scheme to only generate points in dimensions which contribute "significantly" to the mean of the function.

To adaptively select important dimensions, Ma and Zabaras consider the anchored ANOVA decomposition of a function. Specifically given a set $\mathbf{u} \subseteq \mathcal{D} = \{1, \ldots, d\}$, with cardinality $|\mathbf{u}|$, we can define the set of projections $P_{\mathbf{u}} : V^{(d)} \rightarrow V^{|\mathbf{u}|}$ by

$$P_{\mathbf{u}} y(\boldsymbol{\xi}_{\mathbf{u}}) = y(\boldsymbol{\xi})|_{\xi = \mathbf{a} \setminus \xi_{\mathbf{u}}}$$

which projects the d-dimensional function onto a lower dimensional subspace. Here $\boldsymbol{\xi}_{\mathbf{u}}$ denotes the $|\mathbf{u}|$-dimensional vector that contains the components of $\boldsymbol{\xi}$ whose indices belong to the set $\mathbf{u}$ and

$$y(\boldsymbol{\xi})|_{\xi = \mathbf{a} \setminus \xi_n} = f(a_1, \ldots, a_{n-1}, \xi_n, a_{n+1}, \ldots, a_d)$$

Using these projections the function y can be decomposed into the finite sum

$$y(\boldsymbol{\xi}) = \sum_{\mathbf{u} \subseteq \mathcal{D}} y_{\mathbf{u}}(\boldsymbol{\xi}_{\mathbf{u}}) \tag{10}$$

where the sub-dimensional components can be defined recursively by

$$y_{\mathbf{u}}(\boldsymbol{\xi}_{\mathbf{u}}) = P_{\mathbf{u}} y(\boldsymbol{\xi}_{\mathbf{u}}) - \sum_{\mathbf{v} \subset \mathbf{u}} y_{\mathbf{v}}(\boldsymbol{\xi}_{\mathbf{v}})$$

The first term is the zero-th order effect which is a constant throughout the d-dimensional variable space.

Each sub-dimensional component $y_{\mathbf{u}}$ is approximated using the ASGC method described above. Specifically

$$y_{\mathbf{u}} = \sum_{|\mathbf{l}_{\mathbf{u}}| \leq l} v_{\mathbf{l}_{\mathbf{u}},\mathbf{i}_{\mathbf{u}}} \Psi_{\mathbf{l}_{\mathbf{u}},\mathbf{i}_{\mathbf{u}}}$$

where the n-th element of the multi-indices $\mathbf{l}_{\mathbf{u}}$ and $\mathbf{i}_{\mathbf{u}}$ are only non-zero if $n \in \mathbf{u}$.

In practice it is not necessary to compute all the terms $y_{\mathbf{u}}$. Ma and Zabaras utilise a modification of the dimension-adaptive algorithm advocated by Griebel and Holtz [12] to adaptively select the important components. Given a subset $\mathcal{S}$ of all indices $\mathbf{u} \subseteq \mathcal{D}$, the component $\mathbf{v}$ is computed if the admissibility condition

$$\mathbf{u} \in \mathcal{S} \quad \text{and} \quad \mathbf{v} \subset \mathbf{u} \quad \Longrightarrow \quad \mathbf{v} \in \mathcal{S}$$

is satisfied and if the dimension importance indicator $\eta_{\mathbf{u}} \geq \varepsilon$. Throughout this paper we choose

$$\eta_{\mathbf{u}} = |\sum_{|\mathbf{l}_{\mathbf{u}}| \leq l} v_{\mathbf{l}_{\mathbf{u}},\mathbf{i}_{\mathbf{u}}} w_{\mathbf{l}_{\mathbf{u}},\mathbf{i}_{\mathbf{u}}}|$$

as suggested in [18].

5 Method Comparison

In this section we compare the proposed h-GSG stochastic collocation method with stochastic collocation based upon the generalised sparse grid method, and the ASGC-HDMR method [18].

5.1 Algorithmic Comparison

The adaptive procedures of the h-GSG SC and ASGC-HDMR methods construct the sparse index set $\mathcal{I}$ differently. h-GSG proceeds greedily and chooses index sets only if they are admissible according to (9), whereas the ASGC procedure adds indices level by level such that all points $\boldsymbol{\xi}_{\mathbf{l},\mathbf{i}}$ with $|\mathbf{l}|_1 = l$ are considered at once.

Let us chose the functional form of $\gamma_{\mathbf{l},\mathbf{i}}$ to be the same for both methods. The exact form of the error indicator is unimportant for this discussion. The use of the same error indicator is achievable due to the similar local adaptation procedures of the two methods. It is the way in which this local adaptation is combined with dimension adaptivity where the two methods differ.

Figure 4 depicts the different index sets generated by the ASGC-HDMR and h-GSG methods and the resulting grids for a two-dimensional problem. Here we have assumed that any grid not in the left index set has points whose error

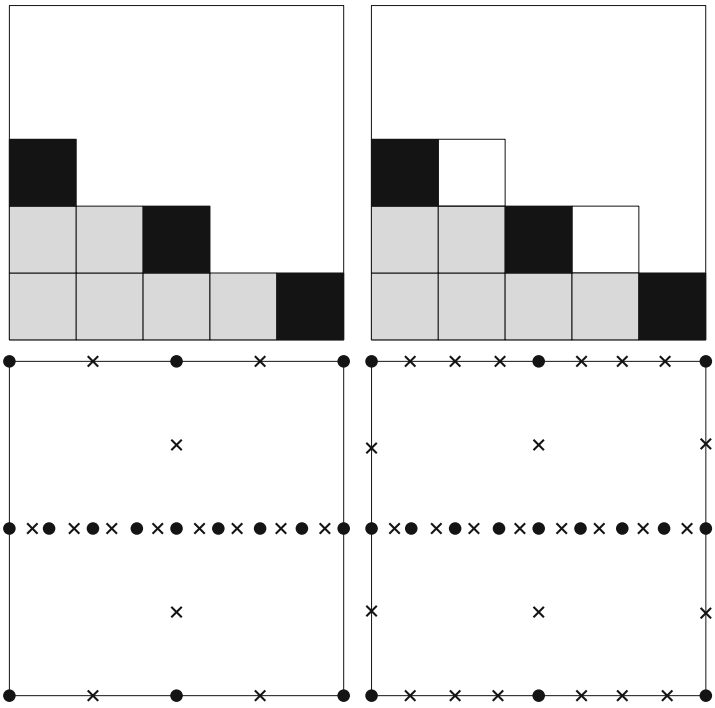


Fig. 4 A possible admissible index set generated by the h-GSG algorithm (*left*) and the corresponding index set generated by ASGC (*right*). *Grey boxes* represent grid indices that contain at least one point with $\gamma_{\mathbf{l},\mathbf{i}} \geq \varepsilon$. *Black boxes* are indices that only contain points with $\gamma_{\mathbf{l},\mathbf{i}} < \varepsilon$. *White boxes* represent grid indices only used by the ASGC-HDMR method. *Circles* represent points with $\gamma_{\mathbf{l},\mathbf{i}} \geq \varepsilon$ and *crosses* depict points with $\gamma_{\mathbf{l},\mathbf{i}} < \varepsilon$.

indicator $\gamma_{\mathbf{l},\mathbf{i}} < \varepsilon$. Grey boxes represent grid indices that contain at least one point with $\gamma_{\mathbf{l},\mathbf{i}} \geq \varepsilon$ and black boxes are indices that only contain points with $\gamma_{\mathbf{l},\mathbf{i}} < \varepsilon$. White boxes represent grid indices only used by the ASGC-HDMR method. In the example shown, two such grid indices are present. The exact number of missing indices and associated grid points is problem dependent.

Figure 5 shows the five steps taken by the ASGC-HDMR algorithm to obtain the index set depicted in Fig. 4. In the example shown, both dimensions are deemed important by ASGC-HDMR and thus we only show the construction of the component $y_{\{1,2\}}$ in (10). Striped boxes indicate indices to be refined, black boxes are grids that only contain points with $\gamma_{\mathbf{l},\mathbf{i}} < \varepsilon$, and grey boxes represent indices already in the sparse grid.

Figures 4 and 5 suggest that given a predefined tolerance and a sub-dimensional problem for which all dimensions under consideration are important, the ASGC method will typically require more points than the h-GSG method. To clarify this point further, consider the refinement of active index sets for the h-GSG and ASGC methods for a d-dimensional sub-problem for which all dimensions are

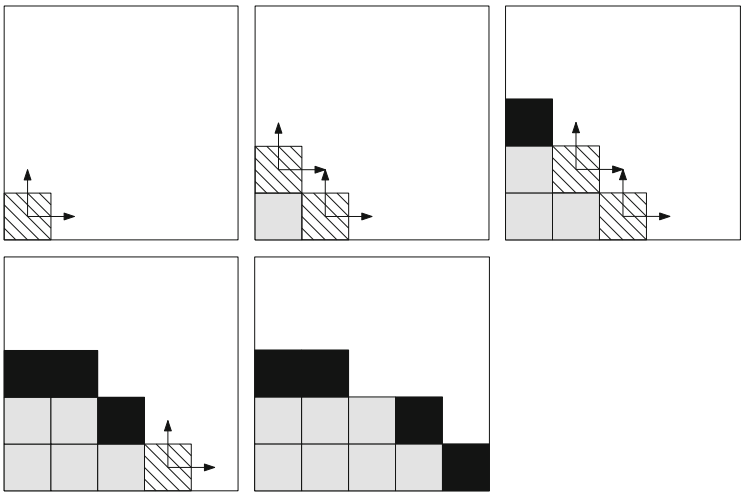


Fig. 5 Five steps of the ASGC method. *Striped boxes* indicate indices to be refined, *black boxes* grids that only contain points with $\gamma_{\mathbf{l},\mathbf{i}} < \varepsilon$, and *grey boxes* represent indices already in the sparse grid

important. This is the situation presented in Fig. 4. ASGC and h-GSG both define an active index as any index $\mathbf{l}$ with at least one point $\boldsymbol{\xi}_{\mathbf{l},\mathbf{i}}$ with an error indicator that satisfies $\gamma_{\mathbf{l},\mathbf{i}} \geq \varepsilon$. Given a particular active index $\mathbf{l}$, the h-GSG method will generate 1 to d new grid indices depending on the existence of backwards neighbours $\mathbf{l} - \mathbf{e}_j$, $j = 1, \ldots, d$ according to the admissibility criteria (9). The ASGC method, however, will always add d new indices. So given any sub-dimensional component of the ANOVA decomposition, h-GSG will typically construct a smaller index set and thus use less points.

5.2 Numerical Comparison

In this section we compare the h-GSG method with the generalised sparse grid (GSG) method [7, 9], and the ASGC-HDMR method [18] for a set of functions of varying dimension and smoothness. The GSG method can be obtained by removing the local adaptation in the h-GSG method. We note that the performance of these methods is somewhat problem dependent and the results shown below may change for different test functions.

We investigate the performance for the following three functions:

$$y_1^d(\boldsymbol{\xi}) = \exp\left(-\sum_{n=1}^{d} c_n^2(\xi_n - w_n)^2\right), \quad \boldsymbol{\xi} \in [0,1]^d \tag{11}$$

$$y_2^d(\boldsymbol{\xi}) = \exp\left(-\sum_{n=1}^{d} c_n|\xi_n - w_n|\right), \quad \boldsymbol{\xi} \in [0,1]^d \tag{12}$$

$$y_3^d(\boldsymbol{\xi}) = \begin{cases} 0 & \text{if } \xi_1 > w_1 \text{ or } \xi_2 > w_2 \\ \exp\left(\sum_{n=1}^{d} c_n \xi_n\right) & \text{otherwise} \end{cases}, \quad \boldsymbol{\xi} \in [0,1]^d \tag{13}$$

Unless otherwise stated, the coefficients $w_n = 0.5, n = 1, \ldots, d$. The choice of c_n determines the effective dimensionality of the function and is defined differently for each problem. Although smooth, the mixed derivatives of the Gaussian function y_1^d can become large and thus degrade performance if not compensated for by appropriate adaptivity. The discontinuities in functions y_2^d and y_3^d also degrade, with increasing magnitude, the efficiency of isotropic methods and subsequently can highlight the strengths and weaknesses of any interpolation method.

We will analyze convergence with respect to the following measures:

$$\varepsilon_{\ell^\infty} = \max_{k=1,\ldots,N} |f(\boldsymbol{\xi}^{(k)}) - g(\boldsymbol{\xi}^{(k)})|$$

$$\varepsilon_{\ell^2} = \left(\frac{1}{N}\sum_{k=1}^{N} |f(\boldsymbol{\xi}^{(k)}) - g(\boldsymbol{\xi}^{(k)})|^2\right)^{1/2}$$

where f and g are the true function and approximation, respectively. In all the following examples we take $N = 1{,}000$ points $\{\boldsymbol{\xi}^{(k)}\}_{k=1}^{N}$ uniformly sampled in $[0,1]^d$.

Numerous local error indicators can be used in conjunction with the ASGC-HDMR and h-GSG methods. In an attempt to provide a fair comparison we use the same local error indicator, used for local adaptation, for both methods

$$\gamma_{\mathbf{l},\mathbf{i}} = |v_{\mathbf{l},\mathbf{i}}\, w_{\mathbf{l},\mathbf{i}}|$$

The exact form of the error indicator is unimportant for this discussion. The use of the same error indicator is achievable due to the similar local adaptation procedures of the two methods. It is the way in which this local adaptation is combined with dimension adaptivity where the two methods differ.

Consider the three test functions (11)–(13) with

$$c_n = \frac{1}{2^{n+1}}, \quad n = 1, \ldots, d$$

Figure 6 shows the error of the h-GSG, ASGC-HDMR and GSG methods when applied to each of these three problems when $d = 10$. Specifically the plots depict the change in error associated with decreasing values of ε. Here we wish to compare the performances of the three types of adaptivity and accordingly restrict our attention to piecewise-linear basis functions (5). In the smooth case (Fig. 6a) the

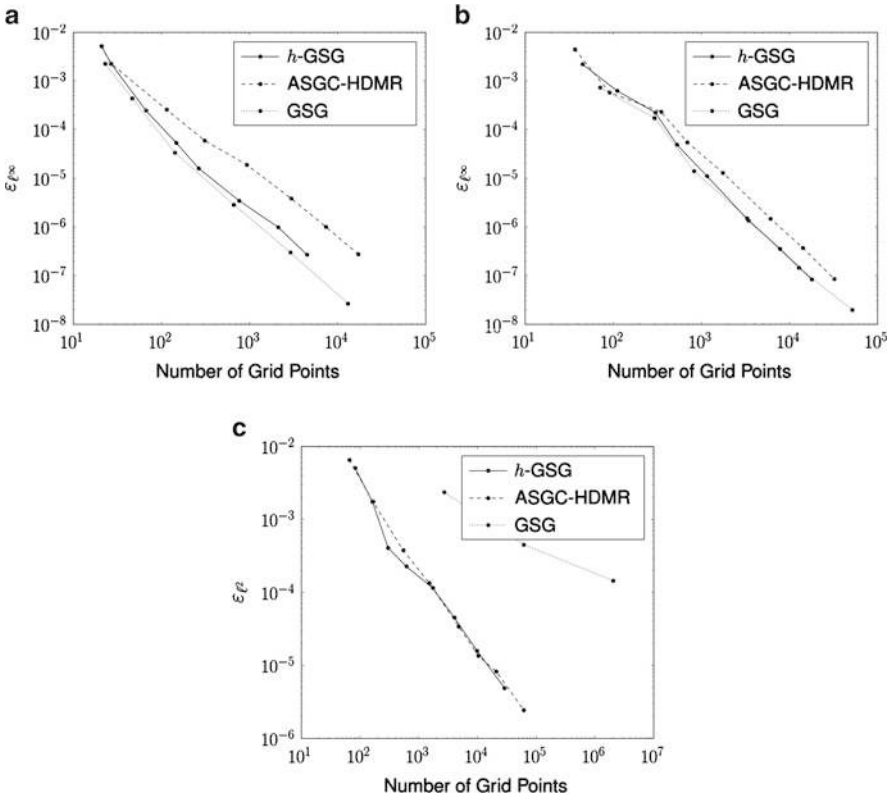


Fig. 6 Convergence rates obtained by the h-GSG method for varying basis degree $p_{\max}$ when applied to the three test functions $d = 10$. Convergence is for decreasing values of ε. Each point corresponds to decreasing values of ε. These values start at 10^{-2} and successively decrease by a factor of 10. (**a**) f_1 (**b**) f_2 (**c**) f_3

h-adaptivity of the proposed method provides no improvement on the generalised sparse grid method. It is well documented that local adaptivity provides little improvement for smooth problems, e.g. [4]. For the piecewise-continuous function the GSG method also performs comparably to the h-GSG method (see Fig. 6b). However the h-GSG method is much more efficient than its GSG counterpart when the function being approximated is discontinuous, as shown in Fig. 6c. If the GSG method determines that the approximation is poor it will add points everywhere along the dimension selected for refinement. This is efficient when the function is smooth but extremely inefficient if only a small region of the input space is contributing to the function variability. Thus for the discontinuous function, the h-GSG method is much more efficient than GSG.

The ASGC-HDMR method is the least efficient of the three methods compared, for the smooth and piecewise-continuous functions. However the ASGC-HDMR

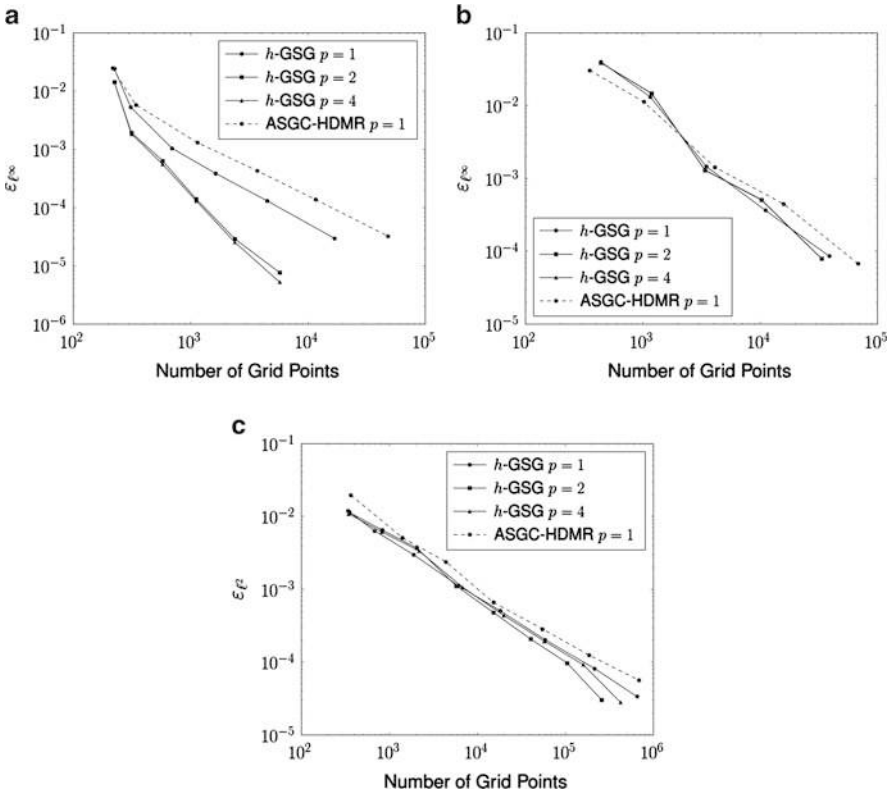


Fig. 7 Convergence rates obtained by the h-GSG and ASGC-HDMR method when applied to the three test functions $d = 100$. Convergence is for decreasing values of ε. Each point corresponds to decreasing values of ε. These values start at 10^{-2} and successively decrease by a factor of 10. (**a**) f_1 (**b**) f_2 (**c**) f_3

variant performs equally well as h-GSG when applied to the discontinuous function. In this moderate dimensional setting the accuracy of the ASGC-HDMR method is comparable to the h-GSG method. However as the dimensionality increases, the disparity between these two methods also increases. Now let

$$c_n = \exp(-\frac{35\,n}{d})$$

Figure 7 shows the error of the h-GSG and ASGC-HDMR for $d = 100$. For all three test problems the linear h-GSG method is significantly more efficient than ASGC-HDMR. The relative accuracy of the h-GSG method is increased further if quadratic or quartic basis functions are used.

Aside from the improvement in the rate of convergence the h-GSG has an additional advantage over ASGC-HDMR. The greedy nature of the proposed

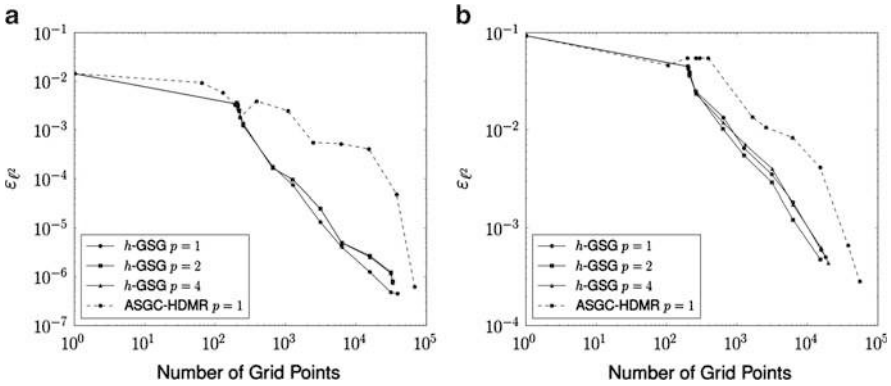


Fig. 8 Error in the h-GSG and ASGC-HDMR interpolants as the respective algorithms evolve when applied to the f_2 and f_3 with $\varepsilon = 10^{-6}$. (**a**) f_2 (**b**) f_3

algorithm means that the error in the approximation during the evolution of the sparse grid algorithm decays more quickly than its ASGC-HDMR counterpart. This is illustrated in Fig. 8. For a fixed computational budget, which may not be able to fully resolve the function being considered, the proposed algorithm will choose the best set of points that give the smallest approximation error. The results are shown for functions f_2 and f_3. The disparity between the two methods is similar for the smooth function f_1.

6 Multi-dimensional Stochastic Application

In this section we consider a model simulating flow through porous media. This model is used to predict the steady state velocity field produced by the time-invariant flow of water in a two-dimensional spatial domain. This velocity field is used to model the advection of a tracer through the domain. The time it takes the tracer to reach a given measurement site from the point of release is referred to as the breakthrough time. Unlike other physical quantities, such as pressure, the measurement of breakthrough times in the field is relatively straightforward. However approximation of breakthrough times using interpolation methods is difficult due to the discontinuous nature of the saturation front.

6.1 Model Formulation

Consider the following model which simulates flow through a porous medium, in the rectangular domain $D = [0, 1.5] \times [0, 1]$, with an underlying permeability field

$K(\mathbf{x})$ represented by a spatially varying random field. The random permeability field is represented using the Karhunen–Loève expansion (KLE). The KLE is derived from a separable exponential covariance function

$$\mathrm{Cov}(\mathbf{x}, \mathbf{y}) = \sigma^2 \exp\left(\frac{|x_1 - y_1|}{L} - \frac{|x_2 - y_2|}{L}\right)$$

where L is the correlation length and σ^2 is the variance of the random field. The resulting KLE is given by

$$X(\mathbf{x}, \boldsymbol{\xi}) = \log(K(\mathbf{x}, \boldsymbol{\xi})) = \sum_{n=1}^{d} \sqrt{\lambda_n}\phi_n(x)\xi_n, \quad \mathbf{x} \in D \tag{14}$$

Here we assume that the ranges of the random variables ξ are contained in the interval $[-1, 1]$ and $\sigma^2 = 1$. The eigenvectors ϕ_n and the corresponding eigenvalues $\lambda_n, n = 1, \ldots, d$ are obtained numerically.

In two-dimensions, the velocity field $\mathbf{v}(\mathbf{x}, \boldsymbol{\xi}) = \{v_1(\mathbf{x}, \boldsymbol{\xi}), v_2(\mathbf{x}, \boldsymbol{\xi})\}$, of an incompressible fluid, can be approximated by

$$\nabla \cdot \mathbf{v}(\mathbf{x}, \boldsymbol{\xi}) = f(\mathbf{x}), \qquad \mathbf{x} \in D, \quad \boldsymbol{\xi} \in I_{\boldsymbol{\xi}} \tag{15}$$

$$\mathbf{v}(\mathbf{x}, \boldsymbol{\xi}) = -K(\mathbf{x}, \boldsymbol{\xi}) \nabla p(\mathbf{x}, \boldsymbol{\xi}), \quad \mathbf{x} \in D, \quad \boldsymbol{\xi} \in I_{\boldsymbol{\xi}} \tag{16}$$

where $p(\mathbf{x}, \boldsymbol{\xi})$ represents the pressure field within D and the source term $f(\mathbf{x})$ models the inflow and outflow at designated well locations. Once the velocity field has been determined, the concentration of the tracer is modelled using the continuity equation

$$\frac{\partial}{\partial t} c(\mathbf{x}, \boldsymbol{\xi}) + \frac{\partial}{\partial x_1} F + \frac{\partial}{\partial x_2} G, \tag{17}$$

where $\{F, G\} = \{v_1(\mathbf{x}, \boldsymbol{\xi})c(\mathbf{x}, \boldsymbol{\xi}), v_2(\mathbf{x}, \boldsymbol{\xi})c(\mathbf{x}, \boldsymbol{\xi})\}$ are the fluxes in the x_1 and x_2 directions respectively.

We wish to model flow driven solely by the source term $f(\mathbf{x})$ and hence set

$$\mathbf{v}(\mathbf{x}) \cdot \mathbf{n} = 0, \quad \mathbf{x} \in \partial D$$

where $\mathbf{n}$ is a vector normal to the boundary ∂D. This no-flow condition prevents water from entering or leaving the domain through the boundaries.

The spatial domain D is decomposed into a rectangular mesh with 30 cells in the x_1 direction and 20 cells in the x_2 direction. The governing equations (15) and (16) are then solved using the two-point flux approximation (TPFA) finite-volume method. The concentration equation is advanced in time using a time-step controlled fourth-order Runge-Kutta algorithm. Flow is driven through this domain by pumping water in through the cell at the bottom-left corner of the domain and extracted water from the top-right corner cell.

6.2 Results

In this section we wish to illustrate the ability of the proposed h-GSG method to construct an efficient high-dimensional surrogate of the porous media flow model (15) and (16). With this aim we construct interpolants of the speed field

$$s(\mathbf{x}, \boldsymbol{\xi}) = \left(v_1(\mathbf{x}, \boldsymbol{\xi})^2 + v_2(\mathbf{x}, \boldsymbol{\xi})^2\right)^{\frac{1}{2}}$$

for three different correlation lengths $L \in \{1, 0.5, 0.3\}$.

The correlation length determines the variability of the permeability field and the effective dimensionality of the parametric space. As the correlation length decreases the decay rate of the eigenvalues λ_n decreases as does the importance of the associated variable ξ_n.

For each correlation length $L \in \{1, 0.5, 0.3\}$ we truncate terms in the KLE (14) with $\lambda_n = O(10^{-14})$. The resulting KLEs have $d = 35$, 60, and 100 terms, respectively. Setting $\varepsilon = 10^{-6}$ we use h-GSG to construct interpolants over the input space $I_{\boldsymbol{\xi}}$ for each of the 600 cells used to discretise the spatial domain D. The decoupled nature of stochastic collocation means that one could simply construct an independent sparse grid in each cell, however greater efficiency can be achieved by considering the cells simultaneously. Every time the true model is evaluated a complete time and spatial history becomes available. This information is stored at the corresponding point in the sparse grid. The decision to refine is based upon the worst case error criteria

$$r_{\mathbf{l}} = \max_{\mathbf{x}\in\mathcal{X}} r_{\mathbf{l}}(\mathbf{x}) = \max_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{i}\in B_{\mathbf{l}}} v_{\mathbf{l},\mathbf{i}}(\mathbf{x})\, w_{\mathbf{l},\mathbf{i}}$$

$$\gamma_{\mathbf{l},\mathbf{i}} = \max_{\mathbf{x}\in\mathcal{X}} \gamma_{\mathbf{l},\mathbf{i}}(\mathbf{x}) = \max_{\mathbf{x}\in\mathcal{X}} |v_{\mathbf{l},\mathbf{i}}(\mathbf{x})\, w_{\mathbf{l},\mathbf{i}}|$$

where $\mathcal{X}$ are the set of spatial cells used by the numerical method described in Sect. 6.1. Thus a point is identified for refinement if the approximation is poor in at least one of the spatial cells.

Using quadratic basis functions, the number of grid points (model evaluations) required by h-GSG to resolve the speed fields for the three correlation lengths $L \in \{1, 0.5, 0.3\}$ are 399, 2,382, and 68,788 respectively. The ε_{L_2} error in the three approximations are 3.23×10^{-6}, 8.48×10^{-6}, and 5.99×10^{-5}. Recall that for each point in the computational grid $\mathbf{x}^{(k)} \in \mathcal{X} \in D$ we must calculate the hierarchical coefficient $v_{\mathbf{l},\mathbf{i}}(\mathbf{x}^{(k)})$ for each basis function. Consequently for $L = 0.3$ the sparse grid surrogate has $41{,}272{,}800 = 30 \times 20 \times 68{,}788$ degrees of freedom. Yet despite this large degrees of freedom the h-GSG method obtains a high-degree of accuracy on a single desktop computer.

We can also use h-GSG to construct a surrogate for estimating breakthrough times at 20 sites distributed throughout the spatial domain D. After releasing the tracer at the point of inflow, the tracer is deemed to reach a site $\mathbf{x}$ when the

concentration $c(\mathbf{x}) = 0.05$. The simulation is run for 40 days. If the tracer has not reached a site the breakthrough time is set to 40. This introduces a discontinuity in the first and higher-order derivatives which necessitates the use of a h-adaptive scheme, such as the h-GSG method.

We use h-GSG, again with quadratic basis functions, to construct surrogates for $L = 1.0$ and $L = 0.5$ and as before set $\varepsilon = 10^{-6}$ and $d = 35$ and $d = 60$ for these two cases. The resulting grids contain 3,127 and 378,102 points, respectively. Approximating breakthrough times is much more difficult than approximating speed and more research is needed. However these results show that obtaining a surrogate may be possible. The degradation in performance is likely caused, at least in part, by the error indicators used to guide the local and dimension adaptivity. Further research is needed to construct problem dependent error indicators that will likely improve performance.

7 Conclusion

This paper presents a stochastic collocation method based upon a recently developed dimension and locally-adaptive sparse grid algorithm designed for interpolating high-dimensional functions with discontinuities. The underlying dimension adaptation procedure greedily identifies the model variables and variable interactions that contribute most to the variability of the model. The hierarchical surplus at each point within the sparse grid is used as an error indicator to guide local refinement, with the aim of concentrating computational effort within rapidly varying or discontinuous regions. This approach limits the number of points that are invested in 'unimportant' dimensions and regions within the high-dimensional domain. Piecewise polynomial bases are used to take advantage of any smoothness in the model input-output map. The finite support of these basis functions facilitates local adaptation which is not possible with global polynomial basis functions.

The proposed stochastic collocation method is compared against the generalised sparse grid collocation method and the Adaptive Sparse Grid Collocation High Dimensional Model Representation (ASGC-HDMR) method. When applied to low-dimensional functions $d = 10$, with a piecewise linear basis, all methods perform similarly when applied to smooth functions. However the performance of the GSG stochastic collocation method deteriorates severely when applied to discontinuous functions. The disparity in the proposed h-GSG stochastic collocation method and ASGC-HDMR increases as the number of model inputs increases. In the 10 and 100 dimensional cases tested the h-GSG is always more efficient than its ASGC-HDMR counterpart. The difference in performance between these two methods is further increased if the higher-degree basis functions of the h-GSG are used. Due to the poor performance of generalised sparse grid stochastic collocation, comparison of this method with the other two methods could not be performed in the higher-dimensional setting.

The paper is concluded by the application of the proposed method to the quantification of uncertainty in a model of flow through porous media. Two quantities of interest are considered. The first is the speed field of the fluid and the second is the time a tracer released at a point site takes to reach 20 other sites within the physical domain (referred to as breakthrough or saturation times). Uncertainty is introduced through a stochastic permeability field that is represented using a finite dimensional Karhunen–Loève expansion (KLE). In this setting the uncertain model variables (coefficients of the terms in the KLE) are governed by natural yet unknown weights. In these cases, the proposed method can utilise this implicit weighting to determine and restrict effort to the lower effective dimension of the model. When used to approximate the speed field, the proposed method, through the use of quadratic basis functions, can utilise the underlying high-regularity of the response surface to obtain extremely efficient estimates of the mean and variance of the speed field for as many as a 100 input variables. When breakthrough times are the quantity of interest the efficiency is reduced. This degradation in performance is caused, at least in part, by the error indicators used to guide the local and dimension adaptivity. Further research is needed to construct problem dependent error indicators that will likely improve performance.

Acknowledgements The authors would like to thank Jaideep Ray of Sandia National Laboratories, Livermore, CA, USA for providing us with the model discussed in Sect. 6.1.

References

1. I.M. Babuska, F. Nobile, and R. Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 45(3):1005–1034, 2007.
2. I.M. Babuska, R. Tempone, and G.E. Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825, 2004.
3. V. Barthelmann, E. Novak, and K. Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12:273–288, 2000.
4. H.-J. Bungartz. *Finite elements of higher order on sparse grids*. PhD thesis, Institut für Informatik, TU München, 1998.
5. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.
6. Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta Numerica*, 13:147, June 2004.
7. B. Ganapathysubramanian and N. Zabaras. Sparse grid collocation schemes for stochastic natural convection problems. *Journal of Computational Physics*, 225(1):652–685, 2007.
8. T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numerical Algorithms*, 18(3–4):209–232, 1998.
9. T Gerstner and M Griebel. Dimension–adaptive tensor–product quadrature. *Computing*, 71(1):65–87, 2003.
10. R.G. Ghanem and P.D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer-Verlag New York, Inc., New York, NY, USA, 1991.
11. M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.

12. M. Griebel and M. Holtz. Dimension-wise integration of high-dimensional functions with applications to finance. *Journal of Complexity*, 26(5):455–489, 2010. SI: HDA 2009.
13. M Hegland. Adaptive sparse grids. *Anziam Journal*, 44(April):335–353, 2003.
14. J.D. Jakeman, M. Eldred, and D. Xiu. Numerical approach for quantification of epistemic uncertainty. *Journal of Computational Physics*, 229(12):4648–4663, 2010.
15. J.D. Jakeman and S.G. Roberts. Local and dimension adaptive sparse grid interpolation and quadrature. *Pre-print*, 2011. arXiv:1110.0010v1 [math.NA].
16. O.M. Knio and O.P. Le Maitre. Uncertainty propagation in CFD using Polynomial Chaos decomposition. *Fluid Dynamics Research*, 38(9):616–640, 2006.
17. X. Ma and N. Zabaras. An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *Journal of Computational Physics*, 228:3084–3113, 2009.
18. X. Ma and N. Zabaras. An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations. *Journal of Computational Physics*, 229(10):3884–3915, May 2010.
19. L. Mathelin, M. Hussaini, and T. Zang. Stochastic approaches to uncertainty quantification in CFD simulations. *Numerical Algorithms*, 38(1–3):209–236, MAR 2005.
20. M.G. Morgan and M. Henrion. *Uncertainty: a Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*. Cambridge University Press, New York, 1990.
21. F. Nobile, R. Tempone, and C.G. Webster. An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2411–2442, 2008.
22. F. Nobile, R. Tempone, and C.G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2309–2345, 2008.
23. M. Tatang, W. Pan, R. Prinn, and G. McRae. An efficient method for parametric uncertainty analysis of numerical geophysical model. *Journal of Geophysical Research*, 102(D18):21925–21932, 1997.
24. O.V. Vasilyev and S. Paolucci. A fast adaptive wavelet collocation algorithm for multidimensional PDEs. *Journal of Computational Physics*, 138:16–56, 1997.
25. D. Xiu. *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010.
26. D. Xiu and J.S. Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM Journal on Scientific Computing*, 27(3):1118–1139, 2005.
27. Dongbin Xiu and George Em Karniadakis. The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, 24(2):619, 2002.

The Combination Technique for the Initial Value Problem in Linear Gyrokinetics

Christoph Kowitz, Dirk Pflüger, Frank Jenko, and Markus Hegland

Abstract The simulation of hot fusion plasmas via the five-dimensional gyrokinetic equations is computationally intensive with one reason being the curse of dimensionality. Using the sparse grid combination technique could reduce the computational effort. For the computation of the full grid solutions, the plasma turbulence code GENE is used. It is shown that the combination technique is applicable to linear gyrokinetics by retrieving combination coefficients with a least squares approach. The retrieved sparse grid solution is actually close to the full grid one. Also, combination schemes were found which provided promising results with respect to the computational effort and accuracy.

1 Introduction

A promising approach for producing energy by means of controlled nuclear fusion utilizes magnetic confinement of the fusion plasma in toroidal chambers like tokamaks or stellarators. To be able to create a self sustained fusion reaction in the hot core zone of the plasma, the heat and the particles have to be confined rather effectively. But even strong magnetic fields are unable to prevent both unacceptably

C. Kowitz (✉)
Institute for Advanced Study, Technische Universität München, Munich, Germany
e-mail: kowitz@in.tum.de

D. Pflüger
Institute for Parallel and Distributed Systems, University of Stuttgart, 70569 Stuttgart, Germany
e-mail: Dirk.Pflueger@ipvs.uni-stuttgart.de

F. Jenko
Max-Planck-Institut für Plasmaphysik, Garching, Germany

M. Hegland
Australian National University, Canberra, Australia

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_10,

high fluxes of heat and particles moving out of the confinement. Microturbulence in the plasma is leading to this so called anomalous transport, which is much larger than the originally expected transport. It is induced by microinstabilities, which appear due to gradients in the plasma temperature or density.

Different approaches exist for simulating a hot magnetized plasma. They range from single particle descriptions to continuous models like magnetohydrodynamics. For analyzing the concept of microturbulence, the gyrokinetic description of the plasma has shown its ability to reconstruct the actual physical processes in a tokamak [18]. The gyrokinetic equations describe the time-development of a distribution function in a five-dimensional phase space. They are known to provide an accurate description of plasma microturbulence. The computations are done on a strongly anisotropic dense regular Cartesian grid and thus simulations, which can take up to hundreds of thousands of CPU hours, are heavily affected by the curse of dimensionality [21].

Sparse grids [3] are thus the natural choice to solve the gyrokinetic equations: They can achieve an accuracy which is only matched by dense grids with substantially larger numbers of grid points. However, this competitiveness may be lost when the usual hierarchical basis is used to determine the sparse grid solution. In addition the hierarchical basis approach would require a total reimplementation without even knowing if sparse grids would retrieve comparable solutions. Using the combination technique [14] is thus promising, since with it, a sparse grid solution can be achieved by using existing, highly optimized solvers. It has been successfully applied for simulating turbulent flows before [13]. Using the combination technique furthermore has the property of being inherently parallel, which provides an advantage, for large scale applications.

For retrieving dense grid solutions of the gyrokinetic equations efficiently, the simulation code GENE (Gyrokinetic Electromagnetic Numerical Experiment)[1] has been chosen. It has been developed in the group of Frank Jenko at the Max-Planck-Institute for Plasma Physics (IPP) in Garching, Germany. The code is used extensively in the fusion community; it is under persistent development and users of GENE get support from its developers at IPP.

GENE has basically two main modes of action: on the one hand, a full nonlinear simulation of the plasma microturbulence in time, and on the other hand, an analysis of the linear dynamics of the system. The former is time consuming but gives detailed information of the turbulence and the resulting fluxes of heat and particles, whereas the latter gives a rather fast impression of the microinstabilities, which are driving the microturbulence of the system. Knowing the driving microinstabilities, transport models can be applied to estimate the flux of the resulting microturbulence. The linear analysis is often employed for large scale parameter scans to identify favorable regimes for nonlinear simulations. The microinstabilities can be analyzed using a time-integration of the linear equations or by an eigenvalue decomposition [17, 23].

[1] http://gene.rzg.mpg.de

Both modes of action require a large amount of computational resources and are thus done on current HPC systems [19]. Using the combination technique for gyrokinetics shall thus be aimed at reducing the computational effort for linear as well as nonlinear computations in that respective systems. For linear computations, the computational effort issues from large parameter scans with single computations with $\mathcal{O}(10^5)$ unknowns [22] requiring resources in the order of few CPU hours. Typical nonlinear simulations with their vast grid sizes of $\mathcal{O}(10^8)$ unknowns and long simulation lengths are requiring hundreds of thousands CPU hours or even more for certain physical setups [12]. Since both modes of action cannot increase the parallelism further without a significant loss in scaling efficiency, the combination technique can provide a remedy due to its additional level of parallelism.

In this paper we present results showing the applicability of the combination technique to linear gyrokinetics. First we give a quick introduction to linear gyrokinetics and discuss the nature of the simulation domain. Then we introduce the various combination techniques used. In the last section, we present some results using the combination techniques for linear initial value runs.

2 Linear Gyrokinetics

The gyrokinetic equation is derived from the so-called Vlasov equation

$$\frac{\partial f_s}{\partial t} + \mathbf{v}\frac{\partial f_s}{\partial \mathbf{x}} + \left(\frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B})\right)\frac{\partial f_s}{\partial \mathbf{v}} = \Delta(f_s)\ , \tag{1}$$

which describes the time development of the distribution function $f_s(\mathbf{x}, \mathbf{v}; t)$ of particles of species s with charge q and mass m in phase space under the influence of a magnetic field $\mathbf{B}$ and an electric field $\mathbf{E}$, which are both dependent on f via Maxwell's equations. The Vlasov equation (1) is nonlinear since $\mathbf{E}$ and $\mathbf{B}$ depend on f. We will only give a quick introduction to gyrokinetics here, since a full derivation is done comprehensively elsewhere [2,5]. Since particle collisions only have a minor influence on the dynamics of dilute and hot plasmas like the ones in a tokamak or stellarator, the collision operator $\Delta(f)$ is neglected here. In a strong external magnetic field, charged particles (either ions or electrons) are gyrating around the magnetic field lines due to the Lorentz force. They are almost completely bound to that field line in the perpendicular directions, but they can move freely parallel to it. The particles only move perpendicular to the field line due to slow drifts, which are a result of electric and magnetic fields and their gradients. These drifts are significantly slower than the thermal speed, but together with the movement parallel to the magnetic field, they determine the characteristic time scale of plasma microturbulence. Thus a simulation using the Vlasov equation would have to resolve the full gyration motion in time and would waste a lot of computational resources in a time scale not required for understanding plasma microturbulence.

In gyrokinetics, a coordinate transformation is done, reducing the full motion of a charged particle to the dynamics of the center of gyration (guiding center). The velocities $\mathbf{v}$ are then replaced by the velocity parallel to the magnetic field line $v_\parallel$ and the magnetic moment μ, which is representing the velocity of gyration around the field line. The problem is thus reduced to a five-dimensional one, since only the position of the gyration center is modeled, whereas the phase angle of the rotation drops out.

Further approximations (not described here) yield

$$\frac{\partial f_s}{\partial t} + \left(v_\parallel \mathbf{b} + (\mathbf{v}_{E\times B} + \mathbf{v}_{\nabla B} + \mathbf{v}_\mathrm{c})\right) \left(\frac{\partial f_s}{\partial \mathbf{x}} + \frac{1}{m_s v_\parallel}\left(q\bar{\mathbf{E}}_1 - \mu\nabla(B + \bar{B}_{1\parallel})\right)\frac{\partial f_s}{\partial v_\parallel}\right) = 0 \tag{2}$$

with $\mathbf{v}_{E\times B}$, $\mathbf{v}_{\nabla B}$ and $\mathbf{v}_\mathrm{c}$ being different drift motions of the charged particles, $\bar{\mathbf{E}}_1$ and $\bar{B}_{1\parallel}$ perturbations of the background fields, averaged over the radius of gyration. The quantities $\mathbf{b}$ and B represent the direction and strength of the external magnetic field respectively. Substituting f_s by g_s, which is the perturbation of f_s compared to a Maxwell background distribution, and unifying the different species into one vector g, the gyrokinetic equation can be written as

$$\frac{\partial g}{\partial t} = \mathcal{L}[g] + \mathcal{N}[g] \tag{3}$$

in terms of a linear operator $\mathcal{L}$ and a nonlinear operator $\mathcal{N}$. The nonlinearity is basically coming from the $E\times B$ drift $\mathbf{v}_{E\times B}$, which depends on the current local fields created by the distribution function. For the rest of the paper we focus on linear gyrokinetics, so that the general equation reads

$$\frac{\partial g}{\partial t} = \mathcal{L}[g]\,. \tag{4}$$

3 The Gyrokinetic Electromagnetic Numerical Experiment GENE

The simulation code GENE is a fully parallelized numerical simulation code, which is able to solve the linear and nonlinear gyrokinetic equations. Besides the core computational module, a variety of diagnostic tools exists, which can be used to draw conclusions about the physical behavior. In this section we will focus on some properties of GENE, which influence the application of the combination technique.

3.1 Fluxtube Domain

GENE solves the gyrokinetic equations in a so called fluxtube domain [1,11], which is a small area around a magnetic field line, followed for one poloidal turn, with the poloidal angle being the field line label in z direction so that the domain covers an extent $L_z = 2\pi$ in the range of $[-\pi, \pi]$. Since a tokamak is axisymmetric, this simulation domain is sufficient to simulate the behavior of a whole flux surface (the iso-surface of the magnetic flux). Due to strong anisotropy of the movement of the particles, the grid sizes and mesh widths are strongly distorted. Whereas the mesh width parallel to the magnetic field is on the order of meters, the step size perpendicular to it is on the order of a gyration radius, hence millimeters. Periodic boundary conditions are applied in these perpendicular directions (x, y), since it is a valid assumption that the plasma conditions are not changing in that local area. The x and y directions are then represented in Fourier space as k_x and k_y. This approach also allows for exact numerical derivatives in these directions, and integrals (required for averaging physical quantities over the area of gyration) can be calculated by local computations instead of a nonlocal summation. The simulation domain in this direction (L_x, L_y) has to be chosen large enough to prevent any self-correlation in the simulation. In the direction parallel to the magnetic field lines (z) periodic boundary conditions are applied as well, but here the finite size of the box in x and y has to be taken into account. The safety factor of the magnetic field lines, which is basically the number of toroidal turns per poloidal turn, is not the same for each magnetic flux surface. So there is a defined shear between field lines of different radial positions in y direction, which depends linearly on x. Thus the finite computational domain following those field-lines is distorted in y direction, and does not align with the end of another flux tube. Matching these domains is achieved by choosing the domain size L_x such that the known shear shifts the domain in y direction an integer amount of periods so that the periodicity in x direction is preserved. Due to that and the periodicity in y direction, the shear (i.e. the shift in y direction) can be expressed by an exact shift in k_x direction, which is a change of the frequencies in x direction. Besides that shift, a factor has to be multiplied onto the transformed value depending on the chosen k_y coordinate to match the signs again. The parallel boundary condition is then expressed by Görler [11]

$$g(k_x, k_y, z + L_z) = g(k'_x, k_y, z)(-1)^{Mj} \tag{5}$$

with $M = sk_y^{\min} L_x$ being integer valued, k_x and k_y being the mode numbers in the perpendicular direction and i, j their respective indices. The mode k'_x is the next connected mode in x direction and computed as $k'_x = k_x^{\min}(i + Mj)$. The placement of the points of a computational grid in the z direction thus has to allow the application of that boundary condition. The grid has to range from $-\pi + \Delta z/2$ to $\pi - \Delta z/2$ with $\Delta z = L_z/n_z$ being the grid spacing and n_z being the number of grid points. Having that, a Δz distance to the boundary neighbors is preserved to easily employ the periodic boundaries.

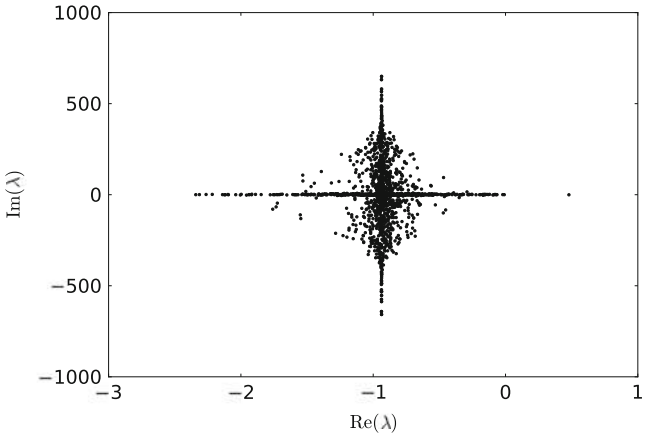


Fig. 1 The typical spectrum of the linear gyrokinetic operator. Note that most of the eigenvalues are close to the imaginary axis and have small negative real parts, which corresponds to a small damping. The eigenmodes of interest are the ones having a positive real part, since that are the only growing modes

In the $v_\parallel$ direction, Dirichlet boundaries are applied and in the μ direction, no derivatives need to be computed at the boundaries.

3.2 Modes of Linear Computations

The time integration of the nonlinear gyrokinetic equations (3) is usually done by using Runge-Kutta methods [21]. In a typical simulation, the most dominant unstable mode becomes visible first, and after a transient phase, turbulent saturation can be observed. If the nonlinearity is neglected, see (4), time integration is done until the largest growing eigenmode dominates the system. Linear computations are also used for the gyrokinetic eigenvalue problem. Solving the eigenvalue problem [17, 23] not only retrieves the dominant modes, but it also allows the computation of subdominant modes, which in turn allows the application of a more detailed transport model [6]. A typical eigenvalue spectrum of $\mathcal{L}$ is shown in Fig. 1.

Due to the structure of the linear operator $\mathcal{L}$, the linear computations of the modes in the y direction are completely decoupled [21]. This allows an exploration of dominant modes for each single k_y mode. The five dimensional problem is then a four dimensional one and can thus be a good point to start an investigation of the combination technique.

4 The Sparse Grid Combination Technique

The basic idea of the sparse grid combination technique is to retrieve a sparse grid approximation $a^{(c)}$ using a combination of smaller full grid solutions $a_i(\mathbf{x})$

$$a^{(c)}(\mathbf{x}) = \sum_i c_i a_i(\mathbf{x}) \tag{6}$$

with appropriate coefficients c_i. Since the required full grids are much smaller in size, these respective solutions can be computed quite fast and because their computation is independent it can be done completely in parallel.

4.1 Classical Scheme

In the classical sparse grid combination technique, a high resolution sparse grid solution $a^{(c)}$ is retrieved by

$$a^{(c)}(\mathbf{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\sum_i l_i = n+(d-1)-q} a_{\mathbf{l}}(\mathbf{x}) \,, \tag{7}$$

having $a_{\mathbf{l}}$ as the full grid solutions on a grid with $2^{l_i}+1$ points in ith of d dimensions and $l > 0$ [9] and n being the maximum level of resolution. The grids with the largest number of grid points, these having $\sum_i l_i = n + (d-1)$, form a hyperplane in the space of the level indices.

This hyperplane can also be shifted to use less grids of higher resolution by a shift parameter s to get the combination as:

$$a^{(c)}(\mathbf{x}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\sum_i l_i = n+(d-1)-q+s; l_i \le n_i - q} a_{\mathbf{l}}(\mathbf{x}) \,. \tag{8}$$

4.2 Dimension Adaptive Combination Schemes

As the linear gyrokinetic problem on a fluxtube domain is strongly anisotropic with respect to the number of grid points in each dimension, a dimension adaptive combination technique has to be used. Let $\mathcal{I}_{\mathbf{n}}$ be an index set with

$$\mathcal{I}_{\mathbf{n}} \subset \{\mathbf{l} \in N^d\} \,, \tag{9}$$

with each element l_j corresponding to $2^{l_j} + 1$ grid points in dimension j. $\mathbf{n}$ is the least upper bound (componentwise) of all elements in $\mathcal{I}_\mathbf{n}$. An index set is admissible if for each member $\mathbf{l} \in \mathcal{I}_\mathbf{n}$ the following holds [10]:

$$\mathbf{l} - \mathbf{e}_j \in \mathcal{I}_\mathbf{n} \quad \text{for} \quad 1 \leq j \leq d, \quad l_j > l_j^{\min}, \tag{10}$$

with $l_j^{\min}$ representing the minimum level of resolution in the jth direction, which can be chosen a priori.

Having this index set allows the construction of a sparse grid solution of the form

$$a_{\mathcal{I}_\mathbf{n}}^{(c)}(\mathbf{x}) = \sum_{\mathbf{l} \in \mathcal{I}_\mathbf{n}} \underbrace{\left(\sum_{\mathbf{z}=(0,\ldots,0)}^{(1,\ldots,1)} (-1)^{\sum_i z_i} \chi^{\mathcal{I}_\mathbf{n}}(\mathbf{l} + \mathbf{z}) \right)}_{c_\mathbf{l}} a_\mathbf{l}(\mathbf{x}) \tag{11}$$

where $\chi^{\mathcal{I}_\mathbf{n}}(\mathbf{l})$ is the characteristic function which is equal to one if $\mathbf{l}$ is in the index set and zero otherwise [10]. The index set $\mathcal{I}_\mathbf{n}$ is chosen a priori and its shape does impact on the quality of the approximation. Note that there is an algorithm which is adaptively updating the index set towards a more accurate solution [7] using the respective update formulas [15].

We use variants of index sets, which are only slightly different from the classical combination technique. One approach is to extend the index set of the classical combination technique in certain dimensions with a constant shift $\Delta\mathbf{l}$ to achieve a dimension adapted index set $\mathcal{I}_\mathbf{n} + \Delta\mathbf{l}$. Note that all minimal levels $\mathbf{l}^{\min}$ are thus also shifted accordingly. The number of full grids used for combination is the same as for the classical combination. However, the full grids used for the combination do have an increased resolution in some dimensions.

Adjusting the index set of the classical combination to a dimension adaptive setting can also be achieved by tilting the hyperplane, on which the elements with the highest resolution are lying [10]. That can be done by a stretching of the index set $\mathcal{I}_\mathbf{n}$ into the dimension which should be refined. Then

$$\sum_{j=0}^{d-1} \frac{l'_j - 1}{n'_j - 1} \leq 1 \tag{12}$$

is valid for all $\mathbf{l}' \in \mathcal{I}_{\mathbf{n}'}$, where $\mathbf{n}'$ can now be an arbitrary vector with all $n'_i > 0$. The overall number of full grids in the new index set $\mathcal{I}_{\mathbf{n}'}$ will then change compared to the classical combination, but the resolution of the combined solution is dimensionally adapted.

Sometimes the solution of a problem on a full grid with low resolution might introduce unacceptably large errors into the combined solution. It might thus be advantageous to exclude grids having an insufficient resolution in a certain

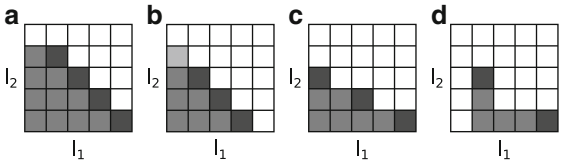


Fig. 2 The index sets of various combination techniques for anisotropic grids. The axis represent the respective dimensions and the active index set (*filled squares*) and its edge indices (*dark squares*). **(a)** A shift of the index in the first dimension, i.e. a doubling of the resolution in this dimension. **(b)** Cutoff of a single strongly anisotropic full grid (*light gray*). The resolution of the combined solution is reduced in the second dimension. **(c)** Tilting the hyperplane of the index set for dimension adaptivity. **(d)** The two-scale scheme with its strongly reduced index set

dimension from the index set $\mathcal{I}_{\mathbf{n}}$. That truncation step is increasing the minimal level $l^{\min}$ in this dimension.

It is also possible, to simply cut off certain full grids from the combination scheme. In contrast to the previously explained truncation, that does not change the minimal level of the scheme, but it rather decreases the maximal level n_i in one dimension.

All of the mentioned approaches can also be combined to achieve a more suitable index set $\mathcal{I}_{\mathbf{n}}$ for combination.

A completely different approach of choosing the index set is the so-called two-scale scheme. The index set is strongly degenerated here, since it is only employing the strongly anisotropic grids for combination:

$$a^{(c)} = -a_{\mathbf{l}^{\min}} + \sum_{j=0}^{d-1} a_{\mathbf{k}_j} \quad \text{with} \quad \mathbf{k}_j = \mathbf{l}^{\min} + \mathbf{e}_j (n_j - l_j^{\min}). \tag{13}$$

The vector $\mathbf{l}^{\min}$ contains the minimum levels in all dimensions. The advantage lies in the fact that less grids with higher resolution each are required, since the $l_j^{\min}$ are typically higher than for the classical combination [20] for reaching an equivalent accuracy.

These variants of the combination technique are illustrated in Fig. 2.

4.3 Optimizing the Combination Coefficients

For any index set one can define a set of combination coefficients which minimizes the approximation error. This choice has been called Opticom [16]. For the solution of elliptic PDEs the error norm chosen is the energy norm. For hyperbolic problems (like the gyrokinetic equations), such an approach does currently not exist. In principle, one can get optimal combination coefficients by minimizing the functional

$$J(\mathbf{c}) = \left\| a^{(f)} - a^{(c)} \right\|^2 = \left\| a^{(f)} - \sum_i c_i a_i \right\|^2 \tag{14}$$

with $a^{(f)}$ being a full grid solution and $a^{(c)}$ being the combined sparse grid solution for the same resolution. Since computing $a^{(f)}$ is not feasible in general, this minimization is only done here to investigate how well the combination technique could perform in an optimal case. The least squares minimization of (14) leads to the following set of normal equations:

$$\mathsf{A}\mathbf{c}_{\mathrm{LS}} = \mathbf{b} \tag{15}$$

with

$$A_{ij} = (a_i, a_j) \quad \text{and} \quad b_i = (a^{(f)}, a_i) \tag{16}$$

where the scalar product $(.,.)$ is computed in the space of the full grid solution $a^{(f)}$. Any projection of full grids of lower resolution into that space is done via linear interpolation.

5 Results

Here we would like to present some preliminary results using the combination technique for linear gyrokinetic computations. We first focus on the time-integration of the linear problem, since the combination of the eigenvectors and eigenvalues requires a different handling [8].

5.1 Alignment of the Full Grids

Using the sparse grid combination technique with GENE requires some adjustments. First, the solutions provided by GENE are complex. But we could not observe any consequences for the classical combination technique, since it is purely additive and it uses real combination coefficients only. We do anticipate, however, that for Opticom we might have to consider complex coefficients.

The layout of the computational domain in GENE requires some special treatment in order to apply the combination technique. The k_x and k_y are coordinates in Fourier space, so that any combination approximation can be implemented in a straightforward manner. The modes with the respective index from each full grid solution are combined and no interpolation is required. Full grids with a higher resolution only append additional grid points at the vicinity of the domain. Any modes not occurring in the coarser grid give a zero contribution to the combination, since that grid does not contain any high frequency information.

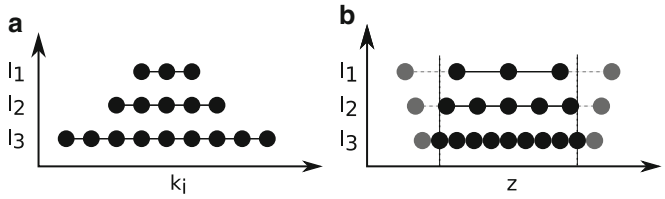


Fig. 3 The alignment of the grids for different phase space directions. **(a)** Alignment in the dimensions in Fourier space (x, y), where grid points of the finer grid are appended on the outer edges of the domain. **(b)** Alignment in z direction where the imaginary boundary points (*gray*) are added to the full grid to allow a combination for the finest grid

Unfortunately the combination in z direction is not as easy. Due to the structure of the computational grid in GENE, the domain size in z is varying with the level of resolution, which leads to non compatible grid spacings Δz for different grid sizes. Thus the grid points do not match so that linear interpolation is required for combination. A combination of the values on the grid points at the domains edge is not possible since there is no data from grids with lower resolution and thus nearly no information at all. To circumvent that lack of data, boundary points are attached to the domain, which are computed according to (5). And using interpolation, a combination for the previous outer grid points can be realized as shown in Fig. 3.

Additional to these restrictions, one has to take into account the minimal sizes of the grid, which is available in GENE. In most directions the minimum level is two thus we have to have at least five grid points.

5.2 Classical Combination Technique

Taking a simple parameter set (see Appendix), the performance of the combination technique is studied for linear initial value problems. The chosen parameter set allows the simulation of a basic instability driven by the gradient of the ion temperature (ITG-instability [4]). This problem is a basic test case of GENE and is thus used for testing the applicability of the combination technique.

The chosen grid size for comparing the combined solution with the full grid solution was $33 \times 1 \times 31 \times 33 \times 33 \times 1$ for the directions $x, y, z, v_\parallel, \mu$ and n_s respectively, with n_s being the number of simulated species. Note that the physical relevant grid size of that problem is smaller and more anisotropic, but that topic is discussed in the following section. In z direction two grid points less were required, since for a combination artificial boundaries are added. First we use the classical combination technique (7) with coarsened full grids in all directions. Then we apply the regular combination technique again, but with one dimension fixed to the maximum grid size, so that the combination is only done in three dimensions.

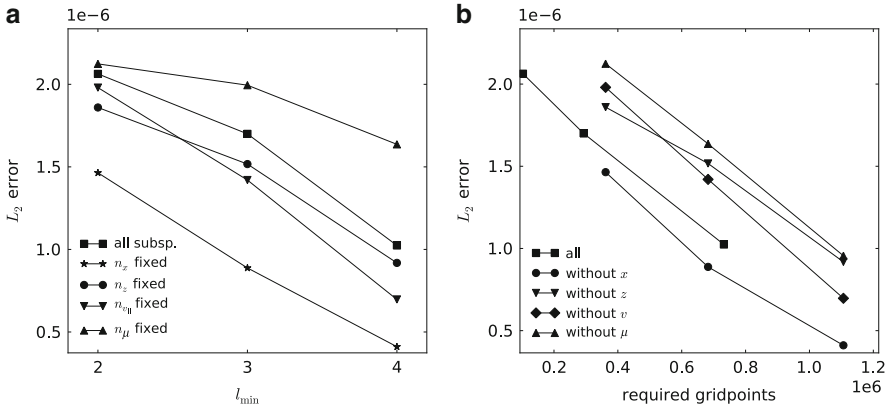


Fig. 4 The influence of the dimensions used for the regular combination technique. **(a)** The development of the normalized L_2 error depending on the minimal level of full grids used in the combination technique with the regular combination coefficients. **(b)** The error of the solutions in dependency of the estimated effort, which is here just denoted by the number of the overall amount of unknowns $\sum_i N_i$. Fixing the grid size in x direction thus seems to give the best results for moderate and high number of grid points

An interesting behavior emerges, showing that fixing the size of the grid in x direction gives much better results than fixing the other dimensions, see Fig. 4.

The error of the combined solution has an interesting structure. The highest positive and negative k_x modes of the combined solution are not interpolated with other grids, since the combination technique only computes one very anisotropic grid for this part of the mode spectrum. And these high frequency informations are directly used without any correction by other full grid solutions. But the result of that strongly anisotropic full grid might not be close to the physically correct solution, so that large portions of the error in the combined solution come from that anisotropic full grid. For combinations with a shifted hyperplane, i.e. an $s > 0$ in (8), the situation gets worse. In this case one combines fewer grids and thus the influence of each component is higher. This effect can be seen in Fig. 5. To counter this effect we cut off the highly anisotropic grid from the combination scheme like shown in Fig. 2b. The combined solution of the highest frequencies in x direction will then be computed by an actual combination of at least a few other full grids. In our case this approach is justified by the observations, which can be seen in Fig. 5.

Comparing the result to the combination with the coefficient $\mathbf{c}_{\mathrm{LS}}$ from solving the least squares problem (14) showed only small deviations from the results shown in Fig. 4. But the values of the coefficients changed more than expected (see Fig. 6), even though they do not seem to have a lot of impact on the quality of the solution. The optimized combination coefficients also show, that the influence of the coarse grids is reduced to achieve a better combination result. That might be necessary due to the fact that the coarse approximation does not yet carry the important physical behavior so that some error is introduced here.

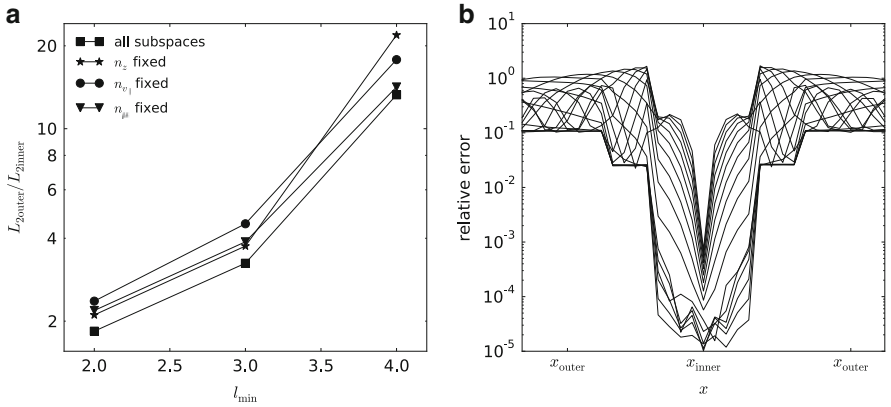


Fig. 5 The errors in dependence of the frequencies in the x coordinate. **(a)** The ratio of the L_2 error for the high frequencies and the low frequencies. For higher minimal levels the introduced combination error is even shifted towards the high frequencies. **(b)** The relative errors of the combined solution for an ensemble of different velocities $v_\parallel$ (*solid lines*) is shown for fixed z and μ. At higher frequencies (outer parts of the x coordinate), the error of the combined solution is much larger compared to the error for the lower frequencies

5.3 Anisotropic Grids

The typical grid size of the chosen ITG test problem for physical relevant linear computations is rather anisotropic and has $5 \times 1 \times 15 \times 33 \times 9 \times 1$ grid points in $x, y, z, v_\parallel, \mu$ and n_s direction respectively.

A first step to apply the combination technique on this test problem would be to just restrict the grid size in x direction to level two and apply the shifted combination scheme. Together with that, we also employed the scheme of the tilted hyperplane for a wider analysis of different combinations.

Applying this for the anisotropic test problem, we can observe that the combined solution with the least squares optimized coefficients is about twice as accurate as with the regular coefficients. So an a priori optimization of the coefficients would have a significant impact here.

Of course a cutoff of unwanted full grids is also possible. For that, the used index set is cut off in the x direction if l_x is reaching a certain level (two in our case). By that the number of full grids, which are used to combine the higher frequencies in x, is increased and the cascade of the error in higher frequencies is reduced.

In order to establish the computational complexity of the combination technique, we need to know the complexities of the component grid solutions. These complexities very much depend on the nature of the underlying problem and the methods used to solve them. Here we consider two scenarios. In the first scenario we assume

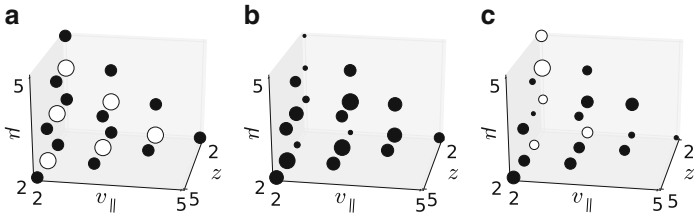


Fig. 6 The combination coefficients for the regular combination technique with a shift of $\Delta\mathbf{l} = (2, 2, 2)$ in $z, v_\parallel$ and μ direction and no combination in x direction. The size of a *circle* represents the absolute value of the combination coefficient for the full grid with the respective resolutions in $z, v_\parallel$ and μ. **(a)** The classical positive (*black*) and negative (*white*) combination coefficients. **(b)** The absolute value of the complex and least squares optimized combination coefficients. Note that the grids having a minimal resolution in $v_\parallel$ and z direction do have a rather small combination coefficient and thus have nearly no influence on the least squares optimized combined solution. **(c)** The difference between the absolute values of the classical combination coefficients and the least squares optimized ones. The small resolution grids have decreased coefficients (*white*) and the ones with higher resolutions increased combination coefficients (*black*) with the size of the *circles* representing the amount of change

that the complexity scales linearly with $\mathcal{O}(N)$ in the number of grid points N. In the second scenario, we assume that the complexity depends cubicly with $\mathcal{O}(N^3)$ on the number of grid points.

We expect that in scenario one combinations of smaller number of larger grids are favorite, whereas in scenario two the combination of larger numbers of smaller grids is favorite. This however assumes that the combination itself does not impact the complexity.

We have determined the complexities for both scenarios and for all the combination schemes considered in the previous section. The results with their respective L_2 errors are displayed in Fig. 7.

From Fig. 7 one can see that the two-scale scheme is optimal for the first scenario, which confirms our expectation that fewer grids with larger resolutions are favored.

For the other scenario, schemes using a larger number of full grids with smaller resolution provided better results than the two-scale scheme. If only a rather rough approximation is required, the more regular combination technique with a cutoff in x direction provided the most favorable results. For more accurate approximations, the scheme of a shifted and tilted hyperplane provided good results but the effort was of course higher than for the cutoff scheme. The highest accuracies can only be achieved by high minimal levels $\mathbf{l}^{\min}$, which can in turn only be realized by the two-scale scheme, since the possible combinations are narrowed down to just that one possible scheme.

The supposed optimal combination strategies can be found in Table 1 with their respective L_2 errors.

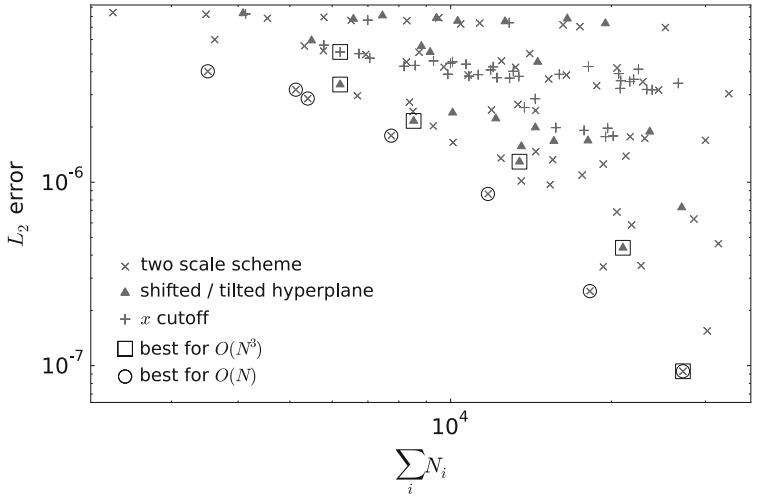


Fig. 7 The accuracy of the combination schemes in dependence of the overall amount of invested grid points. Besides the difference of the used schemes also various minimal levels $\mathbf{l}^{\min}$ are used to create solutions of different quality. Assuming a linear complexity of the computational costs of an evaluation of a full grid, the two-scale scheme gives the most favorable results in terms of accuracy per invested grid point (*circles*). If the scenario assuming cubic complexity of a single full grid evaluation is regarded, other strategies would give more favorable results (*squares*), since then the combination of more grids with smaller resolution is favored

Table 1 The favorable combination techniques with the minimum levels $\mathbf{l}^{\min}$ in $x, z, v_{\|}$ and μ direction. The two-scale approach provided the combined solution most efficiently if the effort of computing a full grid solution has a numerical complexity of $\mathcal{O}(N)$. If in the other case $\mathcal{O}(N^3)$ is assumed, a more regular combination with a shifted and tilted hyperplane worked better

	Linear complexity		
Strategy	$\mathbf{l}^{\min}$	$\sum_i N_i$	L_2 error
Two-scale	$(2,2,2,1)$	3,510	4.0×10^{-6}
Two-scale	$(2,2,2,2)$	5,400	2.8×10^{-6}
Two-scale	$(2,2,3,2)$	7,740	1.8×10^{-6}
Two-scale	$(2,2,3,3)$	11,745	8.6×10^{-7}
Two-scale	$(2,2,4,3)$	18,225	2.6×10^{-7}
Two-scale	$(2,3,4,3)$	27,225	9.3×10^{-8}
	Cubic complexity		
Strategy	$\mathbf{l}^{\min}$	$\sum_i (N_i)^3$	L_2 error
Cutoff	$(1,2,2,1)$	2.1×10^{9}	5.1×10^{-6}
Shifted/tilted hyperplane	$(2,2,2,1)$	6.4×10^{9}	3.4×10^{-6}
Shifted/tilted hyperplane	$(2,2,2,2)$	2.7×10^{10}	2.2×10^{-6}
Shifted/tilted hyperplane	$(2,2,2,3)$	1.6×10^{11}	1.3×10^{-6}
Shifted/tilted hyperplane	$(2,2,3,3)$	5.0×10^{11}	4.4×10^{-7}
Two-scale	$(2,3,4,3)$	2.8×10^{12}	9.3×10^{-8}

6 Summary

The sparse grid combination technique has been applied to linear initial value problems using the gyrokinetic plasma turbulence code GENE. The retrieved full grid solutions need to be aligned so that they can handle the fluxtube domain. An initial study of the regular combination technique on isotropic grids revealed a cascade like structure of the error in the x direction. This is a direct result of the combination of the data in Fourier space. Best results were obtained when the grid size k_x was fixed. Since the test problem considered in this paper only requires a few grid points in x, this behavior does not have any drawbacks.

A different approach was chosen for the combination of anisotropic grids. In addition to a dimension adaptive active set, a cutoff in the x direction was tested to remove the largest errors in the high frequencies. Furthermore the two-scale scheme was employed, in order to reduce the number of full grid solutions. The latter approach seemed to provide best results if the evaluation of a single full grid solution scales linearly with the number of unknowns in the grid. If the effort to compute a single full grid solution is much higher, the other dimension adapted schemes can provide a more efficient solution. A special instance of this uses an index set, where the hyperplane containing the highest resolution grid is shifted and tilted to adjust to the anisotropic nature of the problem. The cutoff of the active set is only favorable for lower approximation qualities and a high effort for each full grid computation.

For analysis purposes the least squares method was used, in order to see how full grids have to be combined to retrieve an optimal solution. In some cases the accuracy of the least squares optimized combined solution could be doubled compared to the classical coefficients. Thus optimizing the combination coefficients a priori is a promising direction for further studies.

It was shown in Sect. 5 that the combination technique is applicable to the linear gyrokinetic equations. The time integration of the initial value problems is usually done until the most unstable mode is dominating the system. But with GENE, a direct computation of the dominant eigenvalues is possible and even favorable so that a computation of eigenmodes and eigenvalues with the combination technique will be part of future work. However, the evaluation of eigenvalues with the combination technique will require more effort than simply combining solutions for multiple regular grids. This is because the scaling of the found eigenvectors is rather difficult [8].

The combination of results of nonlinear computations will be part of future studies. It is thought that the combination technique will lead to a higher gain in this case. The grids are typically larger and actually five-dimensional. Conclusions made in this paper might hence be also applicable for nonlinear initial value runs in GENE.

Appendix: Parameter File

This is the parameter file used for the linear gyrokinetic computations with GENE in Sect. 5. It is taken out of the standard test cases of GENE and the instability simulated is driven by the gradient of the ion temperature [4].

```
&parallelization

&box
n_spec =    1

kymin =   0.3000
lv    =     3.00
lw    =     9.00
adapt_lx =   T
ky0_ind =   1
/

&general
nonlinear =   F
comp_type = 'IV'
timescheme = 'RK4'
dt_max     = 0.002
timelim    = 64500
ntimesteps = 100
omega_prec = 0.000001E-03
beta       = 0.0000000
debye2     = 0.0000000
collision_op = 'none'
init_cond = 'alm'
hyp_z =      0.2500
hyp_v =      0.2000
/

&geometry
magn_geometry = 's_alpha'
shat      =    0.7960
trpeps    =    0.18000000
major_R   =    1.0000000
q0        =    1.4000000
/

&species
name   = 'ions'
omn    =    2.2200000
omt    =    6.9200000
mass   =    1.0000000
temp   =    1.0000000
dens   =    1.0000000
charge =  1
/
```

References

1. Beer, M., Cowley, S., Hammett, G.: Field-aligned coordinates for nonlinear simulations of tokamak turbulence. Physics of Plasmas **2**(7), 2687 (1995)
2. Brizard, A., Hahm, T.: Foundations of nonlinear gyrokinetic theory. Reviews of modern physics **79**(2), 421 (2007)
3. Bungartz, H.J., Griebel, M.: Sparse grids. Acta Numerica **13**, 147–269 (2004)
4. Cohen, B., Williams, T., Dimits, A., Byers, J.: Gyrokinetic simulations of $E \times B$ velocity-shear effects on ion-temperature-gradient modes. Physics of Fluids B: Plasma Physics **5**, 2967 (1993)
5. Dannert, T.: Gyrokinetische Simulation von Plasmaturbulenz mit gefangenen Teilchen und elektromagnetischen Effektien. Ph.D. thesis, Technische Universität München (2005)
6. Dannert, T., Jenko, F.: Gyrokinetic simulation of collisionless trapped-electron mode turbulence. Physics of Plasmas **12**, 072,309 (2005)
7. Garcke, J.: A dimension adaptive sparse grid combination technique for machine learning. In: W. Read, J.W. Larson, A.J. Roberts (eds.) Proceedings of the 13th Biennial Computational Techniques and Applications Conference, CTAC-2006, *ANZIAM J.*, vol. 48, pp. C725–C740 (2007)

8. Garcke, J.: An optimised sparse grid combination technique for eigenproblems. In: Proceedings of ICIAM 2007, *PAMM*, vol. 7, pp. 1022,301–1022,302 (2008)
9. Garcke, J., Griebel, M.: On the computation of the eigenproblems of hydrogen and helium in strong magnetic and electric fields with the sparse grid combination technique. Journal of Computational Physics **165**(2), 694–716 (2000)
10. Garcke, J., Griebel, M.: Classification with sparse grids using simplicial basis functions. Intelligent Data Analysis **6**(6), 483–502 (2002)
11. Görler, T.: Multiscale effects in plasma microturbulence. Ph.D. thesis, Universität Ulm (2009)
12. Görler, T., Lapillonne, X., Brunner, S., Dannert, T., Jenko, F., Merz, F., Told, D.: The global version of the gyrokinetic turbulence code GENE. Journal of Computational Physics **230**(18), 7053–7071 (2011)
13. Griebel, M., Huber, W.: Turbulence simulation on sparse grids using the combination method. In: N. Satofuka, J. Periaux, A. Ecer (eds.) Parallel Computational Fluid Dynamics, New Algorithms and Applications, pp. 75–84. North-Holland, Elsevier (1995)
14. Griebel, M., Schneider, M., Zenger, C.: A combination technique for the solution of sparse grid problems. In: P. de Groen, R. Beauwens (eds.) Iterative Methods in Linear Algebra, pp. 263–281. IMACS, Elsevier, North Holland (1992). Also as SFB Bericht, 342/19/90 A, Institut für Informatik, TU München, 1990
15. Hegland, M.: Adaptive sparse grids. In: K. Burrage, R.B. Sidje (eds.) Proc. of 10th Computational Techniques and Applications Conference CTAC-2001, vol. 44, pp. C335–C353 (2003)
16. Hegland, M., Garcke, J., Challis, V.: The combination technique and some generalisations. Linear Algebra and its Applications **420**(2–3), 249–275 (2007)
17. Kammerer, M., Merz, F., Jenko, F.: Exceptional points in linear gyrokinetics. Physics of Plasmas **15**, 052,102 (2008)
18. Lapillonne, X., Brunner, S., Sauter, O., Villard, L., Fable, E., Görler, T., Jenko, F., Merz, F.: Non-linear gyrokinetic simulations of microturbulence in tcv electron internal transport barriers. Plasma Physics and Controlled Fusion **53**, 054,011 (2011)
19. Lederer, H., Tisma, R., Hatzky, R., Bottino, A., Jenko, F.: Application enabling in DEISA: Petascaling of plasma turbulence codes. Parallel Computing: Architectures, Algorithms and Applications **38**, 713–720 (2008)
20. Liu, F., Zhou, A.: Two-scale finite element discretizations for partial differential equations. J. Comput. Math **24**(3), 373–392 (2006)
21. Merz, F.: Gyrokinetic simulation of multimode plasma turbulence. Ph.D. thesis, Universität Münster (2009)
22. Merz, F., Kowitz, C., Romero, E., Roman, J., Jenko, F.: Multi-dimensional gyrokinetic parameter studies based on eigenvalue computations. Computer Physics Communications **1**, 1–9 (2011)
23. Roman, J.E., Kammerer, M., Merz, F., Jenko, F.: Fast eigenvalue calculations in a massively parallel plasma turbulence code. Parallel Computing **36**(5–6), 339–358 (2010). Parallel Matrix Algorithms and Applications

Model Reduction with the Reduced Basis Method and Sparse Grids

Benjamin Peherstorfer, Stefan Zimmer, and Hans-Joachim Bungartz

Abstract The reduced basis (RB) method has become increasingly popular for problems where PDEs have to be solved for varying parameters $\mu \in \mathcal{D}$ in order to evaluate a parameter-dependent output function $s : \mathcal{D} \to \mathbb{R}$. The idea of the RB method is to compute the solution of the PDE for varying parameters in a problem-specific low-dimensional subspace X_N of the high-dimensional finite element space $X^{\mathcal{N}}$. We will discuss how sparse grids can be employed within the RB method or to circumvent the RB method altogether. One drawback of the RB method is that the solvers of the governing equations have to be modified and tailored to the reduced basis. This is a severe limitation of the RB method. Our approach interpolates the output function s on a sparse grid. Thus, we compute the respond to a new parameter $\mu \in \mathcal{D}$ with a simple function evaluation. No modification or in-depth knowledge of the governing equations and its solver are necessary. We present numerical examples to show that we obtain not only competitive results with the interpolation on sparse grids but that we can even be better than the RB approximation if we are only interested in a rough but very fast approximation.

1 Introduction

The reduced basis (RB) method can be employed to rapidly compute outputs depending on the solution of parametrized PDEs. More precisely, such problems are input-output relationships which means that for the evaluation of the *output of interest* a solution of a parametrized PDE is needed. The idea of the reduced

B. Peherstorfer (✉) · H.-J. Bungartz
Technische Universität München, Boltzmannstr. 3, 85748, Garching, Germany
e-mail: pehersto@in.tum.de

S. Zimmer
Universität Stuttgart, Universitätsstr. 38, 70569, Stuttgart, Germany

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_11,

basis method is to compute the solution of the PDE for varying parameters in a problem-specific low-dimensional subspace X_N of the high-dimensional finite element space $X^{\mathcal{N}}$. If done properly, the accomplished reduction of complexity permits a rapid evaluation of the output of interest, and therefore the method can be employed within *real-time* and *many-query* contexts. In many cases, the method has proven to be very efficient [11].

The method as presented in [11] is built on two "key opportunities". First, the computational procedure is divided into an *Offline* and an *Online* stage. Second, this method works well, if the parametrically induced manifold is smooth and low-dimensional and has a simple hidden structure which can be learned during the Offline stage. The Offline stage can be seen as a pre-processing of the Online stage. During the first stage, a few *"truth" solutions* of the problem are computed in the finite element space and a new space, the *reduced basis space*, is constructed based on a new basis set which contains these truth solutions. Then, in the Online stage the reduced basis approximation of the output of interest for a requested parameter is computed by using only the new ad-hoc space. The two application areas we are interested in, real-time and many-query, accept heavy Offline computations but need very fast evaluations of the output of interest during the Online stage. Therefore, the reduced basis method tries to shift as much computational complexity as possible into the Offline stage and to decouple the evaluation during the Online stage from the discretization used in the Offline stage.

In this paper, we start with a brief discussion on how sparse grids can be applied to the reduced basis method. Within the reduced basis framework we have two domains: the spatial domain $\Omega \subset \mathbb{R}^d$ and the parameter domain $\mathcal{D} \subset \mathbb{R}^P$. Sparse grids will be used in the spatial domain $\Omega \subset \mathbb{R}^d$ with the finite element method in order to reduce Offline costs. Note that here the dimension of the sparse grid is determined by the dimension d corresponding to the spatial domain Ω. We will then present procedures which use sparse grids in the parameter domain $\mathcal{D} \subset \mathbb{R}^P$ for the construction of reduced basis spaces. Typically, the dimension P is between 3 and 10, that is, the parameter domain $\mathcal{D}$ is moderate dimensional and therefore suited for sparse grids.

Second, in the main part of the paper, in Sect. 5, we will interpolate the output of interest directly and so circumvent the reduced basis method altogether. With this approach, we can overcome the drawback of the reduced basis method [11] that a solver of the governing equations with respect to the reduced basis is required. For a complex system of equations, where the development and implementation of the solver for the truth solutions is a highly challenging task on its own, the same effort would be necessary to develop a solver with respect to the reduced basis. More often than not, this is infeasible and, thus, a severe limitation of the reduced basis method [11]. In contrast, our approach based on sparse grid interpolation requires just a few truth solutions to construct the interpolant. Then, we can respond to a new parameter $\mu \in \mathcal{D}$ by simply evaluating the sparse grid interpolant at μ, independently of the underlying governing equations. We present results comparing the accuracy of the reduced basis and the sparse grid approximation with respect to the Online costs.

2 Evaluation Costs of Sparse Grid Functions

We do not give an introduction to sparse grids but refer to [3]. However, we will discuss the complexity of a sparse grid function evaluation.

Let $u^n \in V_n^{(1)}$ be a function in the sparse grid space $V_n^{(1)}$ of level n and dimension d. We can represent u^n as linear combination of the hierarchical basis functions $\{\phi_1, \ldots, \phi_M\}$ with the hierarchical coefficients $\{\alpha_1, \ldots, \alpha_M\}$, i.e.,

$$u^n(x) = \sum_{i=1}^{M} \alpha_i \phi_i(x). \tag{1}$$

This sum has to be computed, to evaluate u^n at point $x \in [0,1]^d$. Clearly, this can be done in $\mathcal{O}(M)$ operations. However, this is not a lower bound.

Consider the sum in (1) again. We can sort the basis functions by their hierarchical increment and obtain

$$u^n(x) = \sum_{|\boldsymbol{l}|_1 \leq n+d-1} \sum_{\boldsymbol{i} \in I_{\boldsymbol{l}}} \alpha_{\boldsymbol{l},\boldsymbol{i}} \phi_{\boldsymbol{l},\boldsymbol{i}}(x),$$

where $I_{\boldsymbol{l}}$ is the set of the indices of the basis functions in the hierarchical increment $W_{\boldsymbol{l}}$ with level $\boldsymbol{l} = (l_1, \ldots, l_d)$, i.e.,

$$I_{\boldsymbol{l}} = \left\{ \boldsymbol{i} = (i_1, \ldots, i_d) : 1 \leq i_j \leq 2^{l_j} - 1,\ i_j \text{ odd},\ 1 \leq j \leq d \right\}. \tag{2}$$

Let us first assume that we do not have basis functions at the boundary, i.e., all functions of the space are zero at the boundary. Then the supports of the basis functions in a hierarchical increment are pairwise disjoint and we only need to consider one basis function of each hierarchical increment. This reduces the costs for an evaluation of u^n at point $x \in [0,1]^d$ from $\mathcal{O}(M)$ to the number of used hierarchical increments. A sparse grid space of level n and dimension d consists of

$$\mathrm{G}(n,d) = \sum_{i=d}^{n+d-1} \binom{i-1}{d-1} \tag{3}$$

hierarchical increments [3]. However, in the following examples, we will use sparse grids with boundaries and therefore the matter is not quite so simple anymore. Note that if we want to have non-zero boundaries, we permit indices 0 and 2 for level 1 and set the basis functions $\phi_{1,0}$ and $\phi_{1,2}$ to zero at $x \in \mathbb{R}\backslash[0,1]$. For dimension $d > 1$ the well-known tensor product approach is applied [3].

We introduce *inner* hierarchical increments $W_{\boldsymbol{l}}$ with $\boldsymbol{l} = (l_1, \ldots, l_d)$ and $l_i > 1$, $1 \leq i \leq d$.

Lemma 1. *The number of inner hierarchical increments of a sparse grid of level n and dimension d is*

$$\mathrm{iG}(n,d) = \mathrm{G}(n-d,d).$$

Proof. There are $\binom{i-1}{d-1}$ ways to form a sum equal i with d positive integers. Hence, there are $\binom{i-d-1}{d-1}$ ways to form a sum equal $i-d$ with d positive integers (or a sum equal i with d integers greater than one). Therefore, we obtain

$$\sum_{i=d+1}^{n+d-1} \binom{i-d-1}{d-1} = \sum_{i=1}^{n-1} \binom{i-1}{d-1} = \sum_{i=d}^{n-1} \binom{i-1}{d-1}.$$

Note, that the number of *inner* hierarchical increments $\mathrm{iG}(n,d)$ of a sparse grid of level n and dimension d equals the number of *all* hierarchical increments of a sparse grid of level $n-d$ and dimension d.

Lemma 2. *Let $W_{\boldsymbol{l}}$ be a hierarchical increment where the level $\boldsymbol{l}$ has a 1 in k components. Then we have to consider at most 2^k basis functions of $W_{\boldsymbol{l}}$ for the evaluation of $u^n \in V_n^{(1)}$ at $x \in [0,1]^d$.*

Proof. Let $x = (x_1, \ldots, x_d)$ be a point in $[0,1]^d$ and $W_{\boldsymbol{l}}$ the hierarchical increment with (w.l.o.g.) $\boldsymbol{l} = (l_1, \ldots, l_d) = (1, \ldots, 1, l_{k+1}, \ldots, l_d)$ and $l_j > 1$, $k < j \leq d$. We only need to evaluate those basis functions whose support contains x. Which means, we have to count the number of indices $(i_1, \ldots, i_d) \in I_{\boldsymbol{l}}$ with

$$\exists j : (i_j - 1)2^{-l_j} < x_j < (i_j + 1)2^{-l_j}.$$

We consider every direction $1 \leq j \leq d$ separately. If $l_j > 1$ then there exists only one j with $(i_j-1)2^{-l_j} < x_j < (i_j+1)2^{-l_j}$ because we do not have boundary basis functions. If $l_j = 1$, then i_j can take 1 and either 0 or 2 to satisfy $(i_j-1)2^{-l_j} < x_j < (i_j+1)2^{-l_j}$. Therefore, we have two feasible values for component j. Altogether, we have at most

$$\prod_{j=1}^{k} 2 \cdot \prod_{j=k+1}^{d} 1 = 2^k,$$

basis functions $\phi_{\boldsymbol{l},\boldsymbol{i}} \in W_{\boldsymbol{l}}$ whose support contains x.

Lemma 3. *In the case of inhomogeneous Dirichlet boundary conditions and the sparse grid space of level n and dimension d, we need to consider at most*

$$\mathrm{opSG}(n,d) = 2^d + \sum_{k=0}^{d-1} 2^k \binom{d}{k} \mathrm{iG}(n, d-k), \tag{4}$$

basis functions to evaluate u^n at point $x \in [0,1]^d$.

Proof. We have to count the number of hierarchical increments $W_{\boldsymbol{l}}$ with a 1 in $k < d$ components. We know that there are $\mathrm{iG}(n,d-k)$ ways to form a sum with $d-k$ components greater than 1 which is less or equal to $n+(d-k)-1$. The remaining k components are filled with 1. There are $\binom{d}{k}$ ways to distribute the k ones across

d places, or the other way around, there are $\binom{d}{d-k}$ ways to distribute the numbers greater than 1 across d places. In any case, we obtain

$$\binom{d}{k}\mathrm{iG}(n, d-k) = \binom{d}{d-k}\mathrm{iG}(n, d-k),$$

hierarchical increments $W_{\boldsymbol{l}}$ with 1 in k components. Furthermore, we know, that there is exactly one hierarchical increment with $\boldsymbol{l} = (1, \ldots, 1)$. Finally, formula (4) follows with Lemma 2.

3 Reduced Basis Method

We give an overview of the reduced basis method for the case of *parametrically coercive and compliant affine linear elliptic problems* as described in [11].

3.1 Reduced Basis Approximation

After some preliminary definitions we give an abstract formulation of our problem class and define the associated function spaces. Next, the truth approximation is developed, on which finally our reduced basis approximation is built.

3.1.1 Preliminaries

Let Z be a (real) vector space and $\mathcal{D} \subset \mathbb{R}^P$ be a closed bounded parameter domain which is suitably regular. We follow [11] and call a *parametric bounded linear form* $g : Z \times \mathcal{D} \to \mathbb{R}$ *affine in the parameter* if there exist a $Q_g \in \mathbb{N}$ and $\Theta_g^q : \mathcal{D} \to \mathbb{R}$, $1 \le q \le Q_g$ as well as parameter-independent bounded linear forms $g^q : Z \to \mathbb{R}$, $1 \le q \le Q_g$, such that

$$g(v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_g} \Theta_g^q(\boldsymbol{\mu}) g^q(v), \qquad \forall v \in Z. \tag{5}$$

That is, we can separate the parameter-dependent linear form g into its parameter-dependent parts Θ_g^q and parameter-independent parts g^q. Similarly, we call a parametric bilinear form $b : Z \times Z \times \mathcal{D} \to \mathbb{R}$ *affine in the parameter* if we can represent it as

$$b(w, v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_b} \Theta_b^q(\boldsymbol{\mu}) b^q(w, v), \qquad \forall w, v \in Z. \tag{6}$$

Furthermore, we say an affine parametric, symmetric and coercive[1] bilinear form $b : Z \times Z \times \mathcal{D} \to \mathbb{R}$ is *parametrically coercive* if b permits an affine decomposition as in (6) that satisfies $\Theta_b^q(\boldsymbol{\mu}) > 0, \forall \boldsymbol{\mu} \in \mathcal{D}$, $1 \leq q \leq Q_b$ and $b^q(v, v) \geq 0, \forall v \in Z$, $1 \leq q \leq Q_b$ with symmetric b^q, $1 \leq q \leq Q_b$.

3.1.2 Abstract Formulation and Truth Approximation

We first introduce our spatial (or physical) domain $\Omega \in \mathbb{R}^d$, a suitably regular open bounded domain with spatial dimension $d = 1, 2$, or 3. We denote the boundary of Ω as $\partial\Omega$ and assume $\partial\Omega$ is Lipschitz continuous. We next introduce the space X^e (^{e} refers to "exact") as

$$X^e = X^e(\Omega) = \left\{ v \in H^1(\Omega) : v \,|_{\Gamma^D} = 0 \right\}. \tag{7}$$

We define our parametric field variable u: $\mathcal{D} \to X^e$ and denote the field for parameter $\boldsymbol{\mu} \in \mathcal{D}$ as $u(\boldsymbol{\mu})$. If we view the field variable fully as $u : \Omega \times \mathcal{D} \to \mathbb{R}$, we denote the value of the field at point $x \in \Omega$ for parameter $\boldsymbol{\mu} \in \mathcal{D}$ as $u(x; \boldsymbol{\mu})$.

Finally, we can state the problem we are interested in. Let $f : X^e \times \mathcal{D} \to \mathbb{R}$ be an affine parametric linear form bounded over X^e and $a : X^e \times X^e \times \mathcal{D} \to \mathbb{R}$ be a continuous and parametrically coercive bilinear form. Given $\boldsymbol{\mu} \in \mathcal{D}$, we find $u^e(\boldsymbol{\mu}) \in X^e$ such that

$$a(u^e(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}), \qquad \forall v \in X^e, \tag{8}$$

and evaluate the output of interest

$$s^e(\boldsymbol{\mu}) := f(u^e(\boldsymbol{\mu}); \boldsymbol{\mu}). \tag{9}$$

Due to our coercivity and continuity assumptions on a we may introduce the (well-defined) inner product and induced norm as

$$\begin{aligned} (((w, v)))_{\boldsymbol{\mu}} &:= a(w, v; \boldsymbol{\mu}), \qquad \forall w, v \in X^e, \\ |||w|||_{\boldsymbol{\mu}} &:= \sqrt{a(w, w; \boldsymbol{\mu})}, \qquad \forall w \in X^e. \end{aligned}$$

Note, that the inner product as well as the norm are parameter-dependent. To get parameter-independent inner product and norm, we specify a parameter $\overline{\mu} \in \mathcal{D}$ and associate the inner product and norm to X^e

$$\begin{aligned} (w, v)_{X^e} &:= (((w, v)))_{\overline{\mu}}, \qquad \forall w, v \in X^e, \\ \|w\|_{X^e} &:= |||w|||_{\overline{\mu}}, \qquad \forall w \in X^e. \end{aligned}$$

[1] We refer to [11, Sect. 1.2.2] for a definition of *coercive* and *continuous* for *parametric* bilinear forms.

The choice of $\overline{\mu}$ influences the quality of the a posteriori error bounds, see [11] for details.

Let $X^{\mathcal{N}} \subset X^e$ be a subspace with $\dim(X^{\mathcal{N}}) = \mathcal{N} \in \mathbb{N}$. We equip $X^{\mathcal{N}}$ with the inner product and norm of the exact space X^e. Both the inner product and the norm are independent of the dimension $\mathcal{N}$. We can then apply a standard Galerkin projection. Given $\boldsymbol{\mu} \in \mathcal{D}$, find $u^{\mathcal{N}}(\boldsymbol{\mu}) \in X^{\mathcal{N}}$ such that

$$a(u^{\mathcal{N}}(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}), \qquad \forall v \in X^{\mathcal{N}}, \tag{10}$$

and then evaluate

$$s^{\mathcal{N}}(\boldsymbol{\mu}) := f(u^{\mathcal{N}}(\boldsymbol{\mu}); \boldsymbol{\mu}). \tag{11}$$

In the following and throughout this work, we will suppress the superscript $\mathcal{N}$. So, wherever X, $u(\boldsymbol{\mu})$ or $s(\boldsymbol{\mu})$ appears, it refers to $X^{\mathcal{N}}$, $u^{\mathcal{N}}(\boldsymbol{\mu})$ or $s^{\mathcal{N}}(\boldsymbol{\mu})$, respectively. However, we will maintain the superscript ^{e} for the exact solution or space.

3.1.3 Reduced Basis Approximation

Our particular problem stated in (8) and (9) can be seen as an input-output relationship which is induced by a partial differential equation [14]. If we want to approximate the output $s^e(\boldsymbol{\mu})$ by $s^{\mathcal{N}}(\boldsymbol{\mu})$, we have to compute $u^{\mathcal{N}}(\boldsymbol{\mu})$ with the FE method and finally evaluate $s^{\mathcal{N}}$ with $f(\cdot; \boldsymbol{\mu})$ at $u^{\mathcal{N}}(\boldsymbol{\mu})$. The difficulty is the computationally expensive solution $u^{\mathcal{N}}(\boldsymbol{\mu})$ of the PDE. If we consider this problem in real-time or many-query contexts, where we have to evaluate the output for many $\boldsymbol{\mu} \in \mathcal{D}$, we can easily think of many applications where it is not possible to compute a FE approximation for every single $\boldsymbol{\mu}$ we are interested in.

The reduced basis approach is based on two properties of the real-time and many-query context, which permit us to reduce the number of truth or FE approximations significantly. First, in most examples, the field variables $u(\boldsymbol{\mu})$ are very likely not scattered all over the space X but rather lie on a low-dimensional and smooth manifold [11]. That is, we assume

$$\mathcal{M} = \{u(\boldsymbol{\mu}) : \boldsymbol{\mu} \in \mathcal{D}\},$$

to be smooth and quite low-dimensional, and therefore represent $u(\boldsymbol{\mu})$ not as element of X but as element of $\operatorname{span}\{\mathcal{M}\}$. Note that the smoothness in the spatial coordinate x is not crucial [11, Sect. 3.5, p. 102]. Second, in real-time and many-query contexts, we can deal with increased Offline (pre-processing) costs and only have to decrease the Online costs, i.e., the costs for each evaluation for a new $\boldsymbol{\mu} \in \mathcal{D}$. This in turn means that the reduced basis approach is not to be preferred to the classical FE method in the few-query context.

For our reduced basis spaces we choose $N_{\max}$ linearly independent functions

$$\xi^n \in X, \qquad 1 \le n \le N_{\max},$$

and define our reduced basis spaces

$$X_N = \text{span}\{\xi^n,\ 1 \leq n \leq N\}, \qquad 1 \leq N \leq N_{\max}.$$

Although the RB method can be seen in such a general framework, we consider the case where we approximate $u(\boldsymbol{\mu})$ for $\boldsymbol{\mu} \in \mathcal{D}$ in spaces spanned by N precomputed solutions or snapshots $u(\boldsymbol{\mu}^1), \ldots, u(\boldsymbol{\mu}^N)$. Therefore, we introduce the sets of parameter points

$$S_N = \left\{\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^N\right\} \subset \mathcal{D}, \qquad 1 \leq N \leq N_{\max}$$

and the associated snapshots $u(\boldsymbol{\mu}^n)$, $1 \leq n \leq N_{\max}$ which span the reduced basis spaces

$$X_N = \text{span}\left\{u(\boldsymbol{\mu}^n) : 1 \leq n \leq N\right\} \subset X, \qquad 1 \leq N \leq N_{\max}.$$

The question of how to choose the parameter points $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^{N_{\max}}$ and hence the snapshots $u(\boldsymbol{\mu}^n)$, $1 \leq n \leq N_{\max}$ will be answered in Sect. 3.3. In order to ensure a well-conditioned reduced basis algebraic system, Gram-Schmidt orthogonalization in the $(.,.)_X$ inner product is applied to the snapshots $u(\boldsymbol{\mu}^1), \ldots, u(\boldsymbol{\mu}^{N_{\max}})$.

We can use Galerkin projection on the space X_N. Given $\boldsymbol{\mu} \in \mathcal{D}$, we find $u_N(\boldsymbol{\mu}) \in X_N$ such that

$$a(u_N(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}), \qquad \forall v \in X_N, \tag{12}$$

and evaluate

$$s_N(\boldsymbol{\mu}) := f(u_N(\boldsymbol{\mu}); \boldsymbol{\mu}). \tag{13}$$

From our assumptions on a (coercivity and continuity), $X_N \subset X$ and linear independence of $u(\boldsymbol{\mu}^1), \ldots, u(\boldsymbol{\mu}^{N_{\max}})$, we obtain a unique solution and the usual Galerkin optimality [11, Sect. 3.2.2, p. 85].

In [11, Sect. 4], rigorous, sharp and efficient a posteriori error bounds for the output error $|s^{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})|$ as well as for the error of the field variable $|||u^{\mathcal{N}}(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})|||_{\boldsymbol{\mu}}$ have been developed for the case of parametrically coercive and compliant affine linear elliptic problems.[2] Since we are only interested in the accuracy of the approximation of the output of interest s^e, we always use the error estimator $\Delta_N^{\text{s,rel}}$ of the relative error of the output of interest, i.e.

$$\left|\frac{s^{\mathcal{N}}(\boldsymbol{\mu}) - s_N(\boldsymbol{\mu})}{s^{\mathcal{N}}(\boldsymbol{\mu})}\right| \leq C\Delta_N^{\text{s,rel}}(\boldsymbol{\mu}),$$

where C depends on $\overline{\boldsymbol{\mu}}$ and $\boldsymbol{\mu}$. See [11, Sect. 4.3.2] for a definition of $\Delta_N^{\text{s,rel}}$ and computational details.

[2] However, there are also error bounds for more general problems, cf. [4–6, 8].

3.2 Offline-Online Computational Procedure

With the procedure we have discussed so far, for any new $\boldsymbol{\mu} \in \mathcal{D}$ we can evaluate the (reduced basis) output $s_N(\boldsymbol{\mu})$ by computing $u_N(\boldsymbol{\mu})$ with $\mathcal{O}(N^3)$ operations (solving a system of linear equations of size $N \times N$) and $s_N(\boldsymbol{\mu})$ with $\mathcal{O}(N)$ operations (computing a dot product). However, we have not included the formation of the stiffness matrix $A_N(\boldsymbol{\mu})$ which requires $\mathcal{O}(N^2)$ dot products with $\mathcal{O}(\mathcal{N})$ operations each.

We invoke the affine parameter dependence (5) and (6) of f and a which permits us to express the stiffness matrix $A_N(\boldsymbol{\mu})$ and the load/source vector $F_N(\boldsymbol{\mu})$ as linear combinations

$$A_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) A_N^q, \tag{14}$$

$$F_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \Theta_f^q(\boldsymbol{\mu}) F_N^q, \tag{15}$$

with the *parameter-independent* matrices $A_N^q \in \mathbb{R}^{N \times N}$, $1 \le q \le Q_a$,

$$A_{Nnm}^q = a^q(u(\boldsymbol{\mu}^n), u(\boldsymbol{\mu}^m)), \qquad 1 \le n, m \le N,\ 1 \le q \le Q_a,$$

and *parameter-independent* vectors $F_N^q \in \mathbb{R}^N$, $1 \le q \le Q_f$,

$$F_{Nn}^q = f^q(u(\boldsymbol{\mu}^n)), \qquad 1 \le n \le N,\ 1 \le q \le Q_f,$$

where $X_N = \text{span}\{u(\boldsymbol{\mu}^n),\ 1 \le n \le N\}$ is the reduced basis space. We assume that the coefficients $\Theta_a^q(\boldsymbol{\mu})$ for $1 \le q \le Q_a$ and $\Theta_f^q(\boldsymbol{\mu})$ for $1 \le q \le Q_f$ are easy to evaluate.

Hence, we split the computational procedure as follows [11, Sect. 3.3]. In the Offline stage (pre-processing), we compute the reduced basis $u(\boldsymbol{\mu}^j)$, $1 \le j \le N$. This means, we need N FE computations and the Gram-Schmidt orthogonalization. We then form the matrices A_N^q, $1 \le q \le Q_a$ and vectors F_N^q, $1 \le q \le Q_f$. This formation needs $Q_a N^2$ $(\cdot,\cdot)_X$ inner products with $\mathcal{O}(\mathcal{N})$ operations and $Q_f N$ evaluations of f^q. In the Online stage, for any given $\boldsymbol{\mu} \in \mathcal{D}$, we assemble the matrix $A_N(\boldsymbol{\mu})$ from (14) and $F_N(\boldsymbol{\mu})$ from (15) which needs $\mathcal{O}(Q_a N^2 + Q_f N)$ operations. We then solve (12) for $u_N(\boldsymbol{\mu})$ and evaluate $s_N(\boldsymbol{\mu})$.

3.3 Construction of Reduced Basis Spaces

In this section, we discuss how to construct reduced basis spaces. For certain problems with one-dimensional parameter domains $P = 1$, one can give a priori spaces of good quality [10]. However, this approach is very limited, and therefore

we present the *greedy strategy* developed in [11, Sect. 3.4]. This strategy computes an *ad-hoc* or "adaptive" space which is tailored to the problem.

The sampling strategy is not directly applied to the parameter domain $\mathcal{D}$ but to a train sample (a finite sample of points) $\Xi_{\text{train}} \subset \mathcal{D}$ which serves as a surrogate for $\mathcal{D}$. We can think of $\{(\boldsymbol{\mu}, u(\boldsymbol{\mu})) : \boldsymbol{\mu} \in \Xi_{\text{train}}\}$ as the training data for our reduced basis approximation. Typically, this training data is chosen randomly with respect to uniform or log-uniform distributions [12]. All our random training samples are uniformly distributed. The following algorithm constructs a problem-dependent reduced basis space with the snapshots corresponding to samples of Ξ_{train}. We will discuss more sophisticated techniques in Sect. 4.

We assume that we have given a train sample Ξ_{train} and an initial parameter in the set $S^1 = \{\boldsymbol{\mu}^1_{\text{train}}\} \subset \Xi_{\text{train}}$ as well as the space $X^*_1 = \text{span}\{u(\boldsymbol{\mu}^1_{\text{train}})\}$. We now specify a maximum dimension $N_{\max}$ for our reduced basis space and a tolerance ϵ. Then, the greedy algorithm proceeds as follows [11, Sect. 3.4.3]

for $N = 2$ to $N_{\max}$ **do**
  $\boldsymbol{\mu}^N = \arg\max_{\boldsymbol{\mu} \in \Xi_{\text{train}}} \Delta^{\text{s,rel}}_{N-1}(\boldsymbol{\mu})$
  $\epsilon^{N-1} = \max_{\boldsymbol{\mu} \in \Xi_{\text{train}}} \Delta^{\text{s,rel}}_{N-1}(\boldsymbol{\mu})$
  if $\epsilon^{N-1} \leq \epsilon$ **then**
    exit;
  end if
  $S^N = S^{N-1} \cup \{\boldsymbol{\mu}^N\}$
  $X^*_N = X^*_{N-1} + \text{span}\{u(\boldsymbol{\mu}^N)\}$
end for

At iteration N, the greedy algorithm appends to the space X^*_{N-1} the snapshot $u(\boldsymbol{\mu}^N)$ which is predicted to be the least well approximated snapshot of all $u(\boldsymbol{\mu})$, $\boldsymbol{\mu} \in \Xi_{\text{train}}$. Note that the greedy approach has very low cost and therefore permits a large Ξ_{train} and hence an exhaustive search in $\mathcal{D}$. By using the (efficient) a posteriori error bounds, the greedy algorithm needs at most $N_{\max}$ FE computations.

3.4 Model Problem (Thermal Block)

We restrict ourselves to one model problem, the *thermal block* as presented in [11]. This is a (steady) heat conduction problem with conductivities as parameters which satisfies the imposed restrictions in our parametrically coercive and compliant case.

We consider heat conduction in a square domain $\Omega = (0,1) \times (0,1)$. The square comprises $B_1 \times B_2$ rectangular subdomains Ω_i, $i = 1, \ldots, B_1 B_2$. Each of these subdomains is a different *region* with different thermal conductivity (non-dimensional form). Therefore, we have $P = B_1 B_2$ parameters $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_P) \in \mathcal{D}$ with $\mathcal{D} = [0.1, 10]^P$. We allow three different versions of the thermal block *TB*2, *TB*4 and *TB*8 with 2, 4 and 8 regions, respectively. Their geometry is shown in

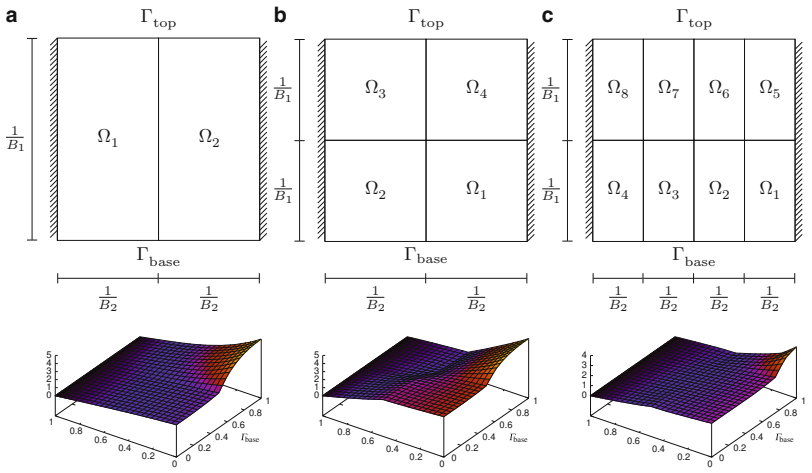


Fig. 1 The three versions of the thermal block problem. The geometries and below plots of $u(\boldsymbol{\mu}) \in X^{\mathcal{N}}$ for an example $\boldsymbol{\mu} \in \mathcal{D}$. (**a**) TB2, (**b**) TB4, and (**c**) TB8

Fig. 1. Our field variable is the temperature which satisfies Laplace's equation in Ω. Following the geometry and notation in Fig. 1, we impose homogeneous Dirichlet conditions on the top boundary Γ_{top}, homogeneous Neumann conditions on the two sides and Neumann 1 conditions on the bottom or base Γ_{base}. Our output of interest is the average temperature over the base Γ_{base}. With the abstract statement in (7) we identify $X^e = \{v \in H^1(\Omega) : v\,|_{\Gamma_{\text{top}}} = 0\}$ which imposes the essential Dirichlet boundary conditions. Our bilinear form is

$$a(w, v; \boldsymbol{\mu}) = \sum_{p=1}^{P} \mu_p \int_{\Omega_p} \nabla w \cdot \nabla v, \qquad \forall w, v, \in X^e,$$

and our source and output functional is

$$f(v; \boldsymbol{\mu}) = \int_{\Gamma_{\text{base}}} v, \qquad \forall v \in X^e,$$

which imposes the inhomogeneous Neumann conditions.

4 Sparse Grids in the Parameter Domain

In this section, we briefly discuss two sampling strategies to construct a reduced basis space. Whereas usually the train samples are simply chosen randomly, our two sampling strategies build on a sparse grid in the parameter domain. The first strategy adapts the sparse grid to the problem at hand, the second one constructs a

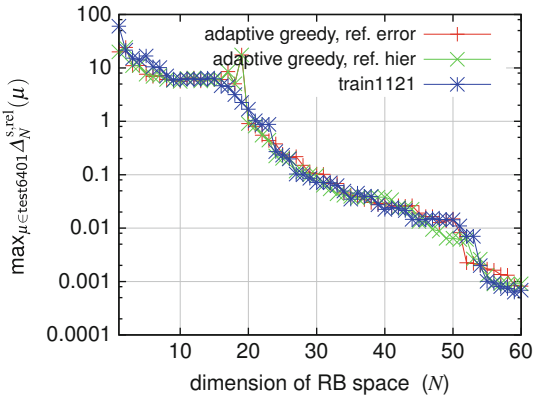


Fig. 2 The adaptive greedy approach for the thermal block *TB*8. Refinement with respect to the (predicted) errors and with respect to the (absolute) hierarchical coefficients compared to the greedy strategy with train sample train1121. The error is computed over the sample test6401

sparse grid interpolant. Note that the purpose of this section is to show by means of a model problem that such strategies are feasible with sparse grids.

4.1 Greedy Sampling Strategy with Adaptive Sparse Grids

For the thermal block, the greedy strategy works very well, even with simple train samples which do not take the problem into account. However, for other problem classes, we cannot expect such good behavior. Therefore, we now use adaptive train samples. Consider the points of a sparse grid as train sample. After every step of the greedy procedure, that is, after a new basis function has been added, we refine the grid point associated with the train sample. The points of the refined grid are then used as new train sample and so the sample gradually adapts to the problem. Note that such an extension is a natural one in the context of sparse grids because adaptivity requires the possibility to add points only in certain regions. Thus, a similar approach would not be possible with an e.g. quasi-Monte Carlo method without further ado.

We now compute an RB space for the *TB*8 thermal block with the parameter domain $\mathcal{D} = [0.1, 10]^8$. The initial train sample contains 1,280 points of a sparse grid of level 1 and dimension 8. The greedy procedure is started with space $X_1^* = \text{span}\{u(\overline{\mu})\}$. After every greedy step, one grid point gets refined. We compare two refinement criteria. The first criteria refines the point for which the error estimator $\Delta_N^{\text{s,rel}}$ yields the greatest (predicted) error. For the second criteria, we create the sparse grid interpolant of $\Delta_N^{\text{s,rel}}$ and refine the point with the greatest absolute hierarchical coefficient. The results are compared to the space constructed with a random train sample train1121 of size 1121 by computing the maximum predicted error over test6401 of size 6401, see Fig. 2. For our thermal block, we could not improve the RB approximation space with this method. Furthermore, the two refinement criteria do not develop a significant difference in the error behavior.

Nevertheless, there are problems for which such greedy strategies yield better RB approximation spaces than the non-adaptive greedy [7]. However, due to the curse of dimensionality such strategies were restricted to parameter domains of dimension 2 or 3. As we have shown in the example above, with sparse grids we can easily cope with parameter domains of higher dimensions.

4.2 Greedy Sampling Strategy with Sparse Grid Interpolant

We present another modification to the greedy strategy. As we have noted in Sect. 3.3, we consider a train sample Ξ as finite surrogate for $\mathcal{D}$ and hope to characterize the error estimator Δ over $\mathcal{D}$ by just using the values $\{(\boldsymbol{\mu}, \Delta(\boldsymbol{\mu}))\,;\, \boldsymbol{\mu} \in \Xi\}$. But if we use sparse grid points as train samples, we can construct the interpolant of the error estimator Δ in a sparse grid space $V_n^{(1)}$. We search the interpolant for maximizers and use the associated snapshots to span our RB space. The Sparse Grid Interpolation Toolbox was employed to build sparse grid interpolants and to search them for maximizers, cf. [9]. We interpolated $\Delta_N^{\mathrm{s,rel}}$ on a sparse grid of level[3] 4 and searched for local maximizers with the `spcompsearch` method.

Because finding global maximizers of sparse grid interpolants is computationally expensive, we want to use local maximizers. Therefore, we compare two methods for constructing RB spaces for *TB*4. The first method starts a search for a maximizer at every sparse grid point, the second method only at 10 different randomly picked start points. The corresponding RB spaces are compared in Fig. 3 (left). The error estimator $\Delta_N^{\mathrm{s,rel}}$ is computed over the grid points grid2769 of a four dimensional sparse grid of level 4 with boundary points and a random set rand2769 of size 2769. For our model problem, the error behavior associated with the local search starting at randomly picked points does not really differ from the one which starts at all sparse grid points. This is true for both test samples.

Let us now compare the RB spaces constructed by the greedy procedure with a train sample and with a sparse grid interpolant, respectively. We only use the method where we start the search for local maximizers at 10 different randomly picked start points. Figure 3 (right) shows the maximum of $\Delta_N^{\mathrm{s,rel}}$ over grid2769 and over rand2769 for RB spaces constructed with a random train sample of size 2769 and with an interpolant on a grid of level 4. We see no significant difference when we measure the approximation quality on the sample rand2769, but considering the sample grid2769, that in contrast to the uniform sample resolves the boundary of the parameter domain, we observe that the greedy strategy leaves a significant larger error in these samples than we get using the interpolant.

[3] In the notation of the `SPT` this is a level 3 grid.

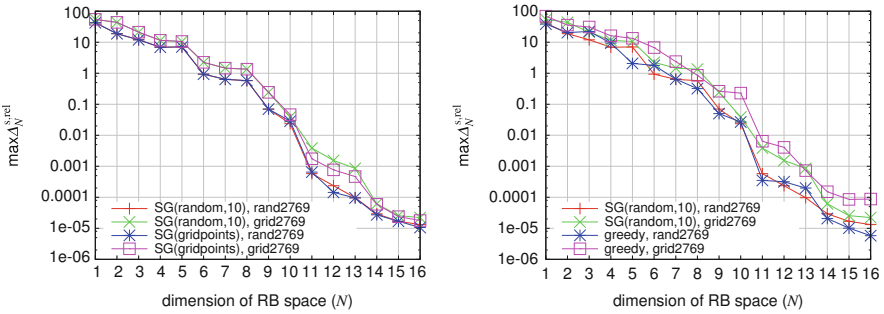


Fig. 3 Thermal block *TB*4. The error behavior for RB approximation space constructed by using sparse grid interpolants with local search starting at randomly picked points and all sparse grid points, respectively (*left*). Results for RB spaces constructed by greedy with ordinary train samples and with sparse grid interpolant (*right*)

5 Interpolation of the Output Function

Consider our problem statement of (12) and (13). Although, in some sense, we need $u(\boldsymbol{\mu})$ to compute our output of interest $s(\boldsymbol{\mu})$, the reduced basis method does not explicitly form $u(\boldsymbol{\mu})$. Actually, if we want to evaluate $u(\boldsymbol{\mu})$ at any $x \in \Omega$, the Online stage of the computational procedure becomes *dependent* on $\mathcal{N}$. This means, we are only interested in the output function $s : \mathcal{D} \to \mathbb{R}$. Instead of making the detour round the RB approximation, we can directly approximate s by interpolation. For higher parameter dimensions, this is not feasible with ordinary tensor product grids, but with sparse grids, we can cope with higher dimensions to some extent.

We can state two advantages of the direct interpolation of the output function s over the reduced basis method. First, we do not need to know anything about the underlying governing equations.[4] In fact, we only need a pool of precomputed truth solutions which are used to construct the sparse grid interpolant. Second, because we only need to evaluate a sparse grid function to compute the value of s at a given parameter $\mu \in \mathcal{D}$, we save the Galerkin projection into the reduced basis space. Hence, we do not need to develop and implement a solver to compute the projection. The reduced basis method as presented in [11] needs such a solver which is a severe limitation, especially for applications (crash test simulation, plasma physics, etc.) building on systems of equations where the implementation of the solver for the truth solutions is a challenging task on its own.

Note, that the Galerkin optimality of $u_N(\boldsymbol{\mu})$ does not in any way preclude an output interpolant that is better than the RB approximation. There are even examples of polynomial interpolants which are distinctly better than the RB approximation,

[4] Of course we can tune the sparse grid interpolation if we know some properties of the underlying governing equations. For instance, special refinement strategies or transformations might then be applicable and might help to improve the accuracy.

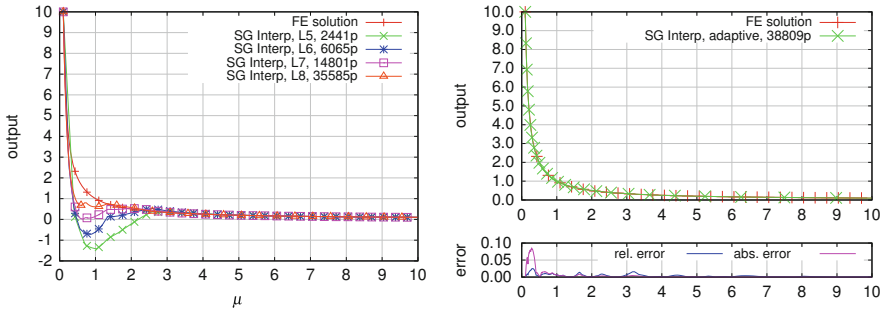


Fig. 4 Thermal block *TB*4. The sparse grid interpolants of level 5–8 compared to the output $s^{\mathcal{N}}$ for the FE solution (*left*). Interpolant of an adaptive sparse grid compared to the FE output $s^{\mathcal{N}}$ (*right*)

see [11, Sect. 3.5.3, p. 125]. However, these are only examples with one-dimensional parameter domains.

5.1 Interpolation on Adaptive Sparse Grids

We first consider the thermal block *TB*4 with four parameters. We interpolate the associated output function $s^{\mathcal{N}} : \mathcal{D} \to \mathbb{R}$ on sparse grids of levels 5–8 and compare the interpolants $s_n^{(1)}$ to $s^{\mathcal{N}}$ for the points on the diagonal

$$\left\{\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3, \mu_4) \in [0.1, 10]^4 : \mu_1 = \mu_2 = \mu_3 = \mu_4\right\},$$

see Fig. 4. This simple interpolation has severe problems between 0.1 and 2 where it differs in parts completely from the desired values. Even for level 8 with about 35,000 points, the (absolute) output error $|s_8^{(1)}(\boldsymbol{\mu}) - s^{\mathcal{N}}(\boldsymbol{\mu})|$ is still about 1. However, the function seems to be only locally difficult to approximate, because starting from 4 the interpolants achieve better results.

One way to handle functions with local singularities is to employ adaptivity. We start with a rather coarse grid and add only points where they are needed. As we have already mentioned in Sect. 4.1, one of the most common refinement criteria in the context of sparse grids is to refine those points associated to the basis functions with large absolute hierarchical coefficients. We start with a sparse grid of level 2 with 136 grid points to which we apply refinement steps. In each step, we sort the grid points by their absolute hierarchical coefficient and refine the first percent. After 12 steps, we arrive at a grid with approximately 38,000 points. The results are shown in Fig. 4 (right). We clearly get better results with adaptivity. That is, our refinement criterion finds the singularities and so makes sure that new points are not wasted. Nevertheless, we still need about 38,000 grid points.

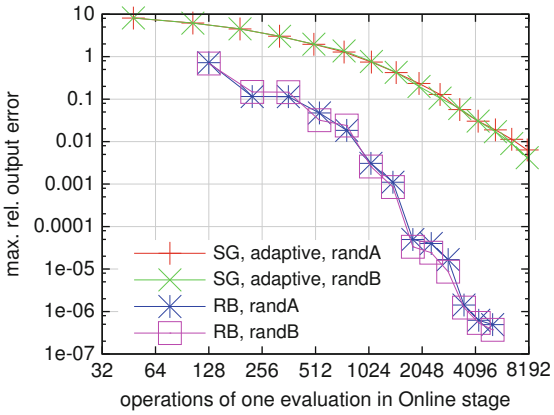


Fig. 5 Thermal block *TB*4. Interpolant of an adaptive sparse grid compared to the reduced basis approximation

We now want to compare the RB approximation to the sparse grid interpolant with respect to the operations needed for one evaluation of the output in the Online stage. For the RB approximation in an RB space of dimension N we need roughly $N^3 + Q_a N^2$ operations, i.e., $\mathcal{O}(Q_a N^2)$ operations to form the (parameter-dependent) stiffness matrix and $\mathcal{O}(N^3)$ operations to solve the system of linear equations. To evaluate a sparse grid interpolant on a grid of level n and dimension d we need $\mathcal{O}(\text{opSG}(n,d))$ operations, cf. Sect. 2. In the following, we do not use the error estimator $\Delta_N^{\text{s,rel}}$ to predict the output error, but the maximum relative output error. The maximum relative output error over Ξ is defined as

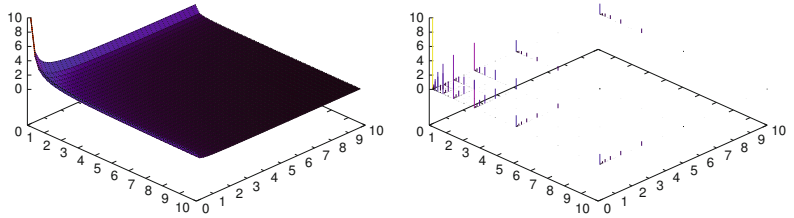
$$\max_{\boldsymbol{\mu}\in\Xi} \frac{|\tilde{s}(\boldsymbol{\mu}) - s^{\mathcal{N}}(\boldsymbol{\mu})|}{|s^{\mathcal{N}}(\boldsymbol{\mu})|},$$

where $\tilde{s}$ is either the RB approximation s_N or the sparse grid interpolant $s_n^{(1)}$.

Consider now Fig. 5, where we compare the adaptive sparse grid interpolant to the reduced basis approximation. Here, we estimate the operations for the evaluation of the interpolant on an adaptive sparse grid with maximum level n by the operations of the regular grid of level n. The RB spaces are constructed with a random train sample of size 2769 and reach from dimension 4 to 16. The maximum relative output error is computed over the random test samples randA and randB of size 2769. In Fig. 5, we clearly see the superiority of the RB approximation even though we employed adaptivity.

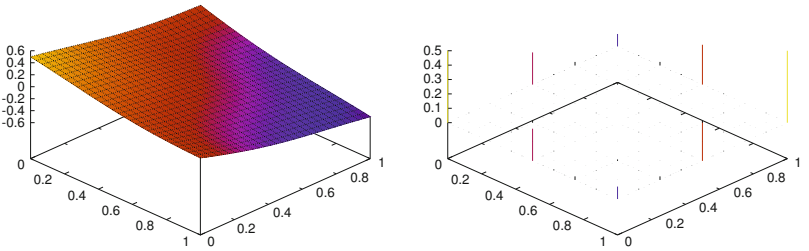
5.2 *The Output Function on a Logarithmic Scale*

Before we discuss the thermal blocks with higher parameter dimensions, we consider the thermal block *TB*2 with only two parameters $\boldsymbol{\mu} = (\mu_1, \mu_2)$ because then we can plot the interpolant of the output function $s^{\mathcal{N}}$ and its absolute



(a) TB2 output function (b) Hier. coefficients of interpolant

Fig. 6 In (**a**) interpolant of output function on sparse grid of level 10 and in (**b**) the corresponding absolute hierarchical coefficients



(a) TB2 output function on log. scale (b) Hier. coefficients of interpolant

Fig. 7 In (**a**) interpolant of output function on sparse grid of level 10 on logarithmic scale and in (**b**) the corresponding absolute hierarchical coefficients

hierarchical coefficients, see Fig. 6. The function $s^{\mathcal{N}}$ is only difficult to approximate near the corner $(0.1, 0.1)$, as we have already seen in Fig. 4. We also see, that the hierarchical coefficients are concentrated near $(0.1, 0.1)$. This indicates, that we use a wrong scale. Consider now Fig. 7 which again shows the interpolant of $s^{\mathcal{N}}$ and its hierarchical coefficients, but on a base 100 logarithmic scale. This change of scale removes the spike at $(0.1, 0.1)$ completely. Furthermore, the hierarchical coefficients are now distributed equally across the domain.

For the transformation, we have to specify $[\mu_{\min}, \mu_{\max}]$. For our thermal blocks we always have $\mu_{\min} = 0.1$ and $\mu_{\max} = 10$. We then define the transformation,

$$t_1 : [0, 1] \to [\mu_{\min}, \mu_{\max}],\ \mu \mapsto \mu_{\min} \left(\frac{\mu_{\max}}{\mu_{\min}} \right)^{\mu}. \tag{16}$$

To $\boldsymbol{\mu} \in [0, 1]^P$, $P > 1$ we apply t_1 component-wise, that is, $t(\boldsymbol{\mu}) = (t_1(\mu_1), \ldots, t_1(\mu_P))$. The inverse of t is denoted as t^{-1}. We then interpolate the function $\hat{s}^{\mathcal{N}} : [0, 1]^P \to [0, 1]$,

$$\hat{s}^{\mathcal{N}}(\boldsymbol{\mu}) = t^{-1} \left(s^{\mathcal{N}}(t(\boldsymbol{\mu})) \right),$$

and can compute the output function with

$$s^{\mathcal{N}}(\boldsymbol{\mu}) = t \left(\hat{s}^{\mathcal{N}}(t^{-1}(\boldsymbol{\mu})) \right).$$

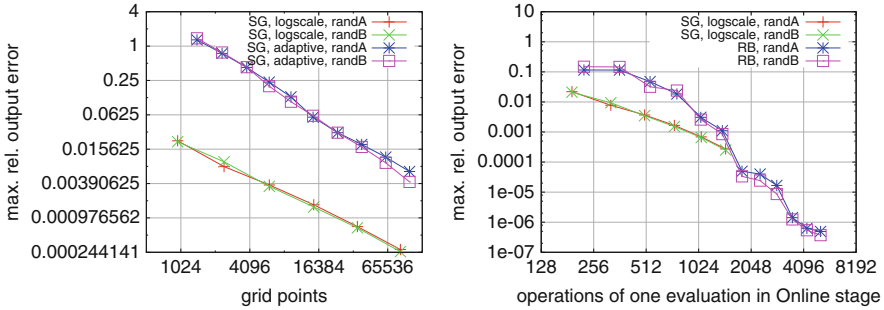


Fig. 8 Error for interpolant of $s^{\mathcal{N}}$ on logarithmic scale compared to the error of interpolant on an adaptive sparse grid (*left*). Maximum relative output error of RB approximation and of interpolant of transformed output $\hat{s}^{\mathcal{N}}$ (*right*)

Figure 8 (left) shows the maximum relative output error for the interpolant of $s^{\mathcal{N}}$ on the adaptive sparse grid and for the interpolant of the transformed output function $\hat{s}^{\mathcal{N}}$ on a regular sparse grid. The error is measured with respect to the samples randA and randB. We see that the transformation of (16) results in a tremendous improvement.

Let us now compare the interpolant of $\hat{s}^{\mathcal{N}}$ to the RB approximation, see Fig. 8 (right). The interpolants are computed on sparse grids from level 4 to 9, the dimensions N of the RB spaces reach from 5 to 16. With the transformation t, the interpolants perform better than the RB approximations at the beginning. This shows that for rough approximations the interpolants yield better results than the RB method with respect to the number of operations needed for one evaluation in the Online stage. However, we also see that the error of the interpolants decreases slower than the error of the RB approximation. Thus, we can expect that the RB approximation is a better approximation if we need a high-fidelity respond. Furthermore, the construction of an interpolant is very expensive, that is, the Offline costs are very high. For an interpolant on a grid with N points, we need to compute N FE solutions. A regular sparse grid with boundary of level 8 and dimension 4 has about 35,000 points, a grid of level 9 already 84,481 points. However, the fast construction of the RB space with the greedy strategy is only possible because of the (inexpensive) a posteriori error estimator. Such an estimator is only available for a limited class of equations and if it is not available, distinctly more costly sampling strategies are necessary as well, cf. [11].

Let us now consider the thermal block *TB*8 with eight parameters and $Q_a = 8$. We again use the transformed output $\hat{s}^{\mathcal{N}}$ and compute the interpolants on sparse grids of level 1–5. Furthermore, we compute an interpolant on an adaptive sparse grid, starting with level 1 and the hierarchical coefficients refinement criterion, see Sect. 5.1. But now we refine only the first percent of the grid points. As is shown in Fig. 9 (left), the adaptivity does not improve the error behavior at all. That again means, we have chosen the right scale.

Finally, we compare in Fig. 9 (right) the RB approximation for *TB*8 to the interpolant on the regular grid of the transformed output $\hat{s}^{\mathcal{N}}$. We show the

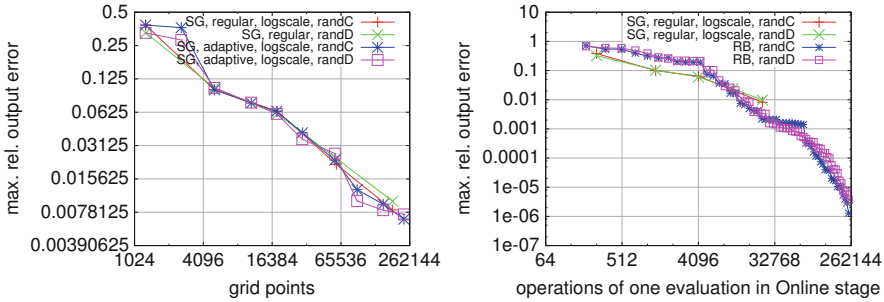


Fig. 9 Interpolation of the transformed $s^{\mathcal{N}}$ on regular sparse grids and on adaptive sparse grids for the thermal block *TB*8 (*left*). Comparison of the interpolation of the transformed $s^{\mathcal{N}}$ and the RB approximation for the thermal block *TB*8 (*right*)

approximations in the RB spaces up to dimension 60. Again, the sparse grid interpolant yields better results at the beginning and, thus, is better suited for rough approximations. Note that in many applications in science and engineering rough approximations are sufficient (e.g. visualization).

6 Conclusion

We addressed the employment of sparse grids in the reduced basis method, i.e. discretizing the PDE in the spatial domain Ω on sparse grids, using sparse grid points as surrogate for the parameter domain $\mathcal{D}$ and interpolating the output of interest on sparse grids.

In the parameter domain, we replaced the random, thus non-deterministic, train samples with sparse grid points. The grid samples, in contrast to e.g. quasi-Monte Carlo (low discrepancy sequences), permit adaptive refinement. Two extensions of the greedy approach have been discussed: adapting the grid sample to the problem at hand and using the sparse grid interpolant as surrogate of the a posteriori error estimator Δ in $\mathcal{D}$. Even though we could not distinctly improve the results for our model problem, similar strategies have already been proven useful in a wider context [7, 13], [11, Sect. 3.4.3]. Furthermore, the RB method gets rapidly extended to other problem classes [1, 2, 5, 6]. Our examples with sparse grids have shown that such strategies are feasible for higher parameter dimensions.

The aim of the RB method is to evaluate the output function $s : \mathcal{D} \to \mathbb{R}$. In the main part of the paper, we took a closer look at the output function of our model problem and, instead of making the detour round the RB method, we directly interpolated s on a sparse grid. This means, we can respond to a new parameter $\mu \in \mathcal{D}$ by simply evaluating a sparse grid function. With this approach we could overcome a severe limitation of the RB method: we do not need a Galerkin projection into the reduced basis space anymore. Thus, we do not need to develop and implement a solver to compute the projection. Especially for applications

building on a system of highly complex equations usually only a solver for the truth solution is available and it is not feasible to add another solver for the reduced basis. With our examples we have shown that the interpolation of the output function s on sparse grids yields competitive results compared to the RB approximation. For rough approximations we could even beat the RB approximation with respect to the number of operations needed for one evaluation in the Online stage. Such rough but very fast evaluations are necessary in e.g. visualization applications.

References

1. Barrault, M., Maday, Y., Nguyen, N.C., Patera, A.T.: An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. Comptes Rendus Mathematique **339**(9), 667–672 (2004)
2. Boyaval, S., Bris, C.L., Maday, Y., Nguyen, N.C., Patera, A.T.: A reduced basis approach for variational problems with stochastic parameters: Application to heat conduction with variable robin coefficient. Computer Methods in Applied Mechanics and Engineering **198**(41–44), 3187–3206 (2009)
3. Bungartz, H.J., Griebel, M.: Sparse grids. Acta Numerica **13**, 147–269 (2004)
4. Chen, Y., Hesthaven, J.S., Maday, Y., Rodriguez, J.: A monotonic evaluation of lower bounds for inf-sup stability constants in the frame of reduced basis approximations. Comptes Rendus Mathematique **346**(23–24), 1295–1300 (2008)
5. Cuong, N.N.: Reduced-basis approximations and a posteriori error bounds for nonaffine and nonlinear partial differential equations: Application to inverse analysis. Ph.D. thesis, HCMC University of Technology - SINGAPORE-MIT ALLIANCE (2005)
6. Grepl, M.A., Patera, A.T.: A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. M2AN **39**(1), 157–181 (2005)
7. Haasdonk, B., Ohlberger, M.: Adaptive basis enrichment for the reduced basis method applied to finite volume schemes. In: Proc. 5th International Symposium on Finite Volumes for Complex Applications, June 08–13, 2008, Aussois, France, pp. 471–478 (2008)
8. Huynh, D., Rozza, G., Sen, S., Patera, A.: A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup stability constants. Comptes Rendus Mathematique **345**(8), 473–478 (2007)
9. Klimke, A.: Sparse Grid Interpolation Toolbox – user's guide. Tech. Rep. IANS report 2007/017, University of Stuttgart (2007)
10. Maday, Y., Patera, A.T., Turinici, G.: Global a priori convergence theory for reduced-basis approximations of single-parameter symmetric coercive elliptic partial differential equations. Comptes Rendus Mathematique **335**(3), 289–294 (2002)
11. Patera, A.T., Rozza, G.: Reduced Basis Approximation and *A Posteriori* Error Estimation for Parametrized Partial Differential Equations. MIT Pappalardo Graduate Monographs in Mechanical Engineering (2007)
12. Rozza, G., Huynh, D.B.P., Patera, A.T.: Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: Application to transport and continuum mechanics. Archives of Computational Methods in Engineering **15**(3), 229–275 (2008)
13. Rozza, G., Veroy, K.: On the stability of the reduced basis method for stokes equations in parametrized domains. Computer Methods in Applied Mechanics and Engineering **196**(7), 1244–1260 (2007)
14. Veroy, K.: Reduced-basis methods applied to problems in elasticity: Analysis and applications. Ph.D. thesis, Department of Civil and Environmental Engineering - Massachusetts Institute of Technology (2003)

Spatially Adaptive Refinement

Dirk Pflüger

Abstract While sparse grids allow one to tackle problems in higher dimensionalities than possible for standard grid-based discretizations, real-world applications often come along with requirements or restrictions which enforce problem-dependent adaptations of the standard sparse grid technique. Consider, for example, interpolations where the function values at grid points are obtained via time-consuming numerical simulations. Then, only very few grid points can be spent; classical convergence might be out of reach. Another hurdle is that real-world problems often do not meet the smoothness requirements of the sparse grid method. Thus, the standard approach has to be fine-tuned to the problem at hand, especially in higher-dimensional settings.

Therefore, a suitable choice of basis functions can be required, as well as criteria for problem-adapted refinement. Fortunately, and in contrast to full grids, the hierarchical basis formulation of the direct sparse grid approach conveniently provides a reasonable criterion for spatially adaptive refinement practically for free. This can serve as a starting point to develop suitable modifications.

We show several problems stemming from different fields of application and demonstrate modifications of the standard sparse grid approach. They enable one to cope with the properties and requirements of the corresponding problem and can serve as examples for similar challenges.

D. Pflüger (✉)
Institute for Parallel and Distributed Systems, University of Stuttgart, 70569 Stuttgart, Germany
e-mail: Dirk.Pflueger@ipvs.uni-stuttgart.de

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_12,

1 Introduction

Sparse grids allow us to cope with the curse of dimensionality, at least to some extent. The term "sparse grids" has been coined for the solution of partial differential equations [28], and sparse grids have meanwhile been applied to various problems, see, for example, the survey in [4]. More recent work includes stochastic and non-stochastic partial differential equations in various settings [3, 11, 25, 27], as well as applications in economics [18, 22], regression [12, 15, 21], classification [6, 14, 21], fuzzy modeling [19], and more.

For sufficiently smooth functions, sparse grids enable one to reduce the number of grid points by orders of magnitude from $O((2^n)^d)$ for full grids to only $O(2^n n^{d-1})$, while keeping a similar accuracy as in the full grid case. To obtain these bounds, only a certain degree of smoothness is required (the mixed second derivatives have to be bounded), but no other knowledge about the problem at hand. Note, that the constants in the Landau-notation depend on the application.

To be able to deal with problems that do not meet the smoothness requirement, or to further reduce the number of grid points for functions that exhibit a low effective dimensionality or that contain regions of small and large variation, adaptivity can be employed. To this end, the hierarchical basis directly provides a straightforward indicator where to refine in general. Whereas this is typically very good to start with, the success for real-world applications often depends on a suitable choice of the basis, the criterion for adaptive refinement, and the selection of refinement parameters. Therefore, available knowledge about the problem should be used wherever possible.

To illustrate this, we picked a few applications which nicely demonstrate several approaches for adaptive refinement, and we give hints what to look for and consider when thinking about adapting to a problem at hand.

2 Sparse Grids

To briefly recall the most important properties and to clarify our notation, we describe the basic principles of sparse grids in the following; see, e.g., [4, 21] for further details. Sparse grids are based on a hierarchical (and thus inherently incremental and adaptive) formulation of the one-dimensional basis which is then extended to the d-dimensional setting via a tensor product approach.

We consider high-dimensional piecewise d-linear functions $f_N : [0,1]^d \to \mathbb{R}$ (which are defined on an equidistant mesh and scaled to the unit-hypercube) as a weighted sum of N basis functions,

$$f_N(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \varphi_i(\mathbf{x}) \,. \tag{1}$$

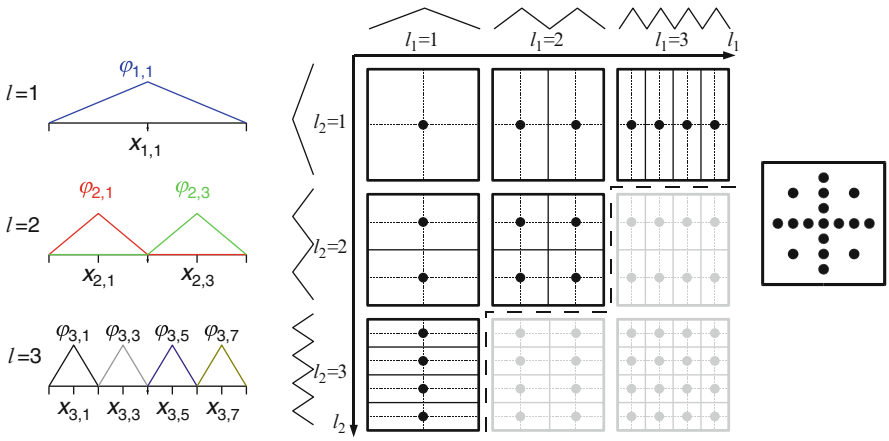


Fig. 1 One-dimensional basis functions up to level 3 (*left*), and tableau of hierarchical increments $W_{\mathbf{l}}$ up to level 3 in both dimensions (*middle*). Leaving out the grayed-out $W_{\mathbf{l}}$, we obtain the sparse grid of level 3 (*right*)

We therefore derive one-dimensional basis functions $\varphi_{l,i}$, depending on a level l and an index i, out of the reference hat function $\varphi(x) := \max(1-|x|, 0)$ via translation and scaling as $\varphi_{l,i}(x) := \varphi(2^l x - i)$. The hierarchical basis for a certain level n with mesh-width $h_n = 2^{-n}$ is then

$$\Phi_n := \left\{\varphi_{l,i}(x) : i = 1, \ldots, 2^l - 1,\ i \text{ odd},\ 1 \le l \le n\right\}, \tag{2}$$

omitting even-indexed basis functions on each level, see Fig. 1 (left) for $n = 3$. If we denote $\mathbf{l}$ and $\mathbf{i}$ as multi-indices of levels and indices for a certain basis function, we can then write d-dimensional basis functions $\varphi_{\mathbf{l},\mathbf{i}}$ as a product of the respective one-dimensional ones,

$$\varphi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) := \prod_{k=1}^{d} \varphi_{l_k,i_k}(x_k)\,,$$

obtaining piecewise d-linear basis functions. They are centered at grid points $\mathbf{x}_{\mathbf{l},\mathbf{i}} = (i_1 2^{l_1}, \ldots, i_d 2^{l_d})$, and $|\mathbf{l}|_1$ denotes the classical l^1 norm for vectors, i.e. the sum of the one-dimensional levels.

In higher-dimensional settings, we obtain hierarchical increments (function spaces) $W_{\mathbf{l}}$ for which the grid points are the Cartesian product of the one-dimensional ones on the respective one-dimensional levels. We denote the corresponding basis $\Phi_{\mathbf{l}}$, i.e. $\text{span}(\Phi_{\mathbf{l}}) = W_{\mathbf{l}}$. Figure 1 (middle) shows the grids of the two-dimensional hierarchical increments $W_{\mathbf{l}}$ up to level 3 in each dimension. Note that in each $W_{\mathbf{l}}$, all basis functions have supports with piecewise disjoint interiors.

The hierarchical representation now allows one to select only those subspaces that contribute most to the overall solution. This can be done by an a priori selection (see [4] for details). We then obtain a sparse grid space such as

$$V_n^{(1)} := \bigoplus_{|\mathbf{l}|_1 \leq n+d-1} W_{\mathbf{l}},$$

which in this case is optimized with respect to both the L^2-norm and the maximum-norm. In the example in Fig. 1, we can neglect the gray $W_{\mathbf{l}}$ for $n = 3$, which leads to the regular (non adaptive) sparse grid in Fig. 1 (right). To this end, the function f under consideration has to be sufficiently smooth, i.e., the mixed second derivative $\left|D^{\mathbf{2}} f\right| := \left|\frac{\partial^{2d}}{\partial x_1^2 \cdots \partial x_d^2} f\right|$ has to be bounded.

3 Adaptivity

A straightforward possibility to adapt to a problem at hand is to refine by adding new subspaces, weakening the diagonal cut-off in the subspace scheme. Adding a new subspace $W_{\mathbf{l}}$ in an incremental way requires that all backward neighbors, i.e., all subspaces $W_{\mathbf{l}'}$ with $\mathbf{l}' \leq \mathbf{l}$ (componentwise comparison of multi-indices), have already been included in the current set of subspaces. As this refinement treads all grid points with respect to a single dimension in a uniform way, this is referred to as dimensionally adaptive refinement. Note that this corresponds to the adaptive refinement in the context of the combination technique [16].

To determine where to refine, one can consider all subspaces that can be added next (for which all the backward neighbors exist) and add the one which reduces the error most (based on a suitable error measure). Unfortunately, this is infeasible in many applications, such as settings where function values are costly to obtain or where PDEs have to be solved: too many grid points which will quite likely never be used have to be examined. A frequently used alternative is to consider all current subspaces that can still be refined. The one which contributes most to the error is identified and refined by creating all the missing direct forward neighbors (incrementing the level in one of the dimensions each). This approach typically reduces the computational effort significantly.

When working in the direct, hierarchical sparse grid basis, the same approaches can be applied. But, in contrast to the combination technique, single grid points can be easily refined, and spatial (local) adaptivity can be employed. New basis functions just extend the current basis, and no side-effects such as hanging nodes have to be considered. In the d-dimensional hierarchical structure, a grid point has $2d$ children and up to d parents. The $2d$ children of a grid point $\mathbf{x}_{\mathbf{l},\mathbf{i}}$ are $\{\mathbf{x}_{\tilde{\mathbf{l}},\tilde{\mathbf{i}}} : \tilde{l}_k = l_k + 1, \tilde{i}_k = i_k \pm 1, \tilde{l}_t = l_t, \tilde{i}_t = i_t, t \neq k, k = 1, \ldots, d\}$. Refinable grid points (or leaves) in the hierarchical structure are all those grid points for which at least one child does not exist yet. New direct candidates are all grid points for which all the parents are contained in the current grid, similar to the subspaces in the dimensionally adaptive refinement.

Considering the refinement of single grid points, it is even more obvious that it is infeasible to look at all new candidates: already in the one-dimensional case there

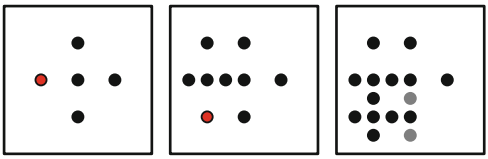


Fig. 2 Starting with a regular grid of level 2 (*left*), we refine one grid point (emphasized) by creating all children in the hierarchical tree of basis functions (*middle*), and we repeat this once more. To keep the grid consistent, two missing parents have to be created (*right*)

are as many grid points on the next level as there are in the current grid—and they all have to be considered. Solving a PDE, e.g., this would require to solve the PDE N times for N new candidates to select the one with the lowest error. Therefore, the standard strategy is to consider all refinable grid points of the current grid (for which all information has already been obtained), and refine the most promising one(s) by adding all missing children.

Note that for conventional algorithms working on sparse grids, all hierarchical ancestors of each grid point have to exist. To keep a grid consistent, all missing parents of new grid points have to be created recursively. This can result in significantly more than $2d$ grid points to be created per refinement, see Fig. 2 for an example.

The hierarchical basis, in contrast to the nodal basis, inherently provides a simple, though effective criterion of where to refine to minimize the L^2-norm of the error. Consider a sparse grid interpolant $f_N \in V_n^{(1)}$, $f_N(\mathbf{x}) = \sum_{|\mathbf{l}|_1 \leq n+d-1} \sum_{\varphi_{\mathbf{l},\mathbf{i}} \in \Phi_{\mathbf{l}}} \alpha_{\mathbf{l},\mathbf{i}} \varphi_{\mathbf{l},\mathbf{i}}(\mathbf{x})$, with surplusses (hierarchical coefficients) $\alpha_{\mathbf{l},\mathbf{i}}$. Recall, that the smoothness requirement which we impose at f is that $\left| D^{\mathbf{2}} f \right|$ has to be bounded. The surplusses can then be expressed via their integral representation as

$$\alpha_{\mathbf{l},\mathbf{i}} = \left(\prod_{j=1}^{d} \frac{-h_j}{2} \right) \int_{\Omega^d} \varphi_{\mathbf{l},\mathbf{i}}(\mathbf{x})\, D^{\mathbf{2}} f \, d\mathbf{x}\,,$$

see [4] for details.

Two important lessons can be learned: First, both the product of the mesh-widths h_j and the size and shape of the basis function $\varphi_{\mathbf{l},\mathbf{i}}$ depend only on $\mathbf{l}$ and not on the index of the grid point. For each grid point on the same level (with constant $|\mathbf{l}|_1$), the absolute value of the surplus depends mainly on $|D^{\mathbf{2}} f|$ within the support of the corresponding basis function. The more f varies on the support, the higher in general the absolute value of the surplus. Thus, the absolute value of the hierarchical coefficient can directly be used as a criterion for adaptive refinement, assuming a similar level sum for all refinement candidates. The fact that the hierarchical basis provides a cheap, simple criterion for refinement and does not necessitate to derive error estimators depending on the problem at hand, is one of the main advantages of the hierarchical approach. Note that, of course, there is no guarantee that this works in every setting, and that each and every criterion of where to refine can be fooled.

Second, $D^2 f$ can be assumed to be locally constant in the case of convergence. Then the contribution of a basis function decreases by one fourth when increasing the level by one in one dimension. The surplus can thus be used to check for convergence with respect to the discretization level or to detect unwanted noise in the function values [21]. For the interpolation of the normalized product of one-dimensional parabolas $f(\mathbf{x}) := 4^d \prod_{k=1}^{d}(1 - x_k)x_k$, this decay of the surplusses can be observed right from the second level. This is the reason why this function is frequently used to test and illustrate sparse grid interpolation.

4 Problem-Awareness

Refining grid points in the hierarchical basis, an incremental method is obtained which is equipped with a simple criterion and adapts locally to the problem at hand. But experience shows that in real-world settings a mere surplus-based refinement might not be sufficient to solve a problem with sufficient accuracy, and convergence might even be out of reach. The reasons for this are manifold and typically caused by the fact that the number of grid points is limited. For example, a problem with low effective dimensionality is posed in a too high-dimensional setting and the number of grid points grows too fast. Or the number of grid points that can be spent is restricted because it is very expensive to obtain or store function evaluations: each grid point might require an expensive simulation, or the memory requirements for storing a whole $3d$ simulation result at each grid point might be too high. There, the number of grid points that can be spent is severely limited. Furthermore, the problem itself could impose additional requirements to the sparse grid function.

In all these cases, the standard adaptive sparse grid approach has to be adapted to the problem at hand. In the following, we show some problems and strategies to give a start even if no additional knowledge about the problem is available. Ingredients are a suitable choice of the one-dimensional basis functions, threshold-based refinement, coarsening, weighted adaptivity, and trading off broad against steep refinement. But first, let us comment on the treatment of the boundary of the sparse grid domain Ω.

So far, we have only considered functions that are zero on the domain's boundary $\delta\Omega$. To allow for non-zero values on the boundary, usually additional grid points located directly on $\delta\Omega$ are introduced. The most common approach is to spend two more degrees of freedom in the one-dimensional hierarchical scheme for the grid on level 1 (which up to now only used $\varphi_{1,1}$), namely the basis functions with level 0 and indices 0 and 1. This leads in the d-dimensional case to 3^d unknowns for the initial grid and introduces an exponential dependency on the dimensionality that is significant for practical computations. For a problem in 100 dimensions, we could not even start computing the solution even if there was only one relevant dimension.

Therefore, the grid points on the boundary should be omitted wherever possible. Instead, the basis functions adjacent to the boundary have to be modified. A good choice is

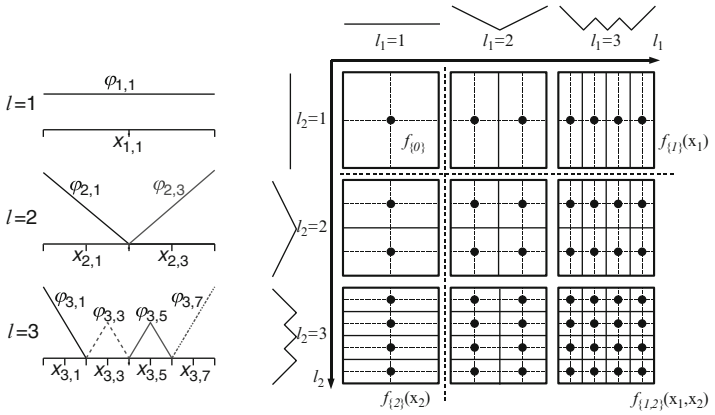


Fig. 3 The classical one-dimensional hierarchical hat basis functions with boundary basis functions on level 0, *dashed* (*left*) and the modified basis functions (*right*)

$$\varphi_{l,i}(x) := \begin{cases} 1 & \text{if } l = 1 \wedge i = 1\,, \\ \left\{ \begin{array}{ll} 2 - 2^l \cdot x & \text{if } x \in \left[0, \frac{1}{2^{l-1}}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 1\,, \\ \left\{ \begin{array}{ll} 2^l \cdot x + 1 - i & \text{if } x \in \left[1 - \frac{1}{2^{l-1}}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 2^l - 1\,, \\ \varphi\left(x \cdot 2^l - i\right) & \text{else} \end{cases}$$

for the one-dimensional basis functions, extrapolating linearly towards the boundary [6, 20, 21], see Fig. 3 (left). This allows to start with only $O(d)$ basis functions (one center point and one pair of points for probing in each coordinate direction); a suitable refinement will then create grid points where it is really necessary. This approach can be similarly applied to other types of basis functions, such as piecewise polynomial ones, B-splines or (pre-)wavelets, see [21] for details.

There is an additional advantage of starting with a constant basis function on the first level. This allows for anchored ANOVA-style decompositions, representing a d-dimensional function f as

$$f(x_1 \ldots x_d) = f_{\{0\}} + \sum_i f_{\{i\}}(x_i) + \sum_{i<j} f_{\{i,j\}}(x_i, x_j) + \ldots + f_{\{1,\ldots,d\}}(x_1 \ldots x_d)\,,$$

anchored at the grid point $\mathbf{x_{1,1}}$ on the first level. Figure 3 (right) illustrates the decomposition in terms of subspaces in two dimensions. If it can be identified which ANOVA components are important for a certain problem, then refinement can be restricted to create only grid points belonging to the respective subspaces. The identification could be analytically, or even empirically by sampling the function on a sparse grid with sufficiently high level, identifying the ANOVA components

based on the sizes of their surplusses, and then coarsening the grid by omitting all grid points belonging to unimportant ANOVA components.

5 Examples and Strategies

As several of our examples focus on the approximation of high-dimensional functions based on noise-prone data sets, we start with a description of sparse-grid-based data mining. The other examples do not require an extra introduction. Please note, that our aim is not to cover the examples in detail and to full extent, but rather to describe approaches to and strategies for adaptive refinement which have shown to be of relevance in plenty of settings.

Two prominent and related tasks in data mining are classification and regression, both of which aim to generalize from known data to predict a target property for new, previously unknown data. Thus, we start with a set S of m data points $\mathbf{x}_j \in \mathbb{R}^d$ of which we know some target value $y_j \in K$ in the d-dimensional feature space,

$$S = \left\{(\mathbf{x}_j, y_j) \in \mathbb{R}^d \times K\right\}_{j=1,\ldots,m} .$$

To be able to learn and generalize, we assume that we obtained S by a random and noise-prone sampling of an unknown function f which we aim to reconstruct. This function will then allow us to predict a target value at new locations $\mathbf{x}$. As we are dealing with finite data, we can scale in the following to the domain $[0, 1]^d$.

For the task of binary classification, think of a bank discriminating potential customers into creditworthy and non-creditworthy, or a production facility predicting the faultiness of products based on standard measurements. Thus, we use two distinct target values, $K = \{-1, +1\}$. Nevertheless, we learn a continuous function f and then check whether the function value is negative or not. This way, the absolute function value provides a measure of confidence in our prediction.

For the task of regression, arbitrary real values are allowed, $K = \mathbb{R}$. An example, which we will use later on, is the prediction of a certain physical property of galaxies which is costly and difficult to obtain. Thus, learning f from the observations that have already been obtained with a lot of effort allows us to predict this property for new galaxies.

We restrict ourselves to reconstructions f_N of f in some sparse grid space $V_n^{(1)}$. To obtain a unique f_N and to be able to deal with noise, we solve the regularized least squares problem

$$f_N \stackrel{!}{=} \operatorname*{arg\,min}_{f_N \in V_N} \left(\frac{1}{m} \sum_{j=1}^{m} \left(y_j - f_N(\mathbf{x}_j)\right)^2 + \lambda \sum_{k=1}^{N} \alpha_k^2 \right) , \tag{3}$$

see [14, 21] and the references cited therein for further details. On the one hand, we ensure closeness to our training data, minimizing the mean square error (MSE) on it. On the other hand, we incorporate the smoothness assumption in data mining, which states that close data points are very likely to have a similar function value, by enforcing some degree of smoothness of f_N and preventing oscillations. Note that working in the hierarchical basis allows us to use an unconventional regularization functional: in the piecewise linear hierarchical setting without grid points on the boundary, $\sum_{k=1}^{N} \alpha_k^2$ corresponds to the squared Sobolev norm $\| f \|_{H^1_{\text{mix}}}$ for normalized basis functions $\tilde{\varphi}_{\mathbf{l},\mathbf{i}}(\mathbf{x}) := \sqrt{2^{-|\mathbf{l}|_1 - d}}\varphi_{\mathbf{l},\mathbf{i}}(\mathbf{x})$ and is thus well-suited for our choice of basis functions [21], i.e.,

$$\| f \|^2_{H^1_{\text{mix}}} := \| \frac{\partial^d}{\partial x_1, \ldots, \partial x_d} f \|^2_{L^2} = \sum_{k=1}^{N} \alpha_k^2 \,.$$

The trade-off between error and smoothness can be influenced by a good choice of the regularization operator λ; for a given data set this can be achieved via cross-validation [2], for example. Note that we follow a general approach: other classification methods can be formulated the same way choosing different error and smoothness terms [9].

Minimizing (3), we obtain a system of linear equations

$$\left(\frac{1}{m} B B^T + \lambda I \right) \boldsymbol{\alpha} = \frac{1}{m} B \mathbf{y} \,, \tag{4}$$

the solution of which is the coefficient vector $\boldsymbol{\alpha}$ of f_N. The identity matrix I stems from the smoothness term; the $N \times m$ and $m \times N$ matrices B and B^T, $b_{ij} = \varphi_i(\mathbf{x}_j)$, and the vector $\mathbf{y}$ of the target values y_i from the error term.

The main advantage of the mesh-based sparse grid approach is that the resulting algorithms scale only linearly in the number of training data points—in contrast to most classical, data-centered approaches which scale typically at least quadratically or even worse. Thus, almost arbitrary amounts of data can be dealt with.

5.1 Straightforward Adaptive Refinement

The first example demonstrates that mere dimensional adaptivity, refining and adding whole subspaces, can be insufficient: in contrast to what is known from the solution of PDEs, spending too many grid points in wrong regions can lead to a higher overall error, as this allows the function to adapt too well to the noise in the training data (overfitting). The example is a two-dimensional artificial classification task, taken from [23]. It consists of two data sets, one for training and one for testing, and has been constructed to comprise 8 % of error. It shows typical characteristics of real-world data sets, as it is neither linearly separable nor very complicated.

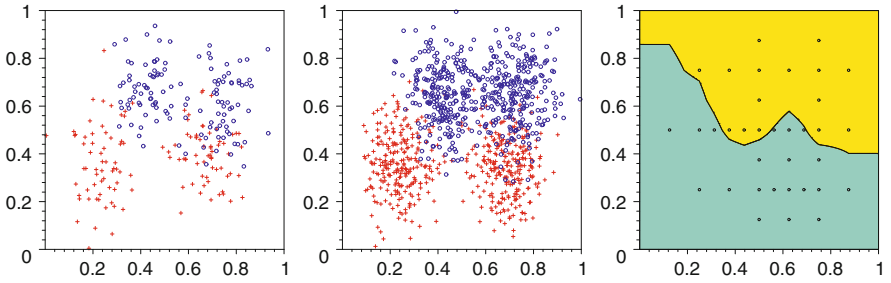


Fig. 4 Ripley data set: 250 data points for training (*left*), 1,000 to test on (*middle*), and the classification areas together with the corresponding sparse grid (*right*)

Figure 4 shows the two data sets as well as the best separation manifold obtained by an adaptive sparse grid. Already eight refinement steps are enough to obtain an excellent accuracy of 91.5 % on the test data—out of a maximum of 92 %.

It can be seen that the separation boundary is resolved better in the central, critical region than in regions where very little information (data points) about the underlying function is given. This is due to the adaptive refinement. Interestingly, after only eight refinements, overfitting starts to take over and the accuracy deteriorates. For this data set a mere dimension-adaptive refinement leads to a lower accuracy: whereas more grid points towards the center of the feature space are necessary to improve, spending the same discretization level on the whole horizontal main axis of the grid leads to local overfitting towards the boundary and to higher errors.

5.2 *Criterion for Adaptive Refinement and Dimensional Adaptivity*

The second example is a classical, artificial 10-dimensional data set, the Friedman1 data set, obtained by a random sampling of an analytical function, enriched by normally distributed noise [10]:

$$f(x_1, \ldots, x_{10}) = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon\,. \qquad (5)$$

All ten variables $x_1, \ldots, x_{10}$ are in $[0, 1]$, thus we do not have to normalize this data set. The function value depends only on the first five variables, the other five variables serve as noise. The additional noise term ϵ is normally distributed, $\mathcal{N}(0, 1)$. We generate data sets uniformly distributed in Ω with 90,000 data points for training and 10,000 each for validation and testing.

As we are dealing with large data sets, the choice of the regularization parameter λ is uncritical, and the focus can be directly put on minimizing the MSE to reduce

Table 1 MSE for the Friedman1 data set for different approaches

	Sparse grids					
Data set	Regular	Adaptive	Opticom	Opticom-dim-adapt	SVM	MARS
Friedman1	0.990	0.976	1.340	1.035	1.148	1.205

the number of grid points further. We therefore modify our criterion for adaptive refinement to target the contribution of a basis function to the squared error. At each point $\mathbf{x}_i$ of the training data, the local error can be computed as

$$e_i = y_i - f_N(\mathbf{x}_i) = y_i - \sum_{j=1}^{N} \varphi_j(\mathbf{x}_i)\alpha_j \,. \tag{6}$$

The contribution of a certain basis function φ_j to $f_N(\mathbf{x}_i)$ and thus to e_i depends both on its function value $\varphi_j(\mathbf{x}_i)$ and its coefficient α_j. To preserve a feasible computational complexity, we presume

$$|\varphi_j(\mathbf{x}_i)\alpha_j e_i^2| \tag{7}$$

as a simplified measure for φ_j's share in the squared local error e_i^2. Accumulating them for all $\mathbf{x}_i$ results in the total contribution

$$c_j := \sum_{i=1}^{m} \left|\varphi_j(\mathbf{x}_i)\alpha_j e_i^2\right| \tag{8}$$

of basis function φ_j. Note that already the mere surplus-based criterion works very well.

Slight additional improvements can be made in terms of the number of grid points that are required, by selecting a suitable refinement strategy, identifying and restricting to the relevant ANOVA terms, and choosing the right basis functions (piecewise polynomials or B-splines); see [21]. Using a good choice, we obtain excellent results, outperforming other techniques [12] and just being matched by the dimensionally adaptive combination technique [13], see Table 1.

Of course, we are dealing with an idealistic setting here: the points to train on are uniformly distributed on the whole domain, and we have plenty of information about the underlying function due to the number of training data. Regular and adaptive sparse grids match each other, apart from the number of grid points (and thus the computational effort that has to be spent).

Furthermore, this example nicely shows that spatially adaptive refinement provides dimensional adaptivity for free. Let $P_{i,j} : [0,1]^d \to [0,1]^2$, $(x_1, \ldots, x_d) \mapsto (x_i, x_j)$ denote the projections of grid points onto the coordinate planes. It can be clearly seen in Fig. 5 that most grid points are spent in the joint dimensions x_1 and x_2 as they correlate most. In contrast, no grid points but those on the second level are wasted in the irrelevant dimensions $x_5, \ldots, x_{10}$: the grid points on the second level are required to probe in these dimensions.

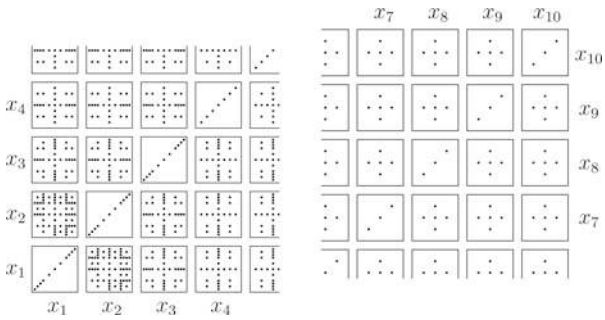


Fig. 5 Grid projections for the first and last four dimensions each. Most correlation can be observed for x_1 and x_2, whereas no additional grid points are spent in the irrelevant dimensions

5.3 Strategies for Adaptive Refinement

Our third example demonstrates different strategies for refinement. It is a real-world data set from astrophysics, targeting the estimation of the redshift of galaxies. The cosmological redshift of a galaxy is related to its distance from earth, see [21] for details. Whereas spectroscopic measurements provide accurate data but are difficult to obtain, estimates based on photometric data are cheap but usually less accurate, leading to a higher amount of uncertainty in the data collected. This raises the question whether it is possible to estimate the redshift of galaxies using photometric measurements, and how well one can generalize from known data (both photometric and spectroscopic).

The Sloan Digital Sky Survey data base in its release 5 [1] provides data for more than 430,000 galaxies, out of which we take 60,000 for testing. This time, the data is not uniformly distributed, but rather on the joint diagonal of the (normalized) domain. Therefore, the use of a suitable strategy of how many grid points are to be refined per refinement step is of importance.

This results in different numbers of grid points per refinement step, see Fig. 6. Refining a certain ratio of (refinable) grid points leads to an exponential increase in the grid size. On the other hand, refining a low, fixed number of grid points per step results in almost a linear increase in the number of grid points. Note that we employ the same surplus criterion for adaptive refinement as before.

The different strategies effect the behavior of the error. To examine this, we train for different choices on a subset of 60,000 data points. Figure 7 shows the MSE on the test data. It can be seen that the choice of the amount of grid points to refine per step significantly impacts the performance. Refining too many grid points at once, the problem cannot be explored well enough before overfitting starts. The four strategies which are refining a certain percentage of grid points at once exhibit the highest minimal error rates. On the other hand, refining too few grid points per step results in an adaptivity which is too greedy. The best results are achieved refining constantly 100 or 200 grid points per refinement step.

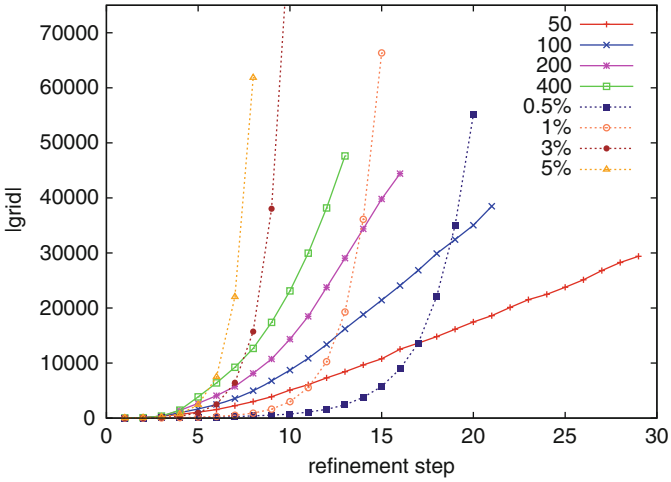


Fig. 6 The growth of the number of grid points for different strategies of how many grid points are to be refined per refinement step

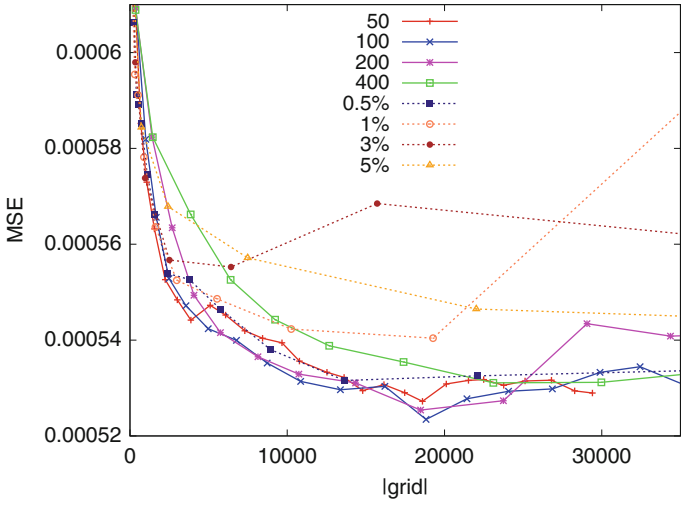


Fig. 7 MSE on the test data for different strategies of how many grid points are to be refined per refinement step. It can be seen that obtaining a low error depends on a suitable choice

The choice of a good trade-off between greedy and broad refinement depends, of course, on the data to train on. Working on huge amounts of data this can be determined, as in the example above, for a smaller validation subset. If we train on the whole data set, other methods can be outperformed, see Table 2 for the root

Table 2 Comparison of the root mean square error for photometric redshift estimation, obtained from several studies on different versions of the SDSS data set

Method	σ_{rms}
CWW [8]	0.0666
Bruzual-Charlot [8]	0.0552
Interpolated spectra [8]	0.0451
1[rst]-nearest neighbors [8]	0.0365
ClassX [24]	0.0340
Polynomial fit [8]	0.0318
SVMs [26]	0.027
Kd-tree [8]	0.0254
Adaptive sparse grids	0.0220

mean square error (RMSE). This is mainly due to the fact that conventional, data-centered methods have to restrict themselves to smaller subsets of the data. Note that the whole data set can really be dealt with in reasonable time if the whole power of current commodity computers is exploited; see [17] for the efficient parallelization on a hybrid system with both multi cores and GPUs.

5.4 Choice of Basis Functions

The next example requires a special choice of basis functions. The task is to determine the price of an early exercise option on d stocks S_i. The holder has to determine at any early exercise time whether to exercise the option or not. He or she will exercise if the current payoff is higher than what can be expected if further holding onto the option, so if

$$P(\mathbf{S}(t_i), t_i) \geq \mathbb{E}\left[V(\mathbf{S}(t_{i+1}), t_{i+1}) \mid \mathbf{S}(t_i), t_i\right] ,.$$

Typically, nested Monte Carlo (MC) simulations are performed to simulate the option and to determine the expectation value for each simulation and each time step. Fortunately, the computationally expensive nesting can be avoided by a least squares Monte Carlo simulation [7], obtaining the expectation values for all simulations at each time-step by solving a regression problem; see [21] for details.

This setting is challenging for sparse grids as the data, obtained by MC simulations, is noisy, clustered, and only little information is available towards the domain's boundary, see Fig. 8 for a two-dimensional example. This does not fit well with the local support of the basis functions, leading to wiggles at the boundary. Furthermore, it is desirable in the context of option pricing to obtain functions which are continuously differentiable and which have zero curvature (second derivatives) at the boundary.

This task can successfully be solved using modified basis functions which adapt to the problem very well. Here, B-splines of a certain degree p are employed, see Fig. 9 (left). They are then modified towards the boundary (right) to avoid to

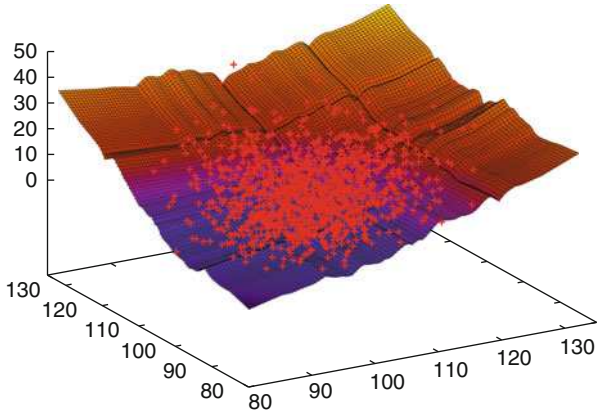


Fig. 8 Data set obtained by MC simulations for a two-dimensional basket option

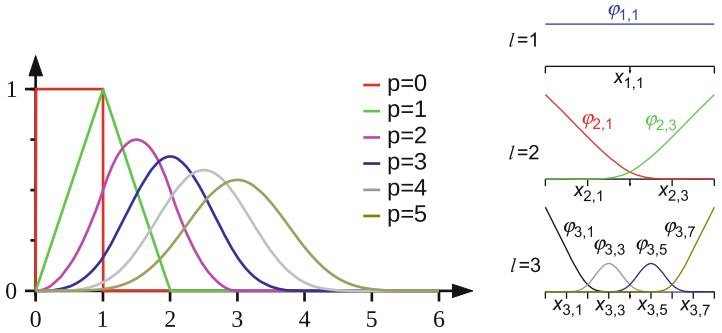


Fig. 9 B-spline construction (*left*) and modified B-splines of degree $p = 3$ (*right*)

spend unnecessary degrees of freedom where little or no information is available. Additionally, the boundary modification ensures zero curvature at the boundary. Furthermore, using B-splines of degree p guarantees $p - 1$ times continuously differentiable functions. The disadvantage of this choice is that basis functions within the same level overlap. Thus, sparse grid algorithms such as function evaluations become much more expensive.

Figure 10 shows the regressed function for a basket of two stocks using a sparse grid of level 6 and B-spline basis functions of degree 3 and 7. Whereas for $p = 3$ still some artifacts can be observed, the function for $p = 7$ is nearly symmetric (the parameters for both stocks S_1 and S_2 are the same) and expresses the training data very well. Note that it shows a reasonable behavior even in regions where there are no training data points at all, due to the less local support of the basis functions. For high degrees, only few typical sparse grid artifacts can be observed. In contrast, often narrow bumps or dips occur for small degrees, which are caused by long basis functions.

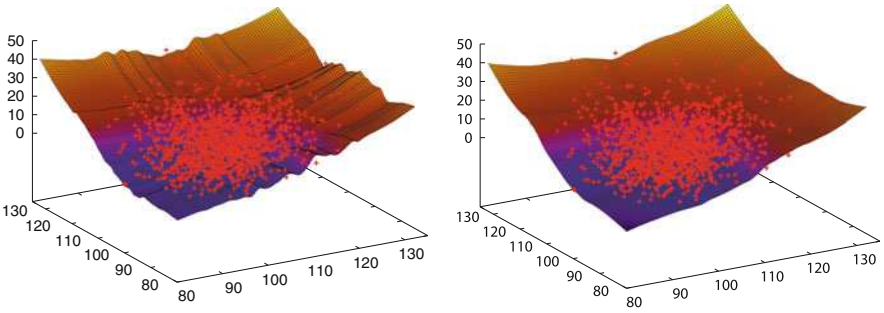


Fig. 10 Regressed function for B-splines of degree 3 (*left*) and 7 (*right*)

If the dynamics of the underlying stocks do not exhibit too high variance, then excellent numerical results can be obtained for up to 8-dimensional options; see [21] for exact results and comparisons. Note that the choice of the basis fulfills additionally the application's requirements to have zero curvature at the boundary and several times continuously differential functions.

5.5 Weighted/Guided Refinement

Finally, we point out how one might guide refinement, depending on the requirements of the application. We take an example from plasma physics, where sparse grids come into play to speed up parameter scans. Several parameters of a gyrokinetic model are to be optimized; consider the minimization of the overall energy to run a fusion plant where the energy depends on the choice of parameters, as an example. As a side-constraint, we require the particle transport in the fusion plant to be zero,

$$\Gamma(\omega_n^{\text{ion}}, \omega_t^{\text{elec.}}, \omega_t^{\text{ion}}, \text{temp}^{\text{elec.}}, \ldots) = 0\,.$$

The optimization can be costly if an iterative method requires to evaluate the function Γ multiple times to approximate gradients, each function evaluation requiring a whole simulation run. The idea is thus to compute a surrogate of Γ, a multi-dimensional sparse grid interpolant, which can then replace the simulation code during optimization. The construction of the surrogate can be done well in advance in an offline stage and thus consume a higher computational effort than a single run of the optimization algorithm. For illustration purposes, we restrict ourselves in the following to the two-dimensional setting.

The function itself is rather smooth, with one continuous kink starting in the middle of the domain and extending to the boundary in one direction, see Fig. 11 (left). Measuring the MSE and maximum error on 100,000 uniformly distributed points in the scaled domain, we can observe that there is almost no difference

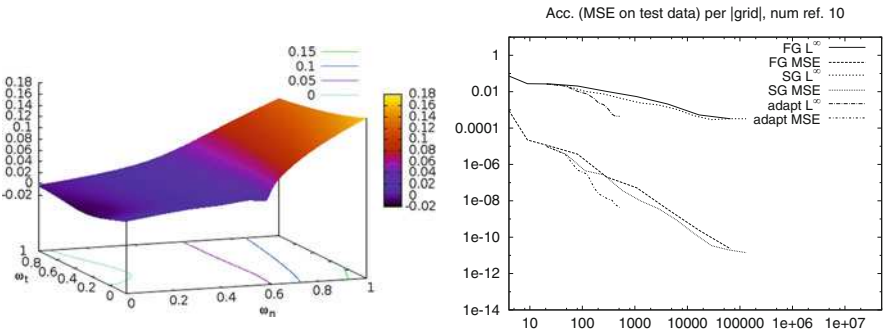


Fig. 11 The underlying function (*left*), and both the MSE on the test data and the maximum error measured for full grids, regular sparse grids and the standard refinement (*right*)

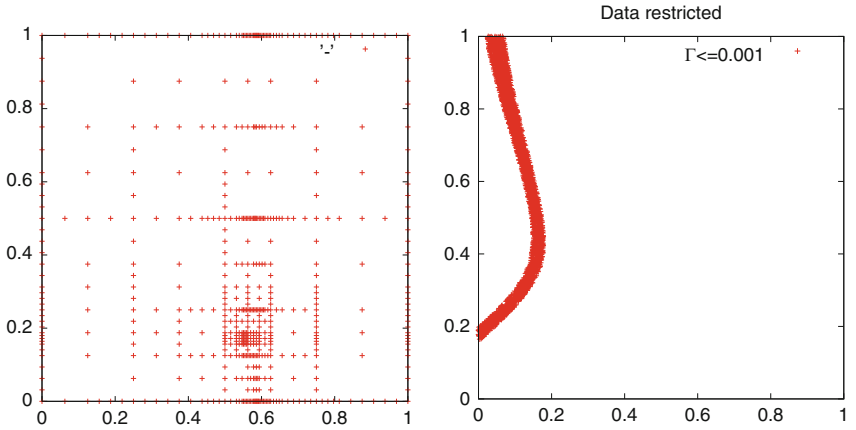


Fig. 12 The sparse grid with standard refinement (*left*), and the region of interest around $\Gamma = 0$ (*right*)

between full grids and regular sparse grids, Fig. 11 (right). Here, we have been employing sparse grids with grid points on the boundary.

If we additionally employ surplus-based adaptive refinement, we obtain a significant improvement, and the error is converging faster (same figure for the first 500 grid points). This is already better, but not good enough. The grid in Fig. 12 (left) shows that adaptivity spends most grid points towards the kink. Reminding that the optimization problem aims for $\Gamma = 0$, it is apparent that it is not the best strategy to aim for a low error in the whole domain. Restricting our data set to $|\Gamma| \leq \epsilon$, $\epsilon = 0.001$, it can be observed that the region of interest is in another part of the domain, see Fig. 12 (right). The refinement should thus be guided to express the region around $\Gamma = 0$.

This can be achieved by weighting the criterion for adaptive refinement. Rather than a mere surplus-based refinement, we take

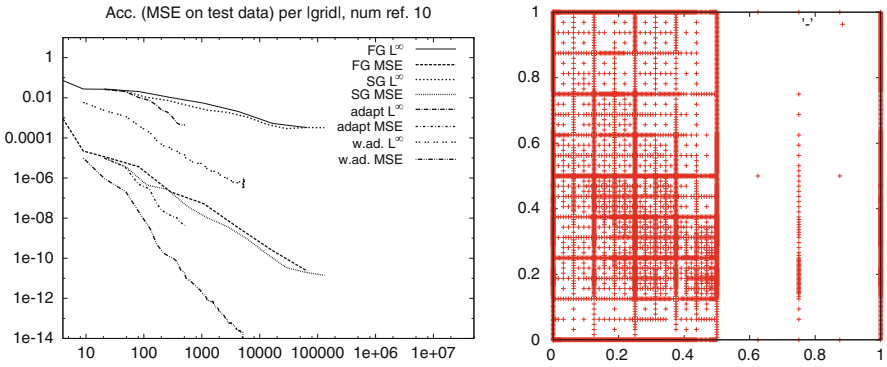


Fig. 13 The MSE on the test data, extended by the weighted adaptivity (*left*), and the corresponding sparse grid (*right*)

$$|\alpha_{\mathbf{l},\mathbf{i}}| \exp\left(-c f(\mathbf{x}_{\mathbf{l},\mathbf{i}})\right)$$

into consideration. This puts less emphasis on grid points with a function value far away from $\Gamma = 0$. Figure 13 shows that the resulting grid is guided to the region of interest. Furthermore, measuring the error only in the region $|\Gamma| \leq \epsilon$, several orders of magnitude in convergence can be gained compared to the previous, straightforward approach. Needing only around 5,000 grid points, the machine accuracy is reached. Note that the function is only coarsely interpolated far away from $\Gamma = 0$, but definitely well enough so that optimization algorithms can get quickly to the region of interest.

We have successfully applied a similar guided refinement in computational finance to compute the PDE solution of the Black-Scholes equation in option pricing. There, the main focus lies on a a good solution at a certain point in the d-dimensional space. Thus, the surplusses are multiplied with a Gaussian weight depending on the distance to the point of interest. As a diffusive part leads to increased smoothness in time, additionally a coarsening step is employed after each time step, removing all grid points with a surplus lower than a certain threshold, see [5] for further details.

6 Conclusions

Whereas sparse grids already enable to deal with higher-dimensional settings than possible for classical mesh-based methods, adaptive refinement is often crucial. It allows to tackle problems that do not meet the smoothness requirements of the sparse grid approach and problems that require to spend as few grid points as possible. This requires to adapt the standard approach to the problem at hand.

We have shown criteria and strategies for adaptive refinement that have proved to be useful in several occasions. They have been illustrated for different problems, both real-world and artificial ones.

Whereas the hierarchical surplus of a grid point can be used as a cheap, free and effective criterion for adaptive refinement, often more considerations have to be spent to reduce the number of grid points even further, e.g.:

- The choice of basis functions has to be adapted in high dimensionalities or where the problem imposes requirements at the sparse grid function itself. Especially the boundary treatment plays an important role.
- The number or percentage of grid points that are to be refined at once determines whether a greedy or a broad refinement occurs. This is especially critical in settings where noise requires not to spend too many grid points in the wrong place.
- If knowledge about the problem is available, weighted (or guided) refinement can significantly improve the results. This allows to encourage refinement in regions of interest. Similarly, refinement can be restricted to important ANOVA components, and coarsening can be employed to get rid of superfluous grid points.

Finally it has to be noted that counter examples can be found for each and every criterion for adaptive refinement. While there is no one-size-fits-all strategy, the simple surplus-based approach frequently gives a good start and can then be adapted to the task at hand. In any case, knowledge about the problem should be incorporated into the criteria and strategies for adaptivity wherever possible.

References

1. J. K. Adelman-McCarthy et al. The fifth data release of the Sloan Digital Sky Survey. *ApJS*, 172:634–644, 2007.
2. D. M. Allen. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16(1):125–127, 1974.
3. J. Benk, H.-J. Bungartz, A.-E. Nagy, and S. Schraufstetter. An option pricing framework based on theta-calculus and sparse grids. In *Progress in Industrial Mathematics at ECMI 2010*, 2010.
4. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.
5. H.-J. Bungartz, A. Heinecke, D. Pflüger, and S. Schraufstetter. Option pricing with a direct adaptive sparse grid approach. *Journal of Computational and Applied Mathematics*, 2011.
6. H.-J. Bungartz, D. Pflüger, and S. Zimmer. Adaptive sparse grid techniques for data mining. In H.G. Bock, E. Kostina, X.P. Hoang, and R. Rannacher, editors, *Modelling, Simulation and Optimization of Complex Processes, Proceedings of the High Performance Scientific Computing 2006, Hanoi, Vietnam*, pages 121–130. Springer, 2008.
7. J. F. Carrière. Valuation of early-exercise price of options using simulation and nonparametric regression. *Insurance: Mathematics and Economics*, 19:19–30, 1996.
8. I. Csabai et al. The application of photometric redshifts to the SDSS Early Data Release. *Astron. J.*, 125:580–592, 2003.
9. T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In *Advances in Computational Mathematics*, pages 1–50. MIT Press, 2000.

10. J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19, 1991.
11. B. Ganapathysubramanian and N. Zabaras. Sparse grid collocation schemes for stochastic natural convection problems. *J. Comput. Phys.*, 225(1):652–685, 2007.
12. J. Garcke. Regression with the optimised combination technique. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 321–328, New York, NY, USA, 2006. ACM Press.
13. J. Garcke. A dimension adaptive sparse grid combination technique for machine learning. In Wayne Read, Jay W. Larson, and A. J. Roberts, editors, *Proceedings of the 13th Biennial Computational Techniques and Applications Conference, CTAC-2006*, volume 48 of *ANZIAM J.*, pages C725–C740, 2007.
14. J. Garcke, M. Griebel, and M. Thess. Data mining with sparse grids. *Computing*, 67(3):225–253, 2001.
15. J. Garcke and M. Hegland. Fitting multidimensional data using gradient penalties and the sparse grid combination technique. *Computing*, 84(1–2):1–25, 2009.
16. M. Hegland. Adaptive sparse grids. In K. Burrage and Roger B. Sidje, editors, *Proc. of 10th Computational Techniques and Applications Conference CTAC-2001*, volume 44, pages C335–C353, 2003.
17. A. Heinecke and D. Pflüger. Multi- and many-core data mining with adaptive sparse grids. In *Proceedings of the 8th ACM International Conference on Computing Frontiers*, pages 29:1–29:10, New York, USA, May 2011. ACM Press.
18. M. Holtz. *Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance*. Dissertation, Institut für Numerische Simulation, Universität Bonn, 2008.
19. A. Klimke, R. Nunes, and B. Wohlmuth. Fuzzy arithmetic based on dimension-adaptive sparse grids: a case study of a large-scale finite element model under uncertain parameters. *Internat. J. Uncertain. Fuzziness Knowledge-Based Systems*, 14:561–577, 2006.
20. D. Pflüger. Data Mining mit Dünnen Gittern. Diplomarbeit, IPVS, Universität Stuttgart, 2005.
21. D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München, 2010.
22. C. Reisinger and G. Wittum. Efficient hierarchical approximation of high-dimensional option pricing problems. *SIAM J. Scientific Computing*, 29(1):440–458, 2007.
23. B. D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA, 1995.
24. A. A. Suchkov, R. J. Hanisch, and B. Margon. A census of object types and redshift estimates in the SDSS photometric catalog from a trained decision-tree classifier. *Astron. J.*, 130:2439–2452, 2005.
25. T. von Petersdorff and C. Schwab. Sparse finite element methods for operator equations with stochastic data. *Appl. Math.*, 51(2):145–180, 2006.
26. Y. Wadadekar. Estimating photometric redshifts using support vector machines. *Publications of the Astronomical Society of the Pacific*, 117:79, 2005.
27. G. Widmer, R. Hiptmair, and C. Schwab. Sparse adaptive finite elements for radiative transfer. *Journal of Computational Physics*, 227:6071–6105, 2008.
28. C. Zenger. Sparse grids. In Wolfgang Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations*, volume 31 of *Notes on Numerical Fluid Mechanics*, pages 241–251. Vieweg, 1991.

Asymptotic Expansion Around Principal Components and the Complexity of Dimension Adaptive Algorithms

Christoph Reisinger

Abstract In this short article, we describe how the correlation of typical diffusion processes arising e.g. in financial modelling can be exploited—by means of asymptotic analysis of principal components—to make Feynman-Kac PDEs of high dimension computationally tractable. We explore the links to dimension adaptive sparse grids (Gerstner and Griebel, Computing 71:65–87, 2003), anchored ANOVA decompositions and dimension-wise integration (Griebel and Holtz, J Complexity 26:455–489, 2010), and the embedding in infinite-dimensional weighted spaces (Sloan and Woźniakowski, J Complexity 14:1–33, 1998). The approach is shown to give sufficient accuracy for the valuation of index options in practice. These numerical findings are backed up by a complexity analysis that explains the independence of the computational effort of the dimension in relevant parameter regimes.

1 Introduction

The motivation for this research comes from financial engineering. Often, the value of financial derivatives is conveniently modelled by partial differential equations. This can lead to highly efficient numerical finite difference and finite element schemes in low to moderate dimensions, however it presents extreme numerical challenges if the dimension is high. There are arguably two main origins: In the first class of applications, the high-dimensionality arises because the financial derivative depends on the whole path of a stock (see e.g. [2]); in the second class, the derivative depends on the value of multiple stocks at a fixed future time. From the PDE perspective, these cases are somewhat different and we focus on the latter category here.

C. Reisinger (✉)
Mathematical Institute, University of Oxford, Oxford, UK
e-mail: christoph.reisinger@maths.ox.ac.uk

J. Garcke and M. Griebel (eds.), *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering 88, DOI 10.1007/978-3-642-31703-3_13,

Consider stocks $S_1, \ldots, S_d$, which are modelled in a standard Black-Scholes setting as geometric Brownian motions with covariance matrix $\Sigma = (\rho_{ij}\sigma_i\sigma_j)_{1\le i,j\le d}$, where σ_i are the volatilities and ρ_{ij} the correlations.

As a running example, consider further a basket $\sum_{i=1}^d \mu_i S_i$ with weights $\mu_i \ge 0$, and a European put option on this basket with strike K and expiry T. The value of this option satisfies the Black-Scholes equation

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} + r\sum_{i=1}^{d} S_i \frac{\partial V}{\partial S_i} - rV = 0, \tag{1}$$

for all $S_i > 0, t \in [0, T]$, supplemented with a terminal condition

$$V(S, T) = P(S) := \max\left(K - \sum\nolimits_{i=1}^{d} \mu_i S_i, 0\right), \tag{2}$$

where $S = (S_1, \ldots, S_d) \in \mathbb{R}^d$ and P is the payoff.

The Black-Scholes equation (1) has the special property that it can be transformed to the (forward) heat equation

$$\frac{\partial u}{\partial t} = \frac{1}{2}\sum_{i=1}^{d} \lambda_i \frac{\partial^2 u}{\partial x_i^2}, \tag{3}$$

for $x \in \mathbb{R}^d$, with appropriate initial data $u(x, 0) = \widetilde{P}(x)$. This is a direct consequence of the assumed log-normality of S_i. Here, λ_i are the eigenvalues of the covariance matrix of the underlying Brownian drivers and are found to decay rapidly in typical applications (see Sect. 3). This opens the possibility of a perturbation analysis.

The computational aspects of such an approach are discussed in [13]; meanwhile, this has been developed further by Hilber et al. [9]. Here, we focus on the underlying expansion itself, and discuss its position within recent work on fundamentally similar ideas. Almost all of this work has been done in the context of high-dimensional cubature problems, and we can relate these approaches to the present context by noting that the solution to (3) can be written as

$$u(x, t) = \frac{1}{t^{d/2}\pi^{d/2}\prod_{i=1}^{d}\lambda_i^{1/2}} \int_{\mathbb{R}^d} \exp\Big(-\sum_{i=1}^{d}(x_i - x_i')^2/(2\lambda_i)\Big)\tilde{P}(x')\,\mathrm{d}x'. \tag{4}$$

Principal component analysis provides a natural ordering of dimensions. Such an ordering can also be found for options whose payoff depends on a Brownian path, through a hierarchical (e.g. Brownian bridge) construction. In both of these settings, the feasibility of the high-dimensional integration problem depends on the speed of

decay of the high-dimensional contributions, as well as their regularity. A natural way to quantify this is via weighted mixed Sobolev norms [15]. We will see later that the λ_i can be directly mapped to those weights.

The upshot is that many applications in finance have a relatively low superposition dimension [16], i.e. their solution can be accurately represented by the sum of solutions to lower-dimensional problems. For integration problems, this can be exploited via quasi-Monte Carlo methods [2, 16], dimension adaptive and generalised sparse grids [3, 7, 8], dimension-wise integration based on anchored ANOVA decomposition [4], or related ideas.

The rest of the article is structured as follows. In Sect. 2, we present results of a simple dimension adaptive sparse grid strategy applied to PDEs of type (1) and (3) to illustrate the dependence of the convergence speed on the choice of coordinates. Section 3 analyses common properties of typical covariance matrices in equity and interest rate markets, and derives an asymptotic expansion around a principal component, which ultimately leads to an ANOVA-type decomposition. Section 4 shows that for exponentially decaying weights of the higher order PCA components, as observed for index options, the complexity can become independent of the nominal dimension. Section 5 critiques the findings and outlines directions for further research.

2 Dimension Adaptive Sparse Grids

In this section, we investigate by numerical experiments the role of the underlying coordinate systems in computations.

2.1 Construction

We explain the construction for the two-dimensional case. It is based on a heuristic algorithm for the approximation of a numerical solution in optimal complexity from computations on suitably chosen Cartesian grids. The candidate set of Cartesian grids is obtained by bisection from a coarse grid, say the unit square $[0, 1]^2$, such that the mesh widths in the two directions are $(h_i, h_j) = (2^{-i}, 2^{-j})$, $(i, j) \in \mathbb{N}_0^2$. We denote the corresponding numerical solution by $U(i, j)$, and have in mind pointwise evaluations of a finite difference solution or other functionals thereof.

To assess the contribution of a particular $U(i, j)$ to the overall solution, we study the *hierarchical surplus* (see e.g. [1])

$$\begin{aligned}\delta U(i,j) &= U(i,j) - U(i-1,j) - U(i,j-1) + U(i-1,j-1)\\ &= \delta_1^- \delta_2^- U(i,j),\end{aligned}$$

where $\delta_1^- U(i,j) = U(i,j) - U(i-1,j)$ and $\delta_2^- U(i,j) = U(i,j) - U(i,j-1)$ for $i, j > 0$ are backward difference operators, and $\delta_1^- U(i,j) = U(i,j)$ if $i = 0$, $\delta_2^- U(i,j) = U(i,j)$ if $j = 0$.

A numerical approximation on level n is defined by the choice of grids described by an index set $\mathcal{M}_n \subset \mathbb{N}_0^2$ via

$$u_n := \sum_{(i,j)\in\mathcal{M}_n} \delta U(i,j).$$

We require that $\mathcal{M}_n$ is convex in the sense that $\mathcal{M}_n = C(\mathcal{M}_n)$, where

$$C(\mathcal{M}_n) = \left\{(i,j) \in \mathbb{N}_0^2 : \exists k \geq j, (i,k) \in \mathcal{M}_n \vee \exists k \geq i, (k,j) \in \mathcal{M}_n\right\}.$$

For a similar construction see [7]. For a convergent discretisation, we expect

$$u_n \to u \quad \text{for} \quad \mathcal{M}_n \uparrow \mathbb{N}_0^2,$$

where the last expression is to be understood in the sense $\mathcal{M}_n \subset \mathcal{M}_{n+1}$ and $\forall (i,j) \in \mathbb{N}_0^2 \; \exists n : \; (i,j) \in \mathcal{M}_n$. An error bound will be given by

$$|u - u_n| \leq \sum_{(i,j)\notin\mathcal{M}_n} |\delta U(i,j)|.$$

The following construction of $\mathcal{M}_n$ is *dimension adaptive* (see [3,7] for a similar strategy). Start with $\mathcal{M}_0 = \{(0,0)\}$. For given $\mathcal{M}_n, n \geq 0$, consider candidate nodes

$$\mathcal{C}_n = \left\{(i,j) \in \mathbb{N}_0^2 : \; (i,j) \notin \mathcal{M}_n, (i-1,j) \in \mathcal{M}_n \vee (i,j-1) \in \mathcal{M}_n\right\}.$$

Then refine

$$\mathcal{M}_{n+1} = C\left(\mathcal{M}_n \cup \{(i,j) \in \mathcal{C}_n : \; |\delta U(i,j)| \geq \gamma \, c_n\}\right),$$

where

$$c_n = \max\{|\delta U(i,j)| : \; (i,j) \in \mathcal{C}_n\}.$$

The idea is that we refine only in those directions where there is a significant contribution from the refined grid. We pick γ slightly smaller than the asymptotic ratio of the surpluses between two refinement levels.

This draws on the theoretical properties of the used discretisation. We use below second order finite difference and finite element schemes. From the analysis of the sparse grid combination technique, see e.g. [11] or [14], we expect

$$\delta U(i,j) \sim c h_i^2 h_j^2. \tag{5}$$

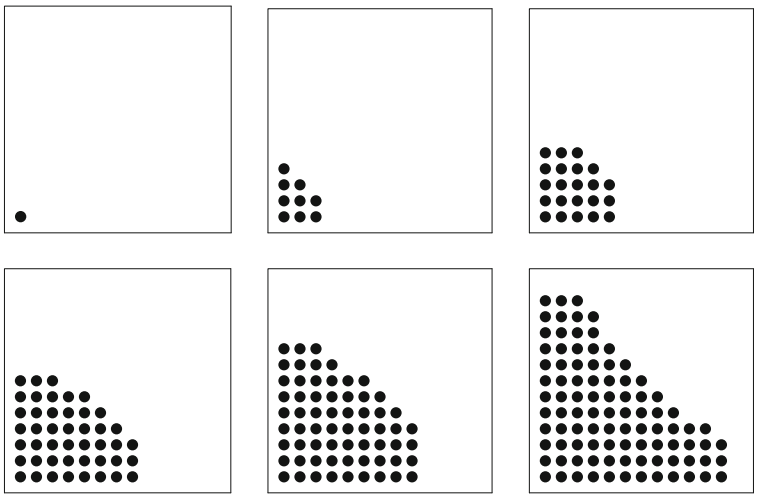


Fig. 1 The *black dots* correspond to indices (i, j) included in $\mathcal{M}_n$ at various refinement step n from $n = 0$, i.e. $i = j = 0$, *top left* box. For this example where none of the two directions is dominant, the refinement is largely symmetric

Refining in one direction, $i \to i+1$ or $j \to j+1$, we expect the surplus to decrease by a factor of $1/4$. Hence we used $\gamma = 0.2$ in the numerical computations below.

2.2 An Example

In the following subsections, we illustrate the behaviour of the above strategy on the example of the Black-Scholes model for a put option on an equity basket consisting of BMW and Daimler. This amounts to solving the PDE (1) for $d = 2$, for parameters $\sigma_1 = 0.438$, $\sigma_2 = 0.616$, $\rho = 0.89$, $r = 0.05$. The terminal condition at $t = T = 1$ is given by (2) with $\mu_1 = 0.384$, $\mu_2 = 0.616$, $K = 0.25$. We are interested in the numerical solution evaluated at $(S_1, S_2) = (K, K)$ at time $t = 0$.

We use a standard central difference scheme with fractional-step θ-timestepping. The domain is truncated at sufficiently large values of S_1 and S_2 and then transformed to the unit square, where appropriate asymptotically exact boundary conditions are set.

Figure 1 illustrates the resulting adaptive refinement strategy. It shows that the grid construction is very similar to that of a standard sparse grid, where $\mathcal{M}_n = \{(i, j) \in \mathbb{N}_0^2 : i + j \leq n\}$, confirming experimentally the optimality of the standard sparse grid [1, 17].

Figure 2 gives an indication of the efficiency of the method. Ideally, grids would be included in decreasing order of importance. It is seen that for large enough n, where the grid sizes are small enough for (5) to be a valid approximation, this is largely the case.

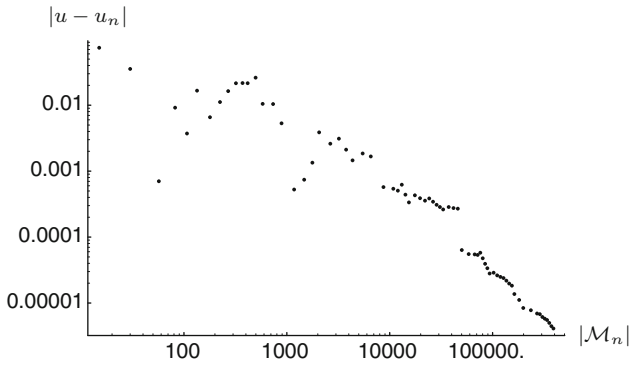


Fig. 2 Denoting by $|\mathcal{M}_n|$ the number of all grid points of all grids in $\mathcal{M}_n$, shown is the approximation error for an increasing number of unknowns. The slope for large n is consistent with asymptotic second order convergence

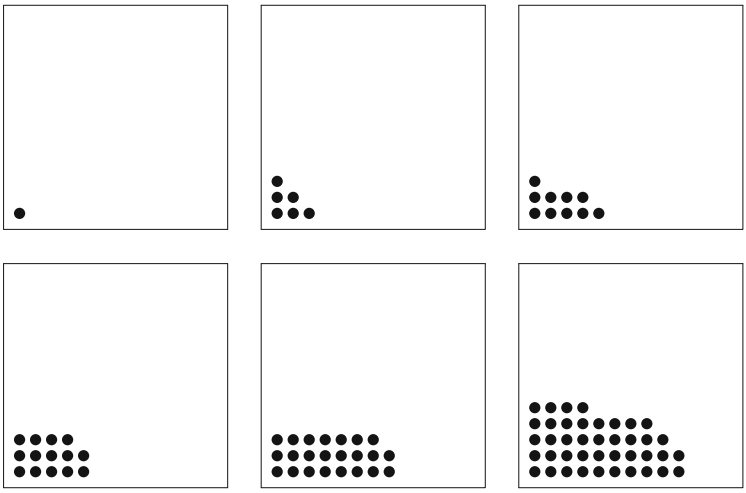


Fig. 3 Similar to Fig. 1, but now in eigenvector coordinates. Refinement is stronger in the direction of the principal component

2.3 Principal Components

Sparse grid solutions, and indeed all approximations based on tensor product spaces, are by construction dependent on the choice of an underlying coordinate system, if the underlying problem is anisotropic. We now seek to exploit the fact that our dimension adaptive method can use this to its advantage. To this end, transform (1)–(3); then, in the above example, one finds $\lambda_1 = 0.431$, $\lambda_2 = 0.024$.

Repeating the above numerical test in transformed coordinates, leads to the refinement strategy in Fig. 3. Reassuringly, the adaptive algorithm detects the

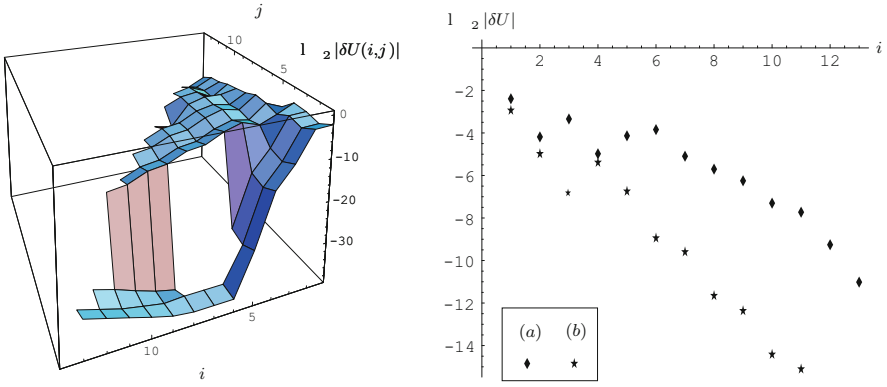


Fig. 4 *Left*: The surplus $U(i, j)$ for different grids (i, j). *Right*: Cross-sections along the edges (a) $\log_2 |\delta U(i, 3)|$, (b) $\log_2 |\delta U(2, i)|$

anisotropy of the operator and refines mostly in the direction where the variation is largest.

Figure 4 provides a different angle on this behaviour. It shows the (logarithm of the) hierarchical surplus for different grids, indexed by (i, j). Revisiting the derivation of the hierarchical surplus for a central finite difference scheme in [14], explicitly taking into account the dependence of the truncation error on the eigenvalues, leads to

$$\delta_1^- U(i, j) \sim c_1 \lambda_1 h_i^2, \tag{6}$$

$$\delta_2^- U(i, j) \sim c_2 \lambda_2 h_j^2, \tag{7}$$

$$\delta U(i, j) \sim c_0 \lambda_1 \lambda_2 h_i^2 h_j^2. \tag{8}$$

This is based on the simplifying assumption that the smoothness of the solution in the two directions does not depend on λ_1 and λ_2, which will be a reasonable assumption for moderate timescales.

The left plot appears to confirm that the surplus is asymptotically a function of $i + j$, as suggested by (8), as long as $i, j > 2$, i.e. excluding cases where the solution is solely or predominantly determined by the boundary values.

The plot on the right, which takes cross-section in the first and second directions, reveals the anisotropy of the problem. Close to the 'edges' of the grid table, i.e. where j or i are small, the surplus will be similar to (6) or (7), respectively. This determines the different 'constant' factors in the surplus in these directions, and can explain the offset of the two curves in Fig. 4 (right). As we will discuss later, the ratio of λ_1/λ_2, here approximately 20, is not the only relevant factor, but also the initial data to the PDE.

An intriguing new phenomenon is observed when we go to three dimensions. When we add Volkswagen to the basket from before, the eigenvalues are estimated

as $\lambda_1 = 0.653$, $\lambda_2 = 0.069$, $\lambda_3 = 0.023$, and we solve the corresponding PDE (3) in $d = 3$ dimensions.

At the finest computed level, 2,197 grids are involved. This compares in complexity roughly to a regular three-dimensional sparse grid on level 20, which consists of 2,023 grids. Out of these 2,197 grids, 469 grids of $\mathcal{M}$ are at the 'boundary' of the grid table described by

$$\mathcal{B} = \left\{(i, j, k) \in \mathbb{N}_0^3 : \; i \cdot j \cdot k = 0\right\},$$

i.e. for which at least one direction is unrefined, while 1,728 grids are in the 'interior' $\mathcal{M}\backslash\mathcal{B}$. However, the contribution from the boundary elements to the solution is

$$\sum_{(i,j,k)\in\mathcal{B}\cap\mathcal{M}} |\delta U(i, j, k)| = 0.4369,$$

while

$$\sum_{(i,j,k)\in\mathcal{M}\backslash\mathcal{B}} |\delta U(i, j, k)| = 0.0035.$$

This suggests that the problem can be well approximated by sums of two-dimensional approximations. We investigate this for general dimensions in the following section.

3 PCA and Asymptotic Expansion

This section explores communalities of the eigensystem of important covariance matrices, and their use for problem-adapted high-dimensional approximation.

A common feature to the examples in Fig. 5 is the presence of a dominant eigenvalue, which is by a factor of 10 or more larger than the rest of the spectrum. This motivates expanding the solution to (3) in these small parameters to obtain approximations to the high-dimensional solution by solving low-dimensional problems. Consider therefore u as a function of $\lambda = (\lambda_i)_{1<i\le d} \in \mathbb{R}^{d-1}$, where

$$\lambda_i \ll \lambda_1, \quad i > 1.$$

Assuming the dependence of u on λ is sufficiently smooth, first order Taylor expansion gives

$$u(\lambda) = u(0) + \sum_{i=2}^{d} \lambda_i \frac{\partial u}{\partial \lambda_i}(0) + O(\|\lambda\|_2^2). \tag{9}$$

For special cases, the eigenvalue sensitivities can be calculated analytically. An example is the value of a put option in the Black-Scholes model, where

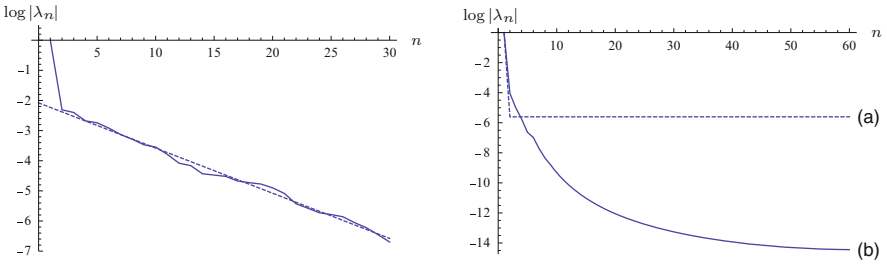


Fig. 5 Shown are the (logarithms of) normalised eigenvalues of three representative examples of covariance matrices from financial modelling. *Left*: Estimated covariances $\sigma_i \sigma_j \rho_{ij}$ from historical time series of the DAX30, i.e. $d = 30$, and a fitted curve (which will be discussed in Sect. 4). *Right*: Two stylized forward rate models, both for $d = 60$: (**a**) constant correlation $\rho = 0.8$ for all tenors, constant volatility; (**b**) humped-shaped volatility $\sigma_i = (A + Bi)\exp(-Ci) + D$ with $A = 0.1$, $B = 0.1$, $C = 1$, $D = 0.1$, and exponentially decaying correlation $\rho_{ij} = \exp(-\alpha|i-j|)$, $\alpha = 0.025$. Eigenvalues are numbered in decreasing order

differentiation of the integral formula (4) gives easily computable closed-form expressions similar to the standard one-factor Black-Scholes formula [12]. The correction terms also satisfy simple PDEs, i.e. for

$$u^{\{i\}} := \frac{\partial u}{\partial \lambda_i}(0), \tag{10}$$

$$\frac{\partial}{\partial t} u^{\{i\}} = \frac{1}{2}\lambda_1 \frac{\partial^2}{\partial x_1^2} u^{\{i\}} + \frac{1}{2}\frac{\partial^2 u}{\partial x_i^2}(0), \tag{11}$$

where the last equation follows from application of (3) to (10).

More generally, the exact sensitivity can be replaced by a finite difference. Writing, by slight abuse of notation, $u(\lambda_i)$ for the solution to the problem where all coefficients except λ_1 and λ_i are zero,

$$\frac{\partial u}{\partial \lambda_i}(0) = \frac{u(\lambda_i) - u(0)}{\lambda_i} + O(\lambda_i).$$

This coincides with the financial industry practice of parameter 'bumping'. Inserting in (9),

$$u(\lambda) = u(0) + \sum_{i=2}^{d} (u(\lambda_i) - u(0)) + O(\|\lambda\|_2^2) \tag{12}$$

is a second order approximation in the small parameters $\lambda_i \leq \lambda_2 \ll \lambda_1$, $i = 2, \ldots, d$.

In a sense, this formula is even 'better' than the one using the exact derivatives, as it captures functions of superposition dimension one *exactly*. We discuss this in more detail below.

Moreover, the lead term $u(0)$ requires the solution of only a one-dimensional PDE in x_1,

$$\frac{\partial u}{\partial t}(0) = \frac{1}{2}\lambda_1 \frac{\partial^2 u}{\partial x_1^2}(0),$$

and the correction terms $u(\lambda_i)$ the solution of $d-1$ two-dimensional PDEs in x_1 and x_i,

$$\frac{\partial u}{\partial t}(\lambda_i) = \frac{1}{2}\lambda_1 \frac{\partial^2 u}{\partial x_1^2}(\lambda_i) + \frac{1}{2}\lambda_i \frac{\partial^2 u}{\partial x_i^2}(\lambda_i).$$

For the test example of the DAX given above, [13] report an error against a Monte Carlo simulation benchmark of <0.06 % of the option value, with an absolute error of 0.000073 at the strike $K = 1$, which is less than 1 basis point. Results for other (and smaller) baskets are comparable, such that the approximation seems sufficient for practical applications.

What is more, it is not difficult to extend this expansion to higher order, e.g. to order two and using an approximation to the cross-derivatives in λ_i and λ_j,

$$u(\lambda) = u(0) + \sum_{i=2}^{d}(u(\lambda_i) - u(0)) + \sum_{i\neq j}(u(\lambda_i,\lambda_j) - u(\lambda_i) - u(\lambda_j) + u(0)) + O(\|\lambda\|_2^3),$$

where $u(\lambda_i,\lambda_j)$ is the solution where all coefficients except λ_1, λ_i and λ_j are set zero. Hence, the additional correction terms are the solution of $(d-1)(d-2)/2$ three-dimensional PDEs.

Proceeding in this way to order d,

$$u(\lambda) = u(0) + \sum_{i=2}^{d} \Delta_i u + \sum_{i\neq j} \Delta_i \Delta_j u + \ldots + \Delta_2 \ldots \Delta_d u,$$

where $\Delta_i u = u(\lambda_i) - u(0)$ etc. This is equivalent to an anchored ANOVA decomposition [7]. The functions $u(\lambda_i)$ have the interpretation

$$u(\lambda_i) = \mathbb{E}(\widetilde{P}(X_T)), \tag{13}$$

$$\mathrm{d}X_t^k = \begin{cases} \lambda_k \,\mathrm{d}W_t^k & k \in \{1,i\}, \\ 0 & \text{else}, \end{cases} \tag{14}$$

$$X_0^k = x_k, \tag{15}$$

where W is a standard Brownian motion. Similarly, $u(\lambda_i,\lambda_j)$ corresponds to a process as per (14), but with diffusion in directions $\{1,i,j\}$, with the obvious extension to higher order. Writing these in integral form,

$$u(\lambda_i) = \frac{1}{t\pi\sqrt{\lambda_1\lambda_i}} \int_{\mathbb{R}^2} \exp\left(-(x_1 - x_1')^2/\lambda_1\right) \exp\left(-(x_i - x_i')^2/\lambda_i\right) \widetilde{P}(x')\,\mathrm{d}x_1'\,\mathrm{d}x_i',$$

and similar for higher order terms, leads to dimension-wise quadrature as in [4].

It should be noted that this choice of coordinates does not take into account the initial data of the PDE. It is conceivable to construct a pathological example where the approximation is arbitrarily bad, e.g. where the principal component is parallel to a level curve of the initial data, and orthogonal to a direction of high curvature.

Basket options are a fortunate case in this respect. The direction of steepest change of the basket value, which determines the payoff, is roughly aligned with the principal component of the covariance matrix. This results from positive basket weights, and the fact that the principal eigenvector of a positive matrix has positive entries (guaranteed by the Perron-Frobenius Theorem).

4 A Simple and Some More Complex Complexity Results

Formula (12) suggests that the complexity is linear in the dimension d of the problem, assuming the quadratic remainder term can be neglected, and that the effort for the eigenvalue decomposition and transformation is negligible, which is the case in practice.

We argue in this section that it is not necessary to include all d terms, by exploiting the decay in λ_i further. The empirical evidence which underpins this analysis is already seen in Fig. 5, where the ordered and normalised ($\lambda_1 = 1$) eigenvalues of the covariance matrix are plotted together with a regression of the set excluding the first one, which gives $\lambda_n \approx 0.108 \cdot 0.861^{n-2}$ for $n \geq 2$. There is clear evidence of exponential decay of the eigenvalues, following a big jump of about 90 % of the first eigenvalue.

We therefore assume in the following that $\lambda_i \leq \delta \cdot \lambda^i$ for some $\delta \ll 1$ and $\lambda < 1$, but with $\lambda \approx 1$. In this parameter setting, it is justified to neglect the δ^2 term (we point to the fact that the residual error was negligible in the numerical case study), but a potentially large number of first order terms gives significant contributions if d is large. As in [15], and more recently [6, 10], we embed the problem in an infinite-dimensional space as $d \to \infty$.

All of this, combined with (9), motivates the following limiting case as basis for a simple complexity analysis.

Assumption 4.1. *Assume that*

$$u(\lambda, \delta) = u_0 + \delta \sum_{i=1}^{\infty} \lambda^i u_i, \tag{16}$$

and there is an algorithm which finds u_i with accuracy ϵ in complexity

$$C \leq \epsilon^{-1/p},$$

i.e. p is the order of the method.

We do not specify for this abstract result what the u_i are, but clearly we have in mind first order ANOVA terms. Then, in the above setting, we require the numerical solution of two-dimensional heat equations, so if we use standard linear finite elements and second order time-steping, $p = 2/3$.

Proposition 1. *Under Assumption 4.1, there exists an algorithm, which finds u with accuracy ϵ in complexity*

$$C \leq c \cdot (\epsilon/\delta)^{-1/p} \cdot (1 - \lambda^{1/(1+p)})^{-(1+p)/p},$$

where c does not depend on ϵ, λ, or δ.

Proof. The proof is constructive. Let ϵ_i the accuracy of u_i, then we minimise the sum of the costs $\epsilon_i^{-1/p}$ under the constraint that the total error is $\delta \sum_{i=0}^{\infty} \lambda^i \epsilon_i \leq \epsilon$. A direct application of Lagrangian multipliers gives the result.

This extends straightforwardly to higher order in δ, i.e. if

$$u(\lambda, \delta) = u_0 + \delta \sum_{i=1}^{\infty} \lambda^i u_i + \ldots + \delta^q \sum_{i_1 \neq \ldots \neq i_q} \lambda^{i_1} \cdot \ldots \cdot \lambda^{i_q} u_{i_1,\ldots,i_q}, \tag{17}$$

and the complexity for solving for $u_{i_1,\ldots,i_q}$ is $\epsilon^{-q/p}$, then the total complexity is of order $(\epsilon/\delta^q)^{-q/p}$, although the constant factors become less explicitly computable.

Ultimately, one would want to adaptively pick q to ensure the overall error is below some desired ϵ, see e.g. [4, 6, 10, 15]. An appropriate measure for the overall complexity, in the PDE setting as for cubature, are weighted Sobolev norms.

The size and numerical approximation of $u_{i_1,\ldots,i_q}$ depends on the regularity of the original problem, and, as [5] show, the lower-order ANOVA terms may have higher regularity. We finish this section by outlining how such a result may be derived in the setting of this article, where the correction terms in (17) are essentially determined by the parameter sensitivities

$$u^{\{i_1,\ldots,i_m\}} := \frac{\partial^m u}{\partial \lambda_{i_1} \ldots \partial \lambda_{i_m}}.$$

For the example of the basket option, these are explicitly computable from the integral formula (4), which is carried out for the first order terms in [12].

We mention an alternative route, which is more generally applicable. Repeated differentiation of (11) gives

$$\frac{\partial}{\partial t} u^{\{i_1,\ldots,i_m\}} = \frac{1}{2} \sum_j \lambda_j \frac{\partial^2}{\partial x_j^2} u^{\{i_1,\ldots,i_m\}} + \frac{1}{2} \sum_{k=1}^{m} \frac{\partial^2}{\partial x_{i_k}^2} u^{\{i_1,\ldots,i_m\} \setminus \{i_k\}},$$

which allows the recursive application of standard regularity results for parabolic PDEs. This will give bounds on the sensitivities (ANOVA terms) and corresponding weights for the complexity analysis, and will be the subject of future research.

5 Conclusions and Extensions

The principal component analysis of multivariate diffusion problems not only gives an ordering of the dimensions, but allows a formal Taylor expansion in terms of the spectrum of the covariance matrix, which leads to a decomposition equivalent to an anchored ANOVA decomposition of the solution. Model problems from financial engineering considered in the literature show that already a first order approximation, which corresponds to an approximation with superposition dimension one, is sufficiently accurate for practical use.

The question arises if this strategy works equally well for models beyond the normal (or log-normal) case. In models with non-constant volatilities and drift, "freezing" the coefficients at the initial value of the process reduces the problem to the present case. The additional approximation error will be acceptable for not too long time horizons. Jump models will be the subject of future research, and the methodology should be transferable in spirit. An arguably more challenging extension is to non-linear problems, e.g. free boundary problems arising in American option valuation, but again the author does not foresee any fundamental difficulties.

References

1. Bungartz, H.-J., Griebel, M.: Sparse grids, Acta Numer., **13**, 11–23 (2004)
2. Caflisch, R.E., Morokoff, W., Owen, A.: Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension. J. Comput. Finance, **1**, 27–46 (1997)
3. Gerstner, T., Griebel, M.: Dimension-adaptive tensor-product quadrature. Computing, **71**, 65–87 (2003)
4. Griebel, M., Holtz, M.: Dimension-wise integration of high-dimensional functions with applications to finance. J. Complexity, **26**, 455–489 (2010)
5. Griebel, M., Kuo, F. Y., Sloan, I.H.: The smoothing effect of the ANOVA decomposition. J. Complexity, **26**, 523–551 (2010)
6. Gnewuch, M.: Infinite-dimensional integration on weighted Hilbert spaces, Preprint cucs-016-10, Department of Computer Science, Columbia University (To appear in Math. Comp.) (2011)
7. Gnewuch, M.: Infinite-Dimensional Integration on Weighted Hilbert Spaces. Math. Comp., **81**, 2175–2205 (2012)
8. Hegland, M., Garcke, J., Challis, V.: The combination technique and some generalisations. Lin. Algebra Appl., **420**, 249–275 (2007)
9. Hilber, N., Kehtari, S., Schwab, C., Winter, C.: Wavelet finite element method for option pricing in highdimensional diffusion market models. SAM Research Report 2010-01, ETH Zürich (2010)
10. Niu, B., Hickernell, F.J., Müller-Gronbach, T., Ritter, K.: Deterministic multi-level algorithms for infinite-dimensional integration on $\mathbb{R}^N$. J. Complexity, **27**, 331–351 (2011)

11. Pflaum, C., Zhou, A.: Error analysis of the combination technique. Numer. Math., **84**, 327–350 (1999)
12. Reisinger, C.: Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben. PhD Thesis, Universität Heidelberg (2004)
13. Reisinger, C., Wittum, G.: Efficient hierarchical approximation of high-dimensional option pricing problems. SIAM J. Sci. Comput., **29**, 440–458 (2007)
14. Reisinger, C.: Analysis of linear difference schemes in the sparse grid combination technique. IMA J. of Numer. Anal., doi:10.1093/imanum/drs004 (2012)
15. Sloan, I.H., Woźniakowski, H.: When are quasi-Monte Carlo algorithms efficient for high-dimensional integrals? J. Complexity, **14**, 1–33 (1998)
16. Wang, X., Sloan, I.H.: Why are high-dimensional finance problems often of low effective dimension? SIAM J. Sci. Comput., **27**, 159–183 (2005)
17. Zenger, C.: Sparse grids. In: Hackbusch, W. (ed) Parallel Algorithms for Partial Differential Equations, Notes on Numerical Fluid Mechanics, **31**, Vieweg, Braunschweig/Wiesbaden (1991)

Editorial Policy

1. Volumes in the following three categories will be published in LNCSE:

i) Research monographs
ii) Tutorials
iii) Conference proceedings

Those considering a book which might be suitable for the series are strongly advised to contact the publisher or the series editors at an early stage.

2. Categories i) and ii). Tutorials are lecture notes typically arising via summer schools or similar events, which are used to teach graduate students. These categories will be emphasized by Lecture Notes in Computational Science and Engineering. **Submissions by interdisciplinary teams of authors are encouraged.** The goal is to report new developments – quickly, informally, and in a way that will make them accessible to non-specialists. In the evaluation of submissions timeliness of the work is an important criterion. Texts should be well-rounded, well-written and reasonably self-contained. In most cases the work will contain results of others as well as those of the author(s). In each case the author(s) should provide sufficient motivation, examples, and applications. In this respect, Ph.D. theses will usually be deemed unsuitable for the Lecture Notes series. Proposals for volumes in these categories should be submitted either to one of the series editors or to Springer-Verlag, Heidelberg, and will be refereed. A provisional judgement on the acceptability of a project can be based on partial information about the work: a detailed outline describing the contents of each chapter, the estimated length, a bibliography, and one or two sample chapters – or a first draft. A final decision whether to accept will rest on an evaluation of the completed work which should include

– at least 100 pages of text;
– a table of contents;
– an informative introduction perhaps with some historical remarks which should be accessible to readers unfamiliar with the topic treated;
– a subject index.

3. Category iii). Conference proceedings will be considered for publication provided that they are both of exceptional interest and devoted to a single topic. One (or more) expert participants will act as the scientific editor(s) of the volume. They select the papers which are suitable for inclusion and have them individually refereed as for a journal. Papers not closely related to the central topic are to be excluded. Organizers should contact the Editor for CSE at Springer at the planning stage, see *Addresses* below.

In exceptional cases some other multi-author-volumes may be considered in this category.

4. Only works in English will be considered. For evaluation purposes, manuscripts may be submitted in print or electronic form, in the latter case, preferably as pdf- or zipped ps-files. Authors are requested to use the LaTeX style files available from Springer at http://www.springer.com/authors/book+authors/helpdesk?SGWID=0-1723113-12-971304-0 (Click on Templates → LaTeX → monographs or contributed books).

For categories ii) and iii) we strongly recommend that all contributions in a volume be written in the same LaTeX version, preferably LaTeX2e. Electronic material can be included if appropriate. Please contact the publisher.

Careful preparation of the manuscripts will help keep production time short besides ensuring satisfactory appearance of the finished book in print and online.

5. The following terms and conditions hold. Categories i), ii) and iii):

Authors receive 50 free copies of their book. No royalty is paid.
Volume editors receive a total of 50 free copies of their volume to be shared with authors, but no royalties.

Authors and volume editors are entitled to a discount of 33.3 % on the price of Springer books purchased for their personal use, if ordering directly from Springer.

6. Commitment to publish is made by letter of intent rather than by signing a formal contract. Springer-Verlag secures the copyright for each volume.

Addresses:

Timothy J. Barth
NASA Ames Research Center
NAS Division
Moffett Field, CA 94035, USA
barth@nas.nasa.gov

Michael Griebel
Institut für Numerische Simulation
der Universität Bonn
Wegelerstr. 6
53115 Bonn, Germany
griebel@ins.uni-bonn.de

David E. Keyes
Mathematical and Computer Sciences
and Engineering
King Abdullah University of Science
and Technology
P.O. Box 55455
Jeddah 21534, Saudi Arabia
david.keyes@kaust.edu.sa

and

Department of Applied Physics
and Applied Mathematics
Columbia University
500 W. 120 th Street
New York, NY 10027, USA
kd2112@columbia.edu

Risto M. Nieminen
Department of Applied Physics
Aalto University School of Science
and Technology
00076 Aalto, Finland
risto.nieminen@aalto.fi

Dirk Roose
Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven-Heverlee, Belgium
dirk.roose@cs.kuleuven.be

Tamar Schlick
Department of Chemistry
and Courant Institute
of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012, USA
schlick@nyu.edu

Editor for Computational Science
and Engineering at Springer:
Martin Peters
Springer-Verlag
Mathematics Editorial IV
Tiergartenstrasse 17
69121 Heidelberg, Germany
martin.peters@springer.com

Lecture Notes in Computational Science and Engineering

1. D. Funaro, *Spectral Elements for Transport-Dominated Equations.*
2. H.P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming.
3. W. Hackbusch, G. Wittum (eds.), *Multigrid Methods V.*
4. P. Deuflhard, J. Hermans, B. Leimkuhler, A.E. Mark, S. Reich, R.D. Skeel (eds.), *Computational Molecular Dynamics: Challenges, Methods, Ideas.*
5. D. Kröner, M. Ohlberger, C. Rohde (eds.), *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws.*
6. S. Turek, *Efficient Solvers for Incompressible Flow Problems.* An Algorithmic and Computational Approach.
7. R. von Schwerin, ***M**ulti **B**ody **S**ystem **SIM**ulation.* Numerical Methods, Algorithms, and Software.
8. H.-J. Bungartz, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.*
9. T.J. Barth, H. Deconinck (eds.), *High-Order Methods for Computational Physics.*
10. H.P. Langtangen, A.M. Bruaset, E. Quak (eds.), *Advances in Software Tools for Scientific Computing.*
11. B. Cockburn, G.E. Karniadakis, C.-W. Shu (eds.), *Discontinuous Galerkin Methods.* Theory, Computation and Applications.
12. U. van Rienen, *Numerical Methods in Computational Electrodynamics.* Linear Systems in Practical Applications.
13. B. Engquist, L. Johnsson, M. Hammill, F. Short (eds.), *Simulation and Visualization on the Grid.*
14. E. Dick, K. Riemslagh, J. Vierendeels (eds.), *Multigrid Methods VI.*
15. A. Frommer, T. Lippert, B. Medeke, K. Schilling (eds.), *Numerical Challenges in Lattice Quantum Chromodynamics.*
16. J. Lang, *Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems.* Theory, Algorithm, and Applications.
17. B.I. Wohlmuth, *Discretization Methods and Iterative Solvers Based on Domain Decomposition.*
18. U. van Rienen, M. Günther, D. Hecht (eds.), *Scientific Computing in Electrical Engineering.*
19. I. Babuška, P.G. Ciarlet, T. Miyoshi (eds.), *Mathematical Modeling and Numerical Simulation in Continuum Mechanics.*
20. T.J. Barth, T. Chan, R. Haimes (eds.), *Multiscale and Multiresolution Methods.* Theory and Applications.
21. M. Breuer, F. Durst, C. Zenger (eds.), *High Performance Scientific and Engineering Computing.*
22. K. Urban, *Wavelets in Numerical Simulation.* Problem Adapted Construction and Applications.

23. L.F. Pavarino, A. Toselli (eds.), *Recent Developments in Domain Decomposition Methods.*

24. T. Schlick, H.H. Gan (eds.), *Computational Methods for Macromolecules: Challenges and Applications.*

25. T.J. Barth, H. Deconinck (eds.), *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics*.

26. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations*.

27. S. Müller, *Adaptive Multiscale Schemes for Conservation Laws*.

28. C. Carstensen, S. Funken, W. Hackbusch, R.H.W. Hoppe, P. Monk (eds.), *Computational Electromagnetics*.

29. M.A. Schweitzer, *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*.

30. T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders (eds.), *Large-Scale PDE-Constrained Optimization*.

31. M. Ainsworth, P. Davies, D. Duncan, P. Martin, B. Rynne (eds.), *Topics in Computational Wave Propagation*. Direct and Inverse Problems.

32. H. Emmerich, B. Nestler, M. Schreckenberg (eds.), *Interface and Transport Dynamics.* Computational Modelling.

33. H.P. Langtangen, A. Tveito (eds.), *Advanced Topics in Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming.

34. V. John, *Large Eddy Simulation of Turbulent Incompressible Flows.* Analytical and Numerical Results for a Class of LES Models.

35. E. Bänsch (ed.), *Challenges in Scientific Computing - CISC 2002.*

36. B.N. Khoromskij, G. Wittum, *Numerical Solution of Elliptic Differential Equations by Reduction to the Interface.*

37. A. Iske, *Multiresolution Methods in Scattered Data Modelling.*

38. S.-I. Niculescu, K. Gu (eds.), *Advances in Time-Delay Systems.*

39. S. Attinger, P. Koumoutsakos (eds.), *Multiscale Modelling and Simulation.*

40. R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Wildlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering.*

41. T. Plewa, T. Linde, V.G. Weirs (eds.), *Adaptive Mesh Refinement – Theory and Applications.*

42. A. Schmidt, K.G. Siebert, *Design of Adaptive Finite Element Software.* The Finite Element Toolbox ALBERTA.

43. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations II.*

44. B. Engquist, P. Lötstedt, O. Runborg (eds.), *Multiscale Methods in Science and Engineering.*

45. P. Benner, V. Mehrmann, D.C. Sorensen (eds.), *Dimension Reduction of Large-Scale Systems.*

46. D. Kressner, *Numerical Methods for General and Structured Eigenvalue Problems.*

47. A. Boriçi, A. Frommer, B. Joó, A. Kennedy, B. Pendleton (eds.), *QCD and Numerical Analysis III*.

48. F. Graziani (ed.), *Computational Methods in Transport.*

49. B. Leimkuhler, C. Chipot, R. Elber, A. Laaksonen, A. Mark, T. Schlick, C. Schütte, R. Skeel (eds.), *New Algorithms for Macromolecular Simulation.*

50. M. Bücker, G. Corliss, P. Hovland, U. Naumann, B. Norris (eds.), *Automatic Differentiation: Applications, Theory, and Implementations.*

51. A.M. Bruaset, A. Tveito (eds.), *Numerical Solution of Partial Differential Equations on Parallel Computers.*

52. K.H. Hoffmann, A. Meyer (eds.), *Parallel Algorithms and Cluster Computing.*

53. H.-J. Bungartz, M. Schäfer (eds.), *Fluid-Structure Interaction.*

54. J. Behrens, *Adaptive Atmospheric Modeling.*

55. O. Widlund, D. Keyes (eds.), *Domain Decomposition Methods in Science and Engineering XVI.*

56. S. Kassinos, C. Langer, G. Iaccarino, P. Moin (eds.), *Complex Effects in Large Eddy Simulations.*

57. M. Griebel, M.A Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations III.*

58. A.N. Gorban, B. Kégl, D.C. Wunsch, A. Zinovyev (eds.), *Principal Manifolds for Data Visualization and Dimension Reduction.*

59. H. Ammari (ed.), *Modeling and Computations in Electromagnetics: A Volume Dedicated to Jean-Claude Nédélec.*

60. U. Langer, M. Discacciati, D. Keyes, O. Widlund, W. Zulehner (eds.), *Domain Decomposition Methods in Science and Engineering XVII.*

61. T. Mathew, *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations.*

62. F. Graziani (ed.), *Computational Methods in Transport: Verification and Validation.*

63. M. Bebendorf, *Hierarchical Matrices.* A Means to Efficiently Solve Elliptic Boundary Value Problems.

64. C.H. Bischof, H.M. Bücker, P. Hovland, U. Naumann, J. Utke (eds.), *Advances in Automatic Differentiation.*

65. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations IV*.

66. B. Engquist, P. Lötstedt, O. Runborg (eds.), *Multiscale Modeling and Simulation in Science.*

67. I.H. Tuncer, Ü. Gülcat, D.R. Emerson, K. Matsuno (eds.), *Parallel Computational Fluid Dynamics 2007.*

68. S. Yip, T. Diaz de la Rubia (eds.), *Scientific Modeling and Simulations*.

69. A. Hegarty, N. Kopteva, E. O'Riordan, M. Stynes (eds.), *BAIL* 2008 – *Boundary and Interior Layers.*

70. M. Bercovier, M.J. Gander, R. Kornhuber, O. Widlund (eds.), *Domain Decomposition Methods in Science and Engineering XVIII.*

71. B. Koren, C. Vuik (eds.), *Advanced Computational Methods in Science and Engineering.*

72. M. Peters (ed.), *Computational Fluid Dynamics for Sport Simulation.*

73. H.-J. Bungartz, M. Mehl, M. Schäfer (eds.), *Fluid Structure Interaction II - Modelling, Simulation, Optimization.*

74. D. Tromeur-Dervout, G. Brenner, D.R. Emerson, J. Erhel (eds.), *Parallel Computational Fluid Dynamics 2008.*

75. A.N. Gorban, D. Roose (eds.), *Coping with Complexity: Model Reduction and Data Analysis.*

76. J.S. Hesthaven, E.M. Rønquist (eds.), *Spectral and High Order Methods for Partial Differential Equations.*

77. M. Holtz, *Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance*.

78. Y. Huang, R. Kornhuber, O.Widlund, J. Xu (eds.), *Domain Decomposition Methods in Science and Engineering XIX*.

79. M. Griebel, M.A. Schweitzer (eds.), *Meshfree Methods for Partial Differential Equations V*.

80. P.H. Lauritzen, C. Jablonowski, M.A. Taylor, R.D. Nair (eds.), *Numerical Techniques for Global Atmospheric Models*.

81. C. Clavero, J.L. Gracia, F.J. Lisbona (eds.), *BAIL 2010 – Boundary and Interior Layers, Computational and Asymptotic Methods*.

82. B. Engquist, O. Runborg, Y.R. Tsai (eds.), *Numerical Analysis and Multiscale Computations*.

83. I.G. Graham, T.Y. Hou, O. Lakkis, R. Scheichl (eds.), *Numerical Analysis of Multiscale Problems*.

84. A. Logg, K.-A. Mardal, G. Wells (eds.), *Automated Solution of Differential Equations by the Finite Element Method*.

85. J. Blowey, M. Jensen (eds.), *Frontiers in Numerical Analysis - Durham 2010.*

86. O. Kolditz, U.-J. Gorke, H. Shao, W. Wang (eds.), *Thermo-Hydro-Mechanical-Chemical Processes in Fractured Porous Media - Benchmarks and Examples*.

87. S. Forth, P. Hovland, E. Phipps, J. Utke, A. Walther (eds.), *Recent Advances in Algorithmic Differentiation*.

88. J. Garcke, M. Griebel (eds.), *Sparse Grids and Applications*.

For further information on these books please have a look at our mathematics catalogue at the following URL: www.springer.com/series/3527

Monographs in Computational Science and Engineering

1. J. Sundnes, G.T. Lines, X. Cai, B.F. Nielsen, K.-A. Mardal, A. Tveito, *Computing the Electrical Activity in the Heart.*

For further information on this book, please have a look at our mathematics catalogue at the following URL: www.springer.com/series/7417

Texts in Computational Science and Engineering

1. H. P. Langtangen, *Computational Partial Differential Equations.* Numerical Methods and Diffpack Programming. 2nd Edition
2. A. Quarteroni, F. Saleri, P. Gervasio, *Scientific Computing with MATLAB and Octave.* 3rd Edition
3. H. P. Langtangen, *Python Scripting for Computational Science*. 3rd Edition
4. H. Gardner, G. Manduchi, *Design Patterns for e-Science*.
5. M. Griebel, S. Knapek, G. Zumbusch, *Numerical Simulation in Molecular Dynamics*.
6. H. P. Langtangen, *A Primer on Scientific Programming with Python.* 3rd Edition
7. A. Tveito, H. P. Langtangen, B. F. Nielsen, X. Cai, *Elements of Scientific Computing.*
8. B. Gustafsson, *Fundamentals of Scientific Computing.*
9. M. Bader, *Space-Filling Curves.*
10. M. Larson, F. Bengzon, *The Finite Element Method: Theory, Implementation and Applications.*

For further information on these books please have a look at our mathematics catalogue at the following URL: www.springer.com/series/5151