Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2015.w05si

Vol. **8**, No. 1, pp. 47-77
February 2015

Selected Recent Applications of Sparse Grids

Benjamin Peherstorfer[1,*], Christoph Kowitz[2], Dirk Pflüger[3] and Hans-Joachim Bungartz[1]

[1] *Scientific Computing Group, Dept. of Computer Science, Boltzmannstr. 3, 85748 Garching, Germany.*
[2] *Institute for Advanced Study, Technische Universität München, Lichtenbergstr. 2a, 85748 Garching, Germany.*
[3] *Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany.*

Received 2 November 2013; Accepted 23 July 2014

Abstract. Sparse grids have become a versatile tool for a vast range of applications reaching from interpolation and numerical quadrature to data-driven problems and uncertainty quantification. We review four selected real-world applications of sparse grids: financial product pricing with the Black-Scholes model, interactive exploration of simulation data with sparse-grid-based surrogate models, analysis of simulation data through sparse grid data mining methods, and stability investigations of plasma turbulence simulations.

AMS subject classifications: 65M60, 65M22, 62P05, 65D05, 62H30, 65F15

Key words: Sparse grids, hierarchical decomposition, adaptivity, combination technique.

1. Introduction

The underlying principle of sparse grids is a hierarchical basis that leads to a hierarchical decomposition of function spaces into hierarchical increments. These hierarchical increments are then the starting point for optimization problems with which one constructs approximation spaces for function spaces by selecting only those increments which have a sufficiently good cost-benefit ratio; the costs equal the dimension of the approximation space, and the benefit is related to the interpolation error in a given norm. Sparse grid spaces are optimal approximation spaces with respect to these criteria for the space H^2_{mix}, which contains functions with bounded, mixed derivatives up to order two. The corresponding theory is presented in the survey article [9].

Sparse grids have been applied to a variety of computational tasks. The purpose of this article is to highlight four selected and recently presented real-world applications.

[*]Corresponding author. *Email address:* `peehersto@in.tum.de` (B. Peherstorfer)

We summarize a few key facts on sparse grids in Section 2 and clarify the notation. Our presentation particularly emphasizes that sparse grids build on hierarchical and multi-level principles. In Section 3, we consider financial product pricing, where the multi-dimensional Black-Scholes equation is solved. In Section 4, a sparse-grid-based surrogate modeling approach for interactive visual exploration of parametrized simulation data is discussed. We construct a surrogate model for a building information model (BIM) that simulates a flow through a building. We continue in Section 5 with a data-driven problem where we analyze simulation data with sparse grid data mining methods. Finally, multi-dimensional eigenvalue problems for plasma turbulence simulations are solved on sparse grids in Section 6. The eigenvalues and eigenvectors give information about whether the plasma is stable or not.

2. Sparse grid spaces

We give a brief overview of sparse grids and particularly emphasize the strong relationship to hierarchical and multi-level computational methods. We also discuss the combination technique, spatial adaptivity, and list a few software libraries that implement common sparse grid routines. We do not go into the details of sparse grid theory but only present the preliminaries for the following applications. We refer to the survey article [9] for details.

2.1. Full grid spaces and their hierarchical decomposition

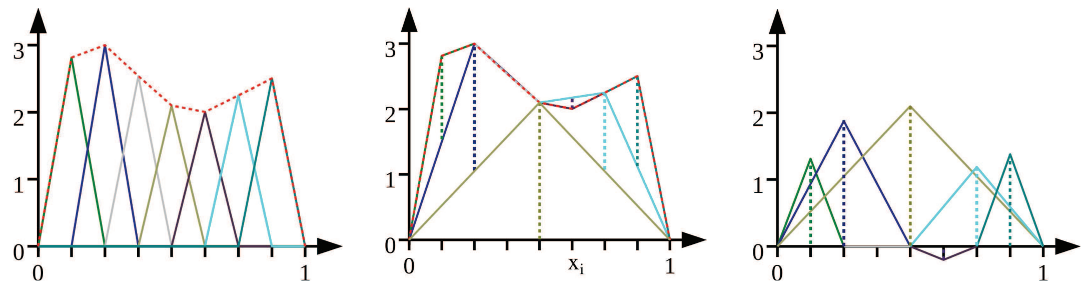
Let $\mathcal{V}$ be a function space with domain $\Omega = [0,1]$ and homogeneous boundaries, e.g., $\mathcal{V} = H_0^2(\Omega)$. We discretize a function $f \in \mathcal{V}$ by constructing its interpolant in the finite-dimensional space $\mathcal{V}_\ell^{(\infty)} \subset \mathcal{V}$ of piecewise linear functions with mesh width $2^{-\ell}$. The accuracy of the interpolant is controlled by the level ℓ of the space. The space $\mathcal{V}_\ell^{(\infty)}$ is spanned by the basis functions

$$\varphi_i(x) := \phi(2^\ell x - i)\,, \quad 1 \le i < 2^\ell\,, \tag{2.1}$$

where $\phi : [-1,1] \to \mathbb{R}$ with $\phi(x) = \max\{1-|x|, 0\}$. The interpolant $\hat{f} \in \mathcal{V}_\ell^{(\infty)}$ of $f \in \mathcal{V}$ can be represented as a linear combination

$$\hat{f} = \sum_{i=1}^{N} a_i \varphi_i$$

of the basis functions (2.1) and coefficients $\boldsymbol{a} = [a_1, \cdots, a_N]^T$ where $N = 2^\ell - 1$. It follows that $a_i = f(i \cdot 2^{-\ell})$ for $i = 1, \cdots, N$. On the one hand this means that $\hat{f}$ is easy (w.r.t. the implementation effort) and cheap (w.r.t. the computational costs) to compute. On the other hand, the coefficients $\boldsymbol{a}$ do not lead to an ordering of the basis functions with respect to the benefit of including a basis function into the linear



(a) nodal basis (b) hierarchical basis (c) hierarchical basis with coefficients

Figure 1: This plot shows a function represented in the nodal point basis (a) and the hierarchical basis (b). The hierarchical basis functions multiplied with their coefficients are shown in (c).

combination (2.1). Thus, the basis (2.1) of $\mathcal{V}_\ell^{(\infty)}$ leads to a representation of $\hat{f}$ that gives all basis functions equal importance, cf. Fig. 1.

Let us now introduce a basis for $\mathcal{V}_\ell^{(\infty)}$ that follows a hierarchical or multi-level approach [9,21,69]. We define the one-dimensional hierarchical basis function of level $l \in \mathbb{N}$ and index $i \in \mathbb{N}$ as

$$\phi_{l,i}(x) := \phi(2^l x - i) \tag{2.2}$$

and define the hierarchical increments

$$\mathcal{W}_l = \operatorname{span}\left\{\phi_{l,i} \,|\, 1 \leq i < 2^l, i \text{ odd}\right\}.$$

The space $\mathcal{V}_\ell^{(\infty)}$ can be represented as a direct sum of hierarchical increments,

$$\mathcal{V}_\ell^{(\infty)} = \bigoplus_{l=1}^{\ell} \mathcal{W}_l\,, \tag{2.3}$$

which lets us represent the interpolant $\hat{f} \in \mathcal{V}_\ell^{(\infty)}$ as

$$\hat{f} = \sum_{l=1}^{\ell} \hat{f}_l = \sum_{l=1}^{\ell} \sum_i \alpha_{l,i} \phi_{l,i} \tag{2.4}$$

with $\hat{f}_l \in \mathcal{W}_l$ and the hierarchical coefficients $\alpha_{l,i}$, see [9] for proofs. The procedure for obtaining the hierarchical coefficients is called hierarchisation; its inverse operation is dehierarchisation. It can be shown [9] that for functions that satisfy certain smoothness assumptions, the coefficients $\alpha_{l,i}$ corresponding to the hierarchical increment $\mathcal{W}_l$ of the interpolant $\hat{f}$ are of order 2^{-2l}, i.e., $|\alpha_{l,i}| \in \mathcal{O}(2^{-2l})$. Thus, the impact of $\hat{f}_l$ onto the overall approximation accuracy of $\hat{f}$ decreases with the level l of the hierarchical increment $\mathcal{W}_l$. Thus, we know that the hierarchical increments and its basis functions with a hierarchically higher level l (i.e., l small) have a higher impact on the result than the basis functions of hierarchically lower levels (i.e., l large). Hence, we obtained an ordering of the basis function with respect to their importance on the interpolation accuracy, see Fig. 1.

2.2. Multi-dimensional hierarchical basis and sparse grid spaces

Consider now a domain $\Omega = [0,1]^d$ with $d \in \mathbb{N}$. We extend (2.2) to the d-dimensional function $\phi_{\boldsymbol{l},\boldsymbol{i}}$ as

$$\phi_{\boldsymbol{l},\boldsymbol{i}}(\boldsymbol{x}) = \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j)\,, \tag{2.5}$$

where $\boldsymbol{l} = (l_1, \cdots, l_d) \in \mathbb{N}^d$ and $\boldsymbol{i} = (i_1, \cdots, i_d) \in \mathbb{N}^d$ are the level and index multi-indices, respectively. Furthermore, we introduce the d-dimensional hierarchical increment $\mathcal{W}_{\boldsymbol{l}}$ of level $\boldsymbol{l}$ which is spanned by the basis functions in

$$\left\{ \phi_{\boldsymbol{l},\boldsymbol{i}} \,|\, \boldsymbol{i} \in \mathbb{N}^d, 1 \leq i_j < 2^{l_j}, i_j \notin 2\mathbb{N}, 1 \leq j \leq d \right\}. \tag{2.6}$$

We define a function space $\mathcal{V}_{\mathcal{L}}$ as direct sum of hierarchical increments

$$\mathcal{V}_{\mathcal{L}} = \bigoplus_{\boldsymbol{l} \in \mathcal{L}} \mathcal{W}_{\boldsymbol{l}}\,, \tag{2.7}$$

where the set $\mathcal{L} \subset \mathbb{N}^d$ is a selection of levels. We then define $\mathcal{I}$ as the set that contains the level-index pairs $(\boldsymbol{l}, \boldsymbol{i})$ of the basis functions and thus can represent a function $\hat{f} \in \mathcal{V}_{\mathcal{L}}$ as linear combination

$$\hat{f} = \sum_{\boldsymbol{l} \in \mathcal{L}} \hat{f}_{\boldsymbol{l}} = \sum_{\boldsymbol{l} \in \mathcal{L}} \sum_{\boldsymbol{i}} \alpha_{\boldsymbol{l},\boldsymbol{i}} \phi_{\boldsymbol{l},\boldsymbol{i}} = \sum_{(\boldsymbol{l},\boldsymbol{i}) \in \mathcal{I}} \alpha_{\boldsymbol{l},\boldsymbol{i}} \phi_{\boldsymbol{l},\boldsymbol{i}}\,. \tag{2.8}$$

We might also write (2.8) as $\hat{f} = \sum_i \alpha_i \phi_i$ if we do not emphasize the hierarchical decomposition. Then, the full grid space $\mathcal{V}_\ell^{(\infty)}$ of piecewise d-linear functions of level ℓ can be represented as

$$\mathcal{V}_\ell^{(\infty)} = \bigoplus_{|\boldsymbol{l}|_\infty \leq \ell} \mathcal{W}_{\boldsymbol{l}}\,, \tag{2.9}$$

where $|\boldsymbol{l}|_\infty = \max_j l_j$ and thus $\mathcal{L} = \{\boldsymbol{l} \in \mathbb{N}^d \,|\, \max_j l_j \leq \ell, 1 \leq j \leq d\}$. The sparse grid space of level ℓ and dimension d with respect to the L^2 norm is

$$\mathcal{V}_\ell^{(1)} = \bigoplus_{|\boldsymbol{l}|_1 \leq \ell + d - 1} \mathcal{W}_{\boldsymbol{l}}\,, \tag{2.10}$$

where $|\boldsymbol{l}|_1 = \sum_j l_j$, see [9, 71].

Let us summarize a few properties of the sparse grid space $\mathcal{V}_\ell^{(1)}$ as defined in (2.10) and the full grid space $\mathcal{V}_\ell^{(\infty)}$ as defined in (2.9) for functions $f \in H^2_{\text{mix}}$. The proofs can be found in [9]. The full grid space $\mathcal{V}_\ell^{(\infty)}$ of level ℓ and mesh width $h = 2^{-\ell}$ is spanned by $\mathcal{O}(2^{\ell d})$ basis functions, whereas the sparse grid space requires only $\mathcal{O}(2^\ell \ell^{d-1})$. The interpolation error of $\hat{f}^{(\infty)} \in \mathcal{V}_\ell^{(\infty)}$ is in $\mathcal{O}(2^{-2\ell})$, and the error of $\hat{f}^{(1)} \in \mathcal{V}_\ell^{(1)}$ is in $\mathcal{O}(2^{-2\ell}\ell^{d-1})$. Thus, whereas we save many basis functions (and thus grid points), the interpolation error of the sparse grid interpolant deteriorates only slightly compared to the full grid interpolant. In particular, the number of basis functions of the sparse grid spaces does not grow so dramatically with the dimension d as for full grid spaces. Hence, sparse grid spaces are well-suited for moderately high-dimensional problems.

Note that the basis functions in (2.6) evaluate to zero at the boundary $\partial\Omega$ and thus we have restricted this introduction to homogeneous boundary conditions. However, the applications of the following sections will require inhomogeneous boundary conditions. We do not discuss the corresponding theory here but refer to, e.g., [24]. Note further that the sparse grid spaces can also be constructed from higher order polynomial basis functions, wavelets, or B-splines [8, 19, 24, 58, 61].

2.3. Combination technique

A different approach to sparse grid approximations is the so-called combination technique [35]. It represents a sparse grid as a superposition of much coarser full grids. Then, instead of solving the approximation problem (e.g., interpolation, solution of PDEs) directly in the sparse grid space, the problem is solved on each of these full grids independently. For each grid, a partial solution is obtained. These are combined according to the combination technique scheme to eventually obtain the final solution. The so obtained final solution has similar properties as the sparse grid solution. Under certain conditions, it can be shown that the combination technique solution is equal to the sparse grid solution [10–12, 38, 60].

Let us consider the combination technique for the interpolation problem. Let $f \in \mathcal{V} = H^2_{\text{mix}}(\Omega)$ and let $\hat{f} \in \mathcal{V}_\ell^{(1)}$ be the sparse grid interpolant of f. We define the full grid spaces $\mathcal{V}_{\boldsymbol{l}}^{(\infty)} = \bigoplus_{\boldsymbol{k} \leq \boldsymbol{l}} W_{\boldsymbol{k}}$ of level $\boldsymbol{l} \in \mathbb{N}^d$ and the corresponding interpolants $\hat{f}_{\boldsymbol{l}} \in \mathcal{V}_{\boldsymbol{l}}^{(\infty)}$ of f. To obtain the combination technique interpolant $\hat{f}^C$ corresponding to level ℓ we combine

$$\hat{f}^C = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{\sum_i l_i = \ell + (d-1) - q} \hat{f}_{\boldsymbol{l}}\,. \tag{2.11}$$

For the interpolation task, $\hat{f}^C$ equals the sparse grid interpolant $\hat{f}^{(1)}$. This means, we combine many coarse approximations and obtain an accurate approximation in the end (extrapolation) [35]. The same principle is possible for PDEs: We solve a PDE with the finite element method and the finite-dimensional spaces $\mathcal{V}_\ell^{(\infty)}$ and combine them according to (2.11) and obtain a solution $\hat{f}^C$ that is more accurate than the partial solutions in the coarse full grid spaces [12]. We discuss eigenvalue problems and the combination technique in detail in Section 6.

The advantages of the combination technique are that we obtain several independent problems that can be solved in parallel, and that these problems are discretized on ordinary full grid spaces which allow us to reuse available codes. The disadvantage of the combination technique is that spatial adaptivity is very limited [59]. Furthermore, for more general problems no error bounds are known for the combination technique solution [38]. However, there are improved combination technique schemes that allow to adaptively select the partial solutions [28,37] and there are also optimized schemes where the coefficients in the combination scheme are optimized with respect to the current problem at hand, see Section 6.

2.4. Adaptively refined sparse grid spaces

We have seen in Section 2.2 that the sparse grid space $\mathcal{V}_\ell^{(1)}$ is optimal with respect to the interpolation error for the approximation of functions in H^2_{mix}. However, since this has to hold for all functions in H^2_{mix}, the space $\mathcal{V}_\ell^{(1)}$ cannot take peculiarities of specific functions of H^2_{mix} into account. Therefore, we speak of the regular sparse grid space $\mathcal{V}_\ell^{(1)}$ of level ℓ. However, in many situations, we want to incorporate the properties of the current function $f \in H^2_{\text{mix}}$ at hand into the approximation approach. This is achieved with adaptive refinement, and we obtain an adaptive sparse grid space $\mathcal{V}^{(1)}$ that is adapted to the function f [9]. In many of the following applications, adaptive refinement plays a crucial role because it allows to further reduce the number of basis functions (grid points) required to achieve a given approximation accuracy. Furthermore, experience shows that even non-smooth problems can be dealt with if adaptivity is employed [30,62,63].

An adaptivity criterion is required to decide in which regions of the domain the sparse grid should be refined. The hierarchical decomposition (2.8) is again advantageous here because the hierarchical coefficients $\alpha_{\boldsymbol{l},\boldsymbol{i}}$ for $(\boldsymbol{l},\boldsymbol{i}) \in \mathcal{I}$ can be used as adaptivity criterion. The most widely-used criterion of this kind is to refine those grid points which correspond to the hierarchical coefficients with the largest absolute values. Even though this simple criterion works well in many situations—see the following applications—there are more sophisticated refinement strategies, see [61–63] for detailed studies. We show an example of a refined sparse grid in Fig. 2 which also demonstrates that many sparse grid algorithms require creating all hierarchical ancestors (gray points), see, e.g., [53,61].

2.5. The SG^{++} sparse grid software library

All of the following applications where computed with the SG^{++} sparse grid library developed primarily by Dirk Pflüger [61]. It is written in C++ but provides interfaces to Python, Java, and MATLAB. Over the years, it has grown to a vast library that contains procedures and routines for basic as well as advanced sparse grid tasks. Besides the standard operations such as creating sparse grids and (spatially)

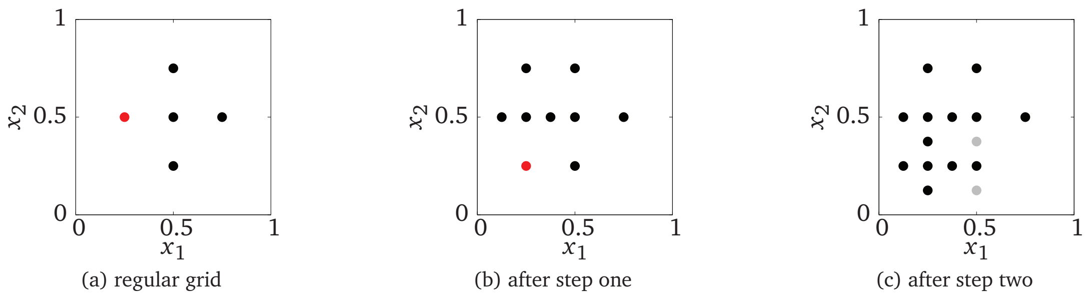


Figure 2: The regular grid of level two in (a) is refined to obtain the sparse grid in (b). After another refinement step, the hierarchical ancestors (gray points) are created in (c) because most sparse grid algorithms require that the hierarchical ancestors of all grid points exist.

adaptively refining them, it also contains the basic algorithms for interpolation (hierarchisation/dehierarchisation). A strong focus is on data-driven sparse grid routines that include regression, classification, and density estimation. It also contains methods to discretize and solve (UpDown algorithm [1,3,8]) second-order parabolic PDEs on spatially adaptive sparse grids. This also includes an environment to solve option pricing problems, see Section 3. Additionally, several routines for numerical quadrature are available.

Another widely-used library is the Sparse Grid Interpolation Toolbox for MATLAB [45,46]. It contains routines for interpolation, numerical quadrature, optimization, and others.

3. PDEs: Pricing of financial products

The volume and variety of financial derivative products has grown significantly over the last years. One computationally challenging task in that context is pricing basket options which often requires solving the multi-dimensional Black-Scholes equation. To cope with the curse of dimensionality, we discretize the Black-Scholes equation with the finite element method on adaptive sparse grids. We then discuss a Krylov subspace solver for the corresponding system of linear equations. Finally, we present results for options with up to six underlying assets.

3.1. Option pricing with the Black-Scholes model

It is common to solve multi-dimensional option pricing problems, e.g., the Black-Scholes equation of the Black-Scholes model, with Monte Carlo (MC) methods because they are easy to implement and can cope with high-dimensional problems; however, they exhibit a poor convergence rate. Even though several techniques (e.g., quasi MC, adaptive MC, multi-level MC) exist that aim at improving the convergence rate [31–33], PDE methods have been examined recently because they often exhibit a faster convergence rate. Furthermore, with PDE methods, one does not only obtain the price of the option but also the (so-called) Greeks (derivatives) which are needed for risk assessment.

Usually, the sparse grid combination technique is employed which allows to reuse available solvers but which cannot be used with adaptively refined grids, cf. Section 2 and see, e.g., [4,65]. In contrast, we present here a fully adaptive approach that directly discretizes the Black-Scholes equation on a sparse grid and performs the Galerkin projection onto a sparse grid space. This is particularly advantageous here, as the payoff function (terminal condition) is only locally non-smooth and so additional grid points are only required locally.

The Black-Scholes model assumes that the stock value S follows a geometric Brownian motion defined by the stochastic differential equation

$$\mathrm{d}S(t) = \mu_{\text{GMB}} S(t)\mathrm{d}t + \sigma_{\text{GMB}} S(t)\mathrm{d}W(t) \tag{3.1}$$

with drift μ_{GMB}, standard deviation σ_{GMB}, and stochastic Wiener process $W(t)$. The Black-Scholes equation is then derived from (3.1). Here, we consider the log-transformed Black-Scholes equation for a European basket option with d assets and expiration time T given by

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{d} \sigma_i\sigma_j\rho_{ij}\frac{\partial^2 u}{\partial S_i \partial S_j} + \sum_{i=1}^{d}\left(\mu_i - \frac{1}{2}\sigma_i^2\right)\frac{\partial u}{\partial S_i} - ru = 0\,, \tag{3.2}$$

where $u(\boldsymbol{S},t)$ denotes the value of the option at stock price $\boldsymbol{S} = (S_1,\cdots,S_d) \in \mathbb{R}^d$ and forward time $t \in [0,T]$. The parameters σ_i, ρ_{ij}, and μ_i are the volatilities, asset correlations, and drifts, respectively. The risk-free interest rate is denoted by r. The terminal condition is the standard payoff function

$$V(\boldsymbol{S},T) = \max\left\{K - \frac{1}{d}\sum_{i=1}^{d} S_i, 0\right\} \tag{3.3}$$

that corresponds to a put option and depends on the price (strike) K. We emphasize that we define the terminal condition rather than the initial condition and so solve (3.2) backward in time. Note that the payoff function has a sharp bend near which adaptivity will be beneficial.

3.2. Discretization of the Black-Scholes equation on sparse grids

We solve the multi-dimensional, log-transformed Black-Scholes equation (3.2) with the finite element method and sparse grids. The weak form of (3.2) is given as

$$\frac{\partial}{\partial t}\langle u,\psi\rangle_{L^2} + \frac{1}{2}\sum_{i,j=1}^{d}\sigma_i\sigma_j\rho_{ij}\left\langle \frac{\partial u}{\partial S_i}, \frac{\partial \psi}{\partial S_j}\right\rangle_{L^2} - \sum_{i=1}^{d}\left(\mu_i - \frac{1}{2}\sigma_i^2\right)\left\langle \frac{\partial u}{\partial S_i}, \psi\right\rangle_{L^2} + r\langle u,\psi\rangle_{L^2} = 0$$

where ψ is a test function of a suitable test space and $\langle\cdot,\cdot\rangle_{L^2}$ the L^2 dot product. We choose the domain $\Omega=[0,S_1^{\max}]\times\cdots\times[0,S_d^{\max}]$ so large that we can safely impose homogeneous Dirichlet boundary conditions. We refer to [13] for an estimation of the error induced by this truncation of the domain. With Ritz-Galerkin projection onto a (possibly adaptively refined) sparse grid space $\mathcal{V}^{(1)}$, which is spanned by the basis functions in $\{\phi_1,\cdots,\phi_N\}$, we obtain a solution $\hat{u}=\sum_{i=1}^N\alpha_i\phi_i\in\mathcal{V}^{(1)}$ by solving the system of ODEs

$$\boldsymbol{B}\frac{\partial}{\partial t}\boldsymbol{\alpha}(t)=-\frac{1}{2}\sum_{i,j=1}^{d}\sigma_i\sigma_j\rho_{ij}\boldsymbol{C}^{(ij)}\boldsymbol{\alpha}+\sum_{i=1}^{d}\left(\mu_i-\frac{1}{2}\sigma_i^2\right)\boldsymbol{D}^{(i)}\boldsymbol{\alpha}-r\boldsymbol{B}\boldsymbol{\alpha} \tag{3.4}$$

with the coefficient vector $\boldsymbol{\alpha}=[\alpha_1,\cdots,\alpha_N]^T$ and the matrices $\boldsymbol{B},\boldsymbol{C}^{(ij)},\boldsymbol{D}^{(i)}\in\mathbb{R}^{N\times N}$ with entries $B_{pq}=\langle\phi_p,\phi_q\rangle_{L^2}, C_{pq}^{(ij)}=\langle\frac{\partial\phi_p}{\partial S_j},\frac{\partial\phi_q}{\partial S_i}\rangle_{L^2}$, and $D_{pq}^{(i)}=\langle\frac{\partial\phi_p}{\partial S_i},\phi_q\rangle_{L^2}$. The quadratic Crank-Nicolson scheme is used for the time discretization where a Rannacher smoothing [64] with backward Euler is employed during the first few time steps due to the non-smooth payoff function. The result is a non-symmetric system of linear equations.

In the context of sparse grids, the challenge is now to solve the system (3.4). Due to the structure of sparse grids and the hierarchical basis functions, the system matrices in (3.4) are not sparse anymore. Thus, assembling them is too costly.

3.3. Sparse grid Krylov subspace solver

Krylov subspace methods to solve systems of linear equations $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$ only require the matrix-vector product with the $N\times N$ system matrix $\boldsymbol{A}$. Because the matrices of the system (3.4) in the hierarchical basis are dense, a straightforward implementation would lead to quadratic costs in the number of sparse grid points. However, due to the tensor product structure of the basis functions (2.5) and the matrices in (3.4), algorithms can be constructed that provide the matrix-vector product in $\mathcal{O}(N)$ rather than in $\mathcal{O}(N^2)$.

These algorithms are summarized under the name UpDown scheme [1, 3, 8, 24, 61, 70], which is based on the uni-directional principle. In the context of sparse grids, d-dimensional algorithms following the uni-directional principle consist of d one-dimensional algorithms which operate in each dimension. For example, let us consider the matrix-vector product $\boldsymbol{B}\boldsymbol{\alpha}$ with the matrix $\boldsymbol{B}$ stemming from the bilinear form $\langle\cdot,\cdot\rangle_{L^2}$, see (3.4). Due to (2.5), we obtain

$$\langle\phi_{\boldsymbol{l},\boldsymbol{i}},\phi_{\boldsymbol{l}',\boldsymbol{i}'}\rangle_{L^2}=\left\langle\prod_{j=1}^{d}\phi_{l_j,i_j},\phi_{l'_j,i'_j}\right\rangle_{L^2}=\prod_{j=1}^{d}\langle\phi_{l_j,i_j},\phi_{l'_j,i'_j}\rangle_{L^2},$$

where we can rewrite the d-dimensional bilinear form as product of one-dimensional ones. A similar decomposition can be derived for the bilinear forms corresponding to the matrices $\boldsymbol{C}^{(ij)}$ and $\boldsymbol{D}^{(i)}$. The one-dimensional subgrids of a multi-dimensional

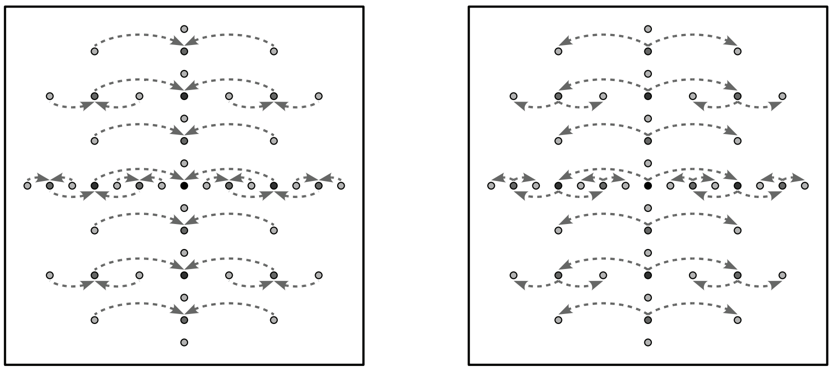


Figure 3: In the figure on the left, the Up traversal of a two-dimensional tree in the first direction is shown, and the corresponding Down traversal on the right. With these tree traversal it is possible to provide the matrix-vector product with the system matrices in linear complexity with respect to the number of sparse grid points [13].

sparse grid are traversed according to a nested and recursive scheme so that intermediate values can be stored without requiring additional storage. This is achieved by splitting the results at each grid point into the contribution from the hierarchically higher and lower basis functions. This also explains the name UpDown which refers to the Up and Down traversal of the hierarchical tree representing the basis functions, see Fig. 3. We refer to [24, 61] for details.

This decomposition leads to efficient algorithms that are applicable in case of non-adaptive sparse grids as well as in the case of adaptively refined sparse grids. Even though these algorithms are recursive, it is possible to efficiently parallelize them [14, 41]. The matrix-vector procedures can then be used within BiCGStab or other Krylov subspace methods. The drawback of this approach is that preconditioners are still required to obtain a fast solver, which is a topic of current research in the context of sparse grids [23, 34, 34, 70].

3.4. Results for European basket options

We present accuracy results of the presented sparse grid approach for option pricing. The results reported here are presented in detail in [41]. We also refer to [13, 14, 41] for a detailed study on adaptivity criteria, implementation details, and runtime measurements on multi- and many-core systems.

Here, we only consider European basket put options with five and six assets, i.e., five- and six-dimensional problems. We use the payoff function (3.3) with strike $K = 1$ and a time to maturity of $T = 1$. The drifts μ_i are in the range $[0.05, 0.1]$, the standard deviations σ_i in $[0.2, 0.4]$, and non-zero correlations ρ_{ij} in $[-0.7, 0.7]$ to obtain complex and computationally expensive scenarios. We compare with a Monte-Carlo simulation with 10^{10} paths. Our adaptive sparse grid approach applies principal axis

Table 1: Accuracy results of sparse grid approach for pricing basket options with the Black-Scholes model [41]. Due to the spatially adaptive approach, a small number of grid points (DoFs) leads already to sufficient accuracy results.

5 asset basket option			6 asset basket option		
#DoFs	abs. err.	rel. err.	#DoFs	abs. err.	rel. err.
1,754	$1.2 \cdot 10^{-3}$	$3.5 \cdot 10^{-2}$	559	$1.1 \cdot 10^{-2}$	$2.8 \cdot 10^{-1}$
6,157	$3.0 \cdot 10^{-4}$	$8.5 \cdot 10^{-3}$	3,588	$7.2 \cdot 10^{-3}$	$1.9 \cdot 10^{-1}$
24,566	$3.2 \cdot 10^{-4}$	$9.5 \cdot 10^{-3}$	48,969	$1.1 \cdot 10^{-2}$	$2.9 \cdot 10^{-1}$
98,039	$2.2 \cdot 10^{-4}$	$6.2 \cdot 10^{-3}$	282,257	$3.8 \cdot 10^{-3}$	$9.9 \cdot 10^{-2}$

transformation to the Black-Scholes equation and uses a refinement criterion based on the hierarchical coefficients, see Section 2. The accuracy results in Table 1 demonstrate that we achieve an accuracy of about 10^{-4} with a low number of sparse grid points.

4. Surrogate models: Interactive exploration of building information models

In this section, we consider low-cost surrogate models of parametrized simulations for interactive visual exploration in cave automatic virtual environments (CAVEs). We first discuss that interactive response times to parameter changes (below 0.2 seconds) usually cannot be achieved for large-scale simulations without surrogate models. We then present sparse-grid-based surrogate models for visual exploration and discuss their main advantage, namely, that they are non-intrusive, i.e., we can treat the large-scale simulation solver as a black box. We continue with the properties and characteristics of our surrogate modeling approach and an application where buildings are explored. We refer to, e.g., [15–18, 57] for details and more applications.

4.1. Interactive visual exploration of parametrized simulations

Numerical simulations depend on many parameters that define, e.g., material properties, geometry configurations, or initial conditions. A common task during design, construction, manufacturing, and production phases is to study and examine the dependencies between the parameters and the simulation result. For example, one could be interested in how the temperature field in a room changes when a window is tilted instead of closed. A straightforward and effective approach is the visual exploration of the simulation results for different parameter configurations. Here, the goal is an interactive exploration where the engineer (user) can change the parameters (e.g., the angle of the window) on a control panel and the result (e.g., the temperature field in the room) is immediately visualized. Interactive means that the system has to respond to parameter changes within 0.2 seconds [15]. However, most of today's simulations require vast computing resources to deliver the result corresponding to only a single parameter configuration, and still require several minutes or even hours to compute.

Therefore, it is not sufficient to simply attach such simulations to visualization environments.

With low-fidelity surrogate models we approximate the large-scale, high-fidelity simulations so that we do not have to perform the expensive simulations for each parameter change. The computational procedure is divided into an expensive offline (pre-processing) and a rapid online phase. In the offline phase, the surrogate model is constructed. Usually, this requires to solve the original, high-fidelity simulation at several parameter configurations from which then the low-fidelity model is derived. In the online phase, the actual exploration takes place, i.e., the surrogate model is evaluated many times. Thus, the offline phase is performed only once, but the online phase, i.e., the evaluation of the surrogate model, is repeated many times. This means, the expensive offline phase is compensated by many, rapid evaluations of the surrogate model in the online phase. Still, when we consider the time to solution, the costs of the offline phase have to be taken into account. In particular, we have to keep the number of large-scale simulations, which are required to build the surrogate model, low.

4.2. Sparse-grid-based surrogate models

In the following, we denote parametrized, discretized simulations with $z : \mathcal{D} \rightarrow \mathbb{R}^{\mathcal{N}}$ where $\mathcal{D} \subset \mathbb{R}^d$ is the parameter domain. The simulation z evaluated at a parameter $\boldsymbol{\mu} \in \mathcal{D}$ is an $\mathcal{N}$-dimensional vector $\boldsymbol{z}(\boldsymbol{\mu}) \in \mathbb{R}^{\mathcal{N}}$. For example, to stay within the thermal problem introduced above, the components of $\boldsymbol{z}(\boldsymbol{\mu}) = [z_1(\boldsymbol{\mu}), \cdots, z_{\mathcal{N}}(\boldsymbol{\mu})]^T$ could be the temperature at $\mathcal{N}$ grid points with which the domain representing the room is discretized. In most cases, there is a system of partial differential equations underlying the simulation z, but we emphasize that this is not important for us because we treat the simulation z as a black box.

Let us now consider our non-intrusive sparse-grid-based surrogate modeling approach. In the offline phase, we choose a (possibly adaptively refined) sparse grid space $\mathcal{V}^{(1)}$ that is spanned by the hierarchical basis functions in $\{\phi_1, \cdots, \phi_N\}$, evaluate the simulation z at parameters $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_N \in \mathbb{R}^d$ that correspond to the N sparse grid points of $\mathcal{V}_\ell^{(1)}$, and then construct a sparse grid interpolant $\hat{z}_i \in \mathcal{V}^{(1)}$ from the data

$$\mathcal{S}_i = \{(\boldsymbol{\mu}_1, z_i(\boldsymbol{\mu}_1)), \cdots, (\boldsymbol{\mu}_N, z_i(\boldsymbol{\mu}_N))\} \subset \mathbb{R}^d \times \mathbb{R}^{\mathcal{N}} \tag{4.1}$$

for each component $z_i, i = 1, \cdots, \mathcal{N}$ of z. Since we interpolate, we know that $\hat{z}_i(\boldsymbol{\mu}_j) = z_i(\boldsymbol{\mu}_j)$ holds for all $i = 1, \cdots, \mathcal{N}$ and $j = 1, \cdots, N$. An interpolant $\hat{z}_i$ is a linear combination of the basis functions $\phi_1, \cdots, \phi_N$ with the coefficients $\boldsymbol{\alpha}^i = [\alpha_1^i, \cdots, \alpha_N^i]^T \in \mathbb{R}^N$. We have $\mathcal{N}$ interpolants that are combined into the surrogate model $\hat{z} : \mathcal{D} \rightarrow \mathbb{R}^{\mathcal{N}}$ with $\hat{\boldsymbol{z}}(\boldsymbol{\mu}) = [\hat{z}_1(\boldsymbol{\mu}), \cdots, \hat{z}_{\mathcal{N}}(\boldsymbol{\mu})]^T$. We employ the same sparse grid space for each component and thus we can write

$$\hat{z}(\boldsymbol{\mu}) = \begin{bmatrix} \hat{z}_1(\boldsymbol{\mu}) \\ \vdots \\ \hat{z}_{\mathcal{N}}(\boldsymbol{\mu}) \end{bmatrix} = \sum_{i=1}^{N} \begin{bmatrix} \alpha_i^1 \\ \vdots \\ \alpha_i^{\mathcal{N}} \end{bmatrix} \phi_i(\boldsymbol{\mu}) \tag{4.2}$$

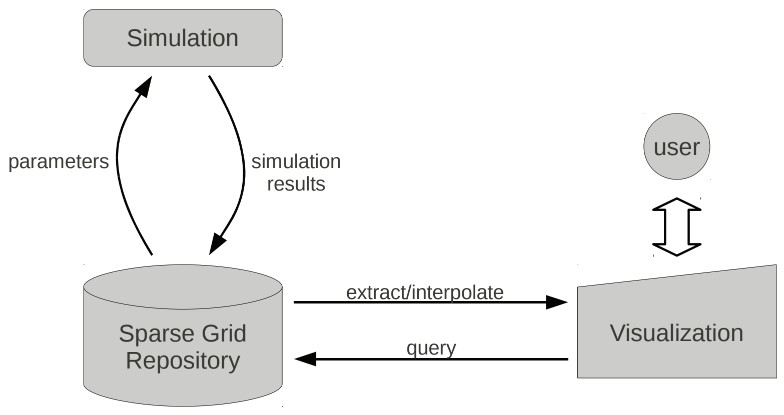


Figure 4: For the interactive visual exploration of sparse-grid-based surrogate models, we first build a sparse grid repository of the original simulation from which the surrogate model is constructed. Then, the visualization environment does not query the large-scale simulation but the surrogate model.

and so have to evaluate each basis function only once. We emphasize that the linear combination (4.2) has coefficient vectors with $\mathcal{N}$ components, i.e., as many components as the large-scale simulation $\boldsymbol{z}(\boldsymbol{\mu})$. The result of the offline phase is the sparse grid repository as shown in the workflow in Fig. 4.

Sparse grid interpolation is accomplished with the hierarchisation method which computes the hierarchical coefficients for a given sparse grid space and set of data points [9]. The hierarchisation is a multi-dimensional algorithm that relies on the unidirectional principle, i.e., it consists of one-dimensional algorithms that act in each dimension separately. The runtime is linear in the number N of sparse grid points. Furthermore, the coefficients are computed in-place, i.e., no additional storage is required. The runtime and storage complexity of the hierarchisation algorithm are crucial because we call the hierarchisation procedure for each component of $\boldsymbol{z} = [z_1, \cdots, z_{\mathcal{N}}]^T$. Thus, if we have N sparse grid points and $\mathcal{N}$ components in z, the overall runtime of the offline phase is in $\mathcal{O}(N\mathcal{N})$ and the storage in $\mathcal{O}(N\mathcal{N})$. Note that this does not include the runtime needed to obtain N large-scale simulations. For details of the hierarchisation algorithm we refer to [9].

After we have obtained the surrogate model $\hat{z}$ in the offline phase, we have to evaluate it in the online phase, see Fig. 4. Evaluating $\hat{z}$ means to compute the linear combination (4.2). Because the hierarchical basis functions have non-overlapping support in each hierarchical increment, only $\mathcal{O}(\ell^d)$ basis functions have to be evaluated from the linear combination (4.2), cf. Section 2. This means, the procedure to evaluate $\hat{z}$ at a point $\boldsymbol{\mu} \in \mathcal{D}$ is split into two parts. First, the so-called affected basis functions are determined. These are the basis functions which evaluate to non-zero values at the point $\boldsymbol{\mu}$. Sophisticated algorithms (and implementations) exist to determine the affected basis functions in $\mathcal{O}(\ell^d)$ rather than in $\mathcal{O}(N) = \mathcal{O}(2^\ell \ell^{d-1})$ as is required by a straightforward algorithm. After it is know which basis functions have to be evaluated, the corresponding $\mathcal{N}$-dimensional coefficients vectors are loaded and combined with SAXPY operations. Overall, the evaluation of the interpolant $\hat{z}$ is in $\mathcal{O}(\ell^d \mathcal{N})$. We note that for other applications it has been shown that a direct evaluation of the linear combination (4.2) is faster on modern hardware than first detecting the affected basis functions with a multi-recursive algorithm and then evaluating only those [40]; however, this is not the case for us here, because each point has a vector of size $\mathcal{N}$. Thus, it is worth the effort to first find the affected basis functions and than take only those into account [15].

4.3. Achieving interactive response times

With our sparse-grid-based surrogate model we circumvent the large-scale simulation in the online phase and thus achieve tremendous savings, cf. Section 4.4. However, to achieve interactive response times, i.e., a respond within 0.2 seconds, a straightforward implementation is not sufficient. That is why we consider spatial adaptivity and a parallel implementation here.

The evaluation costs of $\hat{z}$ are in $\mathcal{O}(\ell^d \mathcal{N})$. We cannot change $\mathcal{N}$, because it is given by the simulation, but with spatially adaptive sparse grids we can keep the number of affected basis functions $\mathcal{O}(\ell^d)$ low. We already discussed in Section 2 that we can refine a sparse grid according to the refinement indicator based on the hierarchical coefficients. Adaptive sparse grids do not only reduce the runtime of the online phase but also the runtime of the offline phase because fewer large-scale simulations are required to obtain an interpolant $\hat{z}$ with sufficient accuracy.

Besides adaptivity, an efficient and parallel implementation of the sparse-grid-based surrogate modeling approach is crucial for the interactive exploration. In [16], a parallel CPU implementation of the online phase is proposed. First, the affected basis functions are determined on a single core. The so obtained basis functions are distributed among the other cores of the CPU where the actual SAXPY operations of (4.2) are performed. With an *all gather* operation, the results are combined and finally visualized. We refer to [15, 16] for details.

4.4. Exploration of building information models

BIMs (building information models) give a detailed description of the geometry of a building and additionally auxiliary information such as material parameters or measured information. These BIMs are used to assess buildings with respect to specific objectives such as efficiency of heating, ventilation, and air conditioning. We consider here a building of the Technische Universität München and simulate a flow (e.g., ventilation) through the building. In particular, we want to investigate the effect of closing and opening two doors at the main entrance, see Fig. 5. Thus, the parameters of our BIM are the angles $\gamma_1, \gamma_2 \in [6^\circ, 90^\circ]$ of these two doors and the inflow velocity $v \in [5 \text{ m/s}, 15 \text{ m/s}]$. The flow field is discretized on a fixed Cartesian grid of size $[512 \times 128 \times 128]$ where we store four values at each point (velocity in x_1, x_2, and x_3

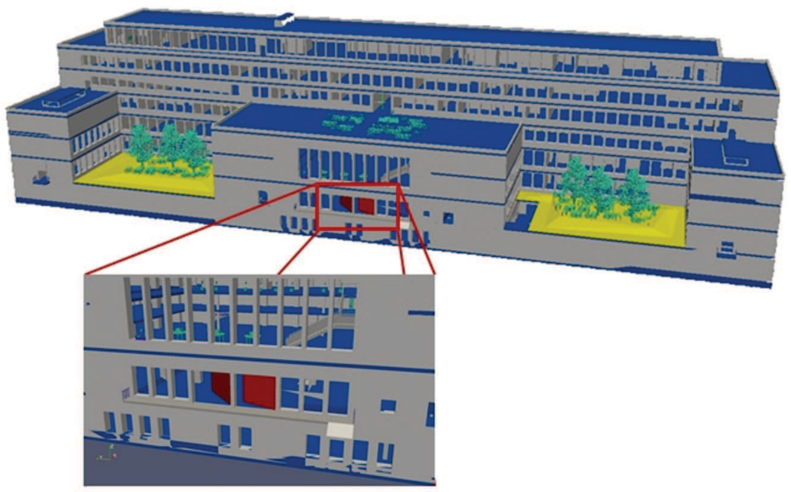
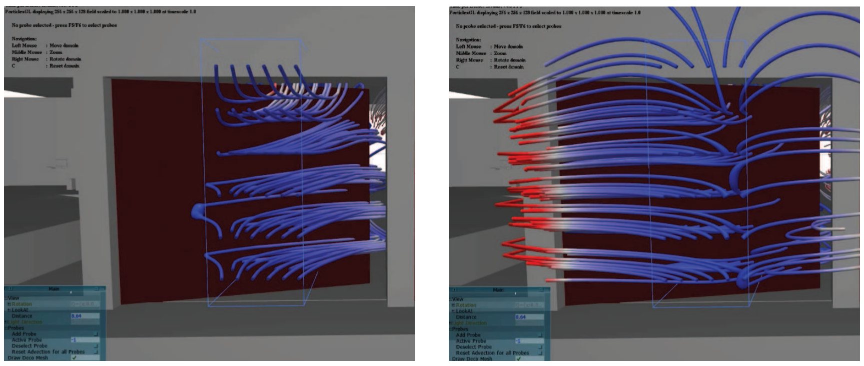


Figure 5: Model of a building on the campus of the Technische Universität München. In the derived building information model, the two doors can be opened and closed.



(a) door is still open (b) door is almost closed

Figure 6: Our sparse-grid-based surrogate model also captures the case where one door is closed and the flow has to enter the building through the other door. This corresponds to a sharp bend in the interpolant $\hat{z}$.

direction, as well as the pressure). Overall, we have about $\mathcal{N} \approx 33$ million degrees of freedom. The simulation is solved with proprietary software, i.e., we do not have access to the underlying equations and so intrusive methods are not an option here. Our approach is non-intrusive and thus applicable. Furthermore, we do not concentrate on only a few outputs of interest (e.g., average velocity) but we reconstruct the whole flow field. Computing one simulation $\boldsymbol{z}(\boldsymbol{\mu})$ for a parameter $\boldsymbol{\mu}$ on several nodes of the KAUST Shaheen[†] supercomputer needs about one hour, and thus the response time would clearly be non-interactive.

[†]http://www.hpc.kaust.edu.sa

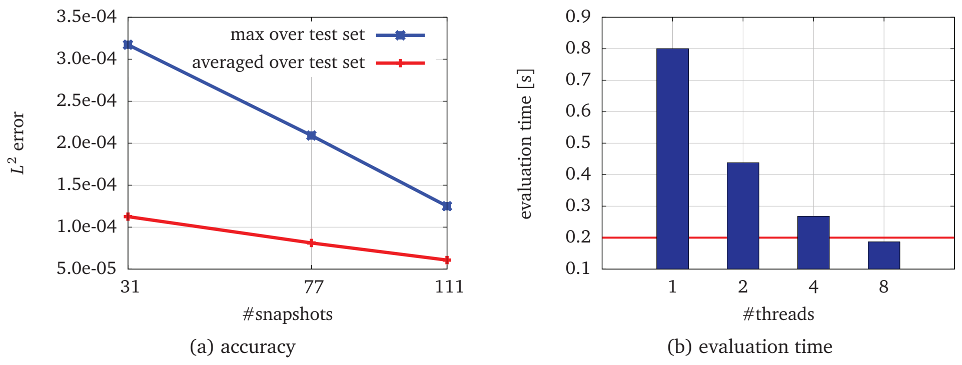


Figure 7: In (a) we plot the accuracy of our sparse-grid-based surrogate model. We achieve accuracies below 10^{-4} which is sufficient for visual exploration. In (b) we report the time needed to evaluate our surrogate model. The evaluation time is below 0.2 seconds (interactive response) if eight threads are employed.

We build our sparse-grid-based surrogate model by starting with a relatively coarse sparse grid of level 3 with 31 grid points only (no boundary points). After two refinement steps we obtain a surrogate model with $N = 111$ grid points, see [15] for details. The refinement indicator relies on the hierarchical coefficients of all $\mathcal{N}$ nodes of the simulation to determine which grid points to refine.

Let us first have a look at the accuracy of our surrogate model. We compare in Fig. 7(a) the averaged and maximal approximation error over a test set $\mathcal{T}$ containing the parameter configurations that match a $5 \times 5 \times 5$ grid in the domain $[16.08, 83.28] \times [16.08, 83.28] \times [6.2, 14.2]$. Note that we do not cover all of $\mathcal{D}$ because special basis functions to better approximate the simulations corresponding to parameters near the boundary of $\mathcal{D}$ are necessary [15]. As shown in Fig. 7, we achieve accuracies below 10^{-4} in both norms which is a sufficient accuracy for visual exploration.

After we made sure that we match the large-scale simulation with sufficient accuracy, we have to discuss the runtime of our surrogate modeling approach. The dominating part of runtime of the offline phase is the computation of the simulation results. We require 111 simulations, needing about 1 hour each, see above. The more interesting runtime for our visual exploration application is the response time in the online phase. We show in Fig. 7(b) the time required to evaluate the surrogate model (4.2) with the implementation discussed in Section 4.3. We evaluate the surrogate model on 1,000 random points and report the averaged result. From the 111 basis functions included in our surrogate model, only 23 are active on average. The plot in Fig. 7 shows that we require eight threads to reach a response time below 0.2 seconds. Compared to the time required to obtain a full simulation z, we achieve a speedup of about 18,000.

5. Data mining: Analysis of simulation data

In the previous section, we discussed an interactive visual exploration approach to investigate the influence of parameters onto simulation results. In this section, we go one step further and automatically extract information from simulation data. For that, we employ a sparse-grid-based data mining method (clustering). We demonstrate our approach on car crash test simulation data.

5.1. Clustering for the analysis of simulation data

The result of a simulation run is a huge pile of data. Only after a careful analysis, one is able to draw conclusions that can eventually get communicated back into the development process. In Section 4, visual exploration was presented as a tool to investigate simulation runs. This exploration can become a tedious task if, for example, one has to browse through several layers of data. That is why we present now data mining methods that take over this exploration task and indicate to the users those parts (e.g., nodes) of the simulation run that might be of interest for a more detailed analysis. Of course, what is to be considered as important highly depends on the context of the application. We focus therefore on car crash test simulations, even though the following methods are readily applicable to other simulations.

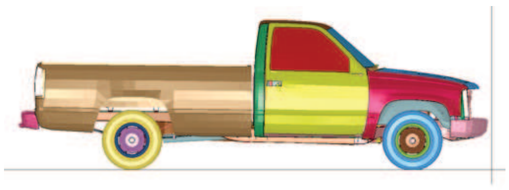
We consider a frontal crash of a Chevrolet pick-up truck, see Fig. 8. To simulate the crash, a finite element model of a truck is created. Each node contains its position in $\mathbb{R}^3$. Overall, the model with $\mathcal{N}$ nodes is described by a matrix $\boldsymbol{X}^0 \in \mathbb{R}^{\mathcal{N}\times 3}$. The actual simulation is performed by proprietary software (LS-Dyna[‡]) at the Fraunhofer SCAI[§]. The result of the simulation are the positions of the nodes $\boldsymbol{X}^{\text{end}} \in \mathbb{R}^{\mathcal{N}\times 3}$ of the car model after the crash. In the following, we want to detect similar moving patterns in the nodes. This is an unsupervised learning task, where we want to detect hidden structure in the data [5]. Therefore, we define the displacements

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_{\mathcal{N}} \end{bmatrix} = \boldsymbol{X}^{\text{end}} - \boldsymbol{X}^0 = \begin{bmatrix} \boldsymbol{x}_1^{\text{end}} \\ \vdots \\ \boldsymbol{x}_{\mathcal{N}}^{\text{end}} \end{bmatrix} - \begin{bmatrix} \boldsymbol{x}_1^0 \\ \vdots \\ \boldsymbol{x}_{\mathcal{N}}^0 \end{bmatrix} \tag{5.1}$$

and consider the rows $\mathcal{S} = \{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_{\mathcal{N}}\}$ of $\boldsymbol{D}$ as the data points. We then cluster $\mathcal{S}$ with respect to the Euclidean distance. The rows, i.e., nodes, that are grouped into one cluster have a similar moving pattern. This and other similar criteria are used in, e.g., [50, 67].

We employ a sparse-grid-based clustering method that is well-suited for large data sets and that automatically detects the number of clusters. Furthermore, our method distinguishes between strong and weak ("noise") clusters. At this point, we emphasize that we employ sparse grids for unsupervised learning here [53, 55]. Sparse grids have also successfully been applied to supervised learning tasks (classification, regression), see, e.g., [26, 61].

[‡]http://www.lstc.com
[§]http://www.scai.fraunhofer.com


(a) Chevrolet truck before crash (X^0)

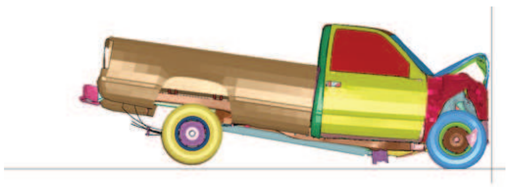

(b) Chevrolet truck after crash (X^{end})

Figure 8: The Chevrolet pick-up truck before and after the crash.

5.2. Density-based clustering with sparse grids

There is a wide variety of clustering methods that are based on very different notions of clustering. We follow a density-based approach where we consider a cluster as a dense region ("where many data points are") surrounded by a low-density region ("where few data points are"). To distinguish between high- and low-density regions, we estimate the probability density function from the data. For other density-based methods we refer to, e.g., DBSCAN [20] and DENCLUE [43].

Let $\mathcal{S} = \{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_{\mathcal{N}}\}$ be the data points (i.e., displacements) as defined in (5.1). Our sparse-grid-based clustering method consists of the following five steps (cf. [56]): First, we estimate the probability density function $\hat{p}$ from the data points $\mathcal{S}$. We consider that step in detail in the next paragraph. Then, we construct a similarity graph $G = (\mathcal{S}, \mathcal{E})$ with vertices $\mathcal{S}$ and edges $\mathcal{E}$ from data $\mathcal{S}$ with respect to the Euclidean distance. The density function $\hat{p}$ is evaluated at each point in $\boldsymbol{d} \in \mathcal{S}$ and if $\hat{p}(\boldsymbol{d})$ is below a threshold value ϵ, it is removed from the set $\mathcal{S}$, and we obtain $\hat{\mathcal{S}}$. We also remove the corresponding edges and obtain a new graph $\hat{G} = (\hat{\mathcal{S}}, \hat{\mathcal{E}}) = (\mathcal{S} \setminus \tilde{\mathcal{S}}, \mathcal{E} \setminus \tilde{\mathcal{E}})$ with k (connected) components. A cluster number in $\{1, \cdots, k\}$ is assigned to each point in $\hat{\mathcal{S}}$ depending on its corresponding component. The result is the set of cluster labels $\{y_{i_1}, \cdots, y_{i_{\hat{\mathcal{N}}}}\} \subseteq \{1, \cdots, k\}$ that assigns each point in $\hat{\mathcal{S}}$ to one cluster. We can now either treat the points in $\tilde{\mathcal{S}} = \mathcal{S} \setminus \hat{\mathcal{S}}$ as noise points (they are in low-density regions), or we train a classifier on the data $\hat{\mathcal{S}}$ with labels $y_{i_1}, \cdots, y_{i_{\hat{\mathcal{N}}}}$ and evaluate at the points in $\tilde{\mathcal{S}}$ to obtain labels for the removed data points.

All the graph operations required for our density-based clustering method can be performed with standard graph libraries (e.g., the BOOST Graph library[¶]). Furthermore, any classification method can be employed in the last (optional) step of the algorithm. However, a crucial component is the density estimation method. For that, we employ a sparse-grid-based method that heavily relies on the idea presented in [39]. Its advantage is that it discretizes the density function on a sparse grid rather than on kernels centered at data points which becomes computationally expensive for large data sets. Let p_ϵ be an initial guess of the estimated density function of $\mathcal{S}$. We are then

[¶]http://www.boost.org

looking for a function $\tilde{p} \in \mathcal{V}$ in a function space $\mathcal{V}$ such that

$$\tilde{p} = \arg\min_{f \in \mathcal{V}} \int_\Omega (f(\boldsymbol{x}) - p_\epsilon(\boldsymbol{x}))^2 \, \mathrm{d}\boldsymbol{x} + \lambda \|\Lambda f\|_{L^2}^2 .$$

where $\|\Lambda \cdot\|_{L^2}^2$ is a regularization term to impose a smoothness constraint on $\tilde{p}$. If we set $p_\epsilon = \frac{1}{N}\sum_i \delta_{\boldsymbol{d}_i}$, where $\delta_{\boldsymbol{d}_i}$ is the Dirac delta function centered at the data point $\boldsymbol{d}_i$, we obtain the variational equations

$$\int_\Omega \tilde{p}(\boldsymbol{x})\psi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} + \lambda \int_\Omega \Lambda\tilde{p}(\boldsymbol{x}) \cdot \Lambda\psi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \frac{1}{M}\sum_{i=1}^{M} \psi(\boldsymbol{x}_i), \quad \psi \in \Psi \tag{5.2}$$

for the test functions $\psi \in \Psi$. We solve (5.2) with Ritz-Galerkin projection onto a sparse grid space $\mathcal{V}^{(1)}$. This can be achieved by solving the system of linear equations

$$(\boldsymbol{R} + \lambda \boldsymbol{C})\boldsymbol{\alpha} = \boldsymbol{b} \,,$$

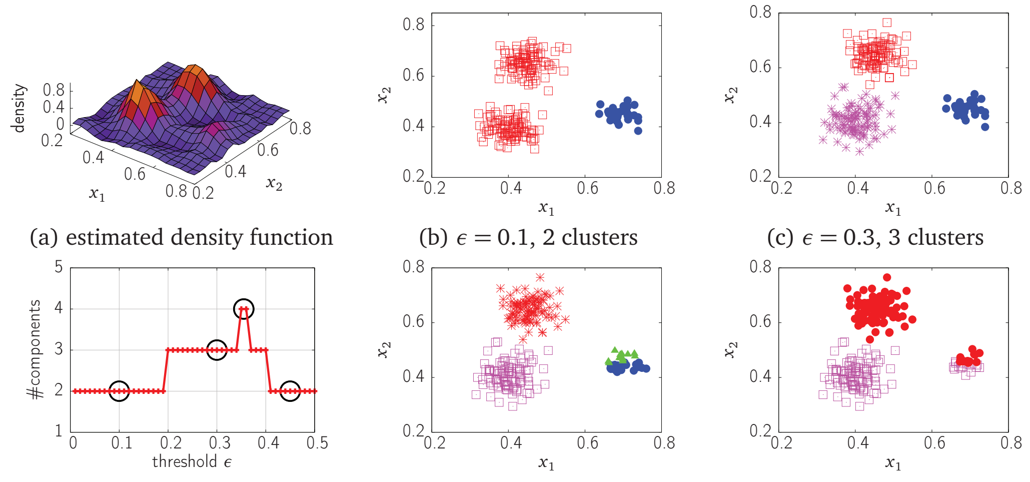
where $R_{ij} = (\phi_i, \phi_j)_{L^2}, C_{ij} = (\Lambda\phi_i, \Lambda\phi_j)_{L^2}$ and the right hand side $b_i = \frac{1}{N}\sum_j \phi_i(\boldsymbol{d}_j)$. For details, and the computational procedure, we refer to [53, 54, 56].

5.3. Indicator for the number of clusters and noise points

Most density-based clustering methods, which directly implement the notion that a cluster is a dense region surrounded by a region with low density, require a parameter that controls what is to be considered a high- and low-density region. In our sparse-grid-based method, this parameter is the threshold ϵ.

Let us consider Fig. 9 to demonstrate its effect on the clustering result. In Fig. 9(a) we see a surface plot of the estimated density function for a data set with three clusters. The density function was estimated with the sparse grid method discussed in Section 5.2. The density function evaluates to higher values at the two clusters on the left than at the cluster on the right. Therefore, we call the two clusters on the left strong clusters, and the one on the right a weak cluster. In Fig. 9(d) we plot the number of connected components versus the threshold ϵ. For $\epsilon < 0.2$, we obtain only two components, because the threshold is set too low and so the two strong clusters on the left are not separated. If we choose $\epsilon > 0.4$, it is too large, and we miss the weak cluster on the right. For the large range between 0.2 and 0.4 we capture all three clusters (three connected components), except for one outlier. The corresponding cluster assignments are plotted in Figs. 9(b)-(c) and 9(e)-(f). Thus, with the threshold ϵ, we can create a graph as in Fig. 9(d) and so determine for what values of the threshold parameter the clusters are present. If they appear for a wide range, we know that the density function has to evaluate to high values and so we consider it as strong cluster. If a cluster is only present for a very narrow range, it probably is just noise and we can ignore it.

However, the threshold cannot only be used to distinguish between strong and weak clusters but also to determine the number of clusters. Therefore, we plot the graph as



(a) estimated density function (b) $\epsilon = 0.1$, 2 clusters (c) $\epsilon = 0.3$, 3 clusters

(d) #components (e) $\epsilon = 0.36$, 4 clusters (f) $\epsilon = 0.45$, 2 clusters

Figure 9: This simple example demonstrates the effect of the threshold ϵ on the clustering. If the threshold is set too low (b) we may miss clusters consisting of many points, if it is set too high (f) we may not be able to capture clusters with just a few points.

in Fig. 9d and look for flat regions. A flat region means that the number of clusters (i.e., the number of connected components) stays constant for a wide range which in turn means all the present clusters have to be strong clusters. This indicator is, for example, used in [6] where it is also shown that it yields similar results as statistically motivated methods.

5.4. Analysis of a frontal crash of a Chevrolet truck

We now use the presented sparse-grid-based clustering method to analyze car crash test simulation data as discussed in Section 5.1. Note that we do not conduct a comparison of our sparse grid clustering method with other clustering methods here. We refer to [56] for such a study.

Let us consider the Chevrolet pick-up truck of Section 5.1 again. First, we are only interested in four beams which are in the front of the truck, see Fig. 10. These beams distinctly influence the crash behavior of the whole truck. We cluster the corresponding $\approx 7,000$ displacements. The density function is approximated on a (non-adaptive) sparse grid of level five with 1,505 grid points (including boundary points). We refer to [56] for details on the parameter selection process and runtime results. Following the indicator discussed in Section 5.3, we mark four flat regions in the plot shown in Fig. 11(a). The corresponding cluster assignments of the nodes of the four beams are shown in Figs. 11(b)-(e). We can verify that the boundaries of the clusters correspond to the nodes at which a different bending behavior occurs, cf. Fig. 10(b). Furthermore, for small ϵ, we have many small clusters in the front of the beams, but they disappear

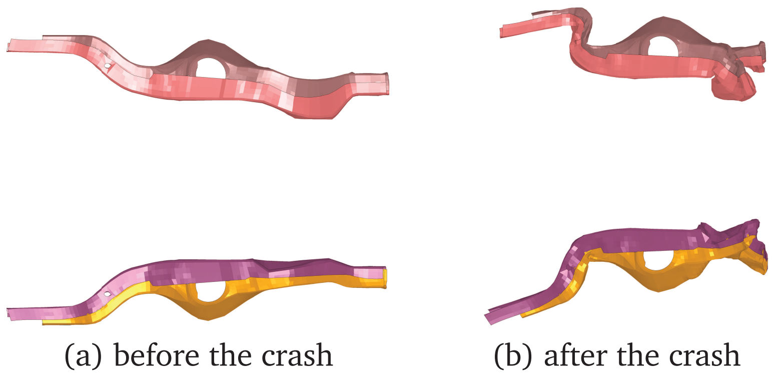


Figure 10: The figures shows four beams where the two on the top and the two on the bottom are connected. Each color indicates a separate beam. They are shown in (a) before the crash and in (b) after the crash, where they are deformed in the rear (left) and in the front (right).

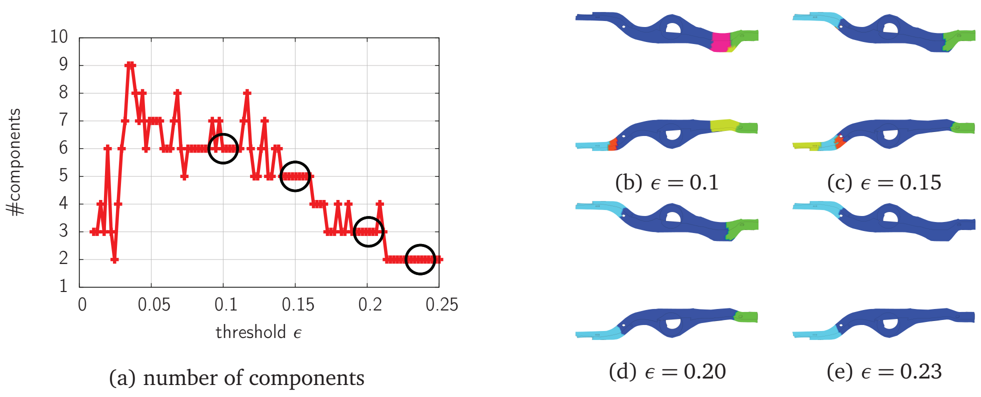
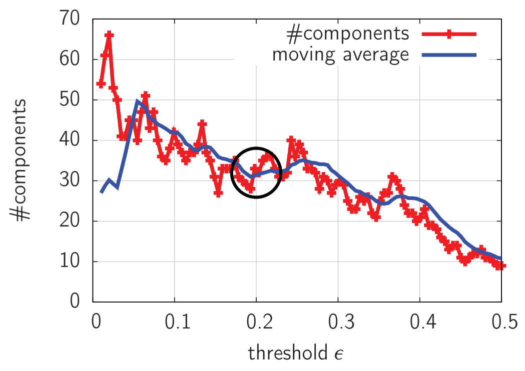


Figure 11: The number of connected components of the similarity graph for the Chevrolet pick-up truck versus the threshold parameter ϵ is shown in (a). The black circles indicate a stable cluster assignment. The corresponding clustering of the beams is shown in (b),(c),(d),(e). Note that the two beams at the top and at the bottom are connected, cf. Fig. 10.

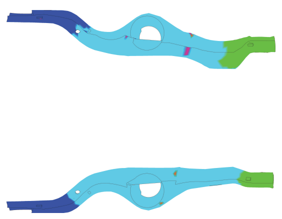
soon when ϵ is increased. Therefore, we consider them as noise. In contrast, the clusters in the rear of the beams are present for a wide range of ϵ and so this suggests that these are strong clusters that heavily contribute to the crash behavior of the truck.

We now cluster the displacements of all nodes of the truck. Again, we do not know the number of clusters and therefore first plot the number of connected components versus the threshold ϵ, see Fig. 12(a). With the help of a moving average, we find a flat region near $\epsilon = 0.2$ and plot the corresponding cluster assignment of the four beams in Fig. 12(b). We obtain a similar clustering as before which confirms that the indicator for the number of clusters yields reasonable results.

We have shown that we can automatically detect moving patterns in the nodes of the car model. This allows us to analyze simulation data in an unsupervised fashion. Details of this approach can be found in [6]. Our sparse grid method allows us to



(a) number of components



(b) four beams, $\epsilon = 0.2$

Figure 12: In (a) we show the number of components versus the threshold ϵ for the whole car, and in (b) the resulting clustering of the four beams with threshold $\epsilon = 0.2$. Note that the two beams at the top and at the bottom are connected, cf. Fig. 10.

process large data sets because the employed density estimation method discretizes the density function on a sparse grid rather than on kernels centered at the data points, cf. [56].

6. Eigenvalue problems: Plasma turbulence simulation

Sparse grids with their hierarchical basis are in some cases hard to deploy on existing simulation software, since it would require a major code redesign. The sparse grid combination technique, in contrast, can directly use existing code which is handling regular Cartesian grids. In this section the combination technique is used to decrease the computational effort of computations related to plasma microturbulence.

6.1. The gyrokinetic eigenvalue problem

Achieving a positive and economically sound energy balance by magnetically confined nuclear fusion can be one remedy to the increasing demand for energy in the 21st century. One milestone to this ambitious goal is the successful operation of the fusion experiment ITER, which is a tokamak [68] currently being built in southern France. But its success also depends on extensive numerical simulation [44] for the operation of the experiment on the one hand and for understanding and interpreting its outcomes on the other hand. The efficiency of the magnetic confinement in such a tokamak is determined by the transport of heat and particles out of the hot core zone of the plasma. The transport is a result of small scale turbulent effects in the plasma. These microturbulences are driven by imminent gradients of temperature and density and the resulting transport is also often referred to as *anomalous transport* since it is much larger than predicted by classical transport models.

Numerical simulations are used to quantitatively and qualitatively understand the phenomenon of microturbulence. Macroscopic models of describing a plasma like magneto-hydrodynamics cannot be applied for the simulation of microturbulence and thus a rather microscopic model is applied. It is based on the Vlasov-Maxwell equation

$$\frac{\partial g_s}{\partial t} + \boldsymbol{v}\frac{\partial g_s}{\partial \boldsymbol{x}} + \left(\frac{q_s}{m_s}\left(\boldsymbol{E}(g_s) + \boldsymbol{v}\times\boldsymbol{B}(g_s)\right)\right)\frac{\partial g_s}{\partial \boldsymbol{v}} = \mathit{\Delta}(g_s)\;, \tag{6.1}$$

where $\boldsymbol{E}$ and $\boldsymbol{B}$ correspond to electric and magnetic fields and g_s represents the distribution function of a certain plasma species s (either certain ions or electrons) in the plasma. The normalized distribution function $g_s(\boldsymbol{x},\boldsymbol{v},t)$ gives the probability to find a particle of species s at time t at position $(\boldsymbol{x},\boldsymbol{v})$ in the 6D phase space. The constant q_s and m_s give the charge and mass of a single particle of species s. Whereas the advection of particles through phase space is governed on the left-hand side of (6.1), the right-hand side governs their collision. This collision term can be neglected, since collisions only have a minor influence on the dynamics of hot fusion plasmas. The electromagnetic fields $\boldsymbol{E}$ and $\boldsymbol{B}$ have to be calculated by solving the Maxwell equations self-consistently using moments of the distribution function itself, thus making (6.1) a nonlinear problem.

For the simulation of microturbulence, the direct application of the Vlasov-Maxwell equation is not suited. In a homogeneous magnetic field, charged particles rotate (gyrate) around a guiding center, which can move freely parallel to the magnetic field. Since the magnetic field is, despite small perturbations, nearly homogeneous on the microscale, the gyration is preserved and only slow drifts of the guiding centers influence the dynamics of the plasma. Simulations using (6.1) would fully resolve the gyration by using small time- and spatial scales, whereas microturbulence is driven by much slower drifts. In gyrokinetics, (6.1) is transformed to not resolve the gyration anymore and to align the model parallel to the magnetic field. The spatial coordinates of the distribution function are then not $\boldsymbol{x}$ and $\boldsymbol{v}$ anymore but only $\boldsymbol{X}$, the position of the guiding center, $v_\parallel$, the velocity of the guiding center parallel to the magnetic field, and the magnetic moment $\mu = m_s v_\perp^2/2B_\parallel$, which is representing the velocity of gyration. The position of the particles is thus not fully resolved anymore. The equations describing the advection and collision of this five-dimensional problem are the gyrokinetic equations, which take the simplified form

$$\frac{\partial g_s}{\partial t} + \tilde{\boldsymbol{v}}\frac{\partial g_s}{\partial \boldsymbol{x}} + \tilde{\boldsymbol{F}}\frac{\partial g_s}{\partial v_\parallel} = \Delta(g_s)\;, \tag{6.2}$$

with $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{F}}$ being rather complex expressions describing the transformed drift velocities and electromagnetic forces respectively. A full representation of the gyrokinetic equation is out of scope here and an extensive derivation can be found in [7]. Since (6.2) is still coupled to the Maxwell equations, the equation is again nonlinear. The equation can be separated into a linear operator L and a non-linear operator N acting on the five-dimensional distribution function g_s to get

$$\frac{\partial g_s}{\partial t} = \mathrm{L}(g_s) + \mathrm{N}(g_s)\;. \tag{6.3}$$

Interestingly, the linear operator L already models the microinstabilities of the plasma which are driving the microturbulence. Discretizing the distribution function $\boldsymbol{g}_s$ using spectral methods and finite differences, the linear operator can be expressed as a matrix $\mathbf{L}$ and the linear discretized gyrokinetic equation as

$$\frac{\partial \boldsymbol{g}}{\partial t} = \mathbf{L}\boldsymbol{g} \, . \tag{6.4}$$

Since the problem is moderately high dimensional, even the coarsest useful resolutions require a size of $\boldsymbol{g}$ of a few hundreds of thousands unknowns.

The eigenvalue spectrum of $\mathbf{L}$ exhibits mostly eigenvalues with a non-positive real part [66]. These are the ones representing damped or stable modes in the plasma. There are only a few eigenvalues with positive real parts and those are the ones representing the unstable modes driving the microturbulence. Knowing the growth rates of these modes allows an estimation of the turbulent transport even without full non-linear simulations [2].

The GENE code[‖] is a mature and well documented code for simulations and computations in linear and non-linear gyrokinetics. It is developed in the IPP (Max-Planck Institute for Plasma Physics) in Garching, Germany, widely used in the fusion community and has been proved to be an HPC application [49].

GENE can solve the gyrokinetic eigenvalue problem using SLEPc [42]. Solving the eigenvalue problem in GENE is time consuming. The performance of the eigenvalue computation is diminished if finer grids are used since the computational complexity of the parallel Jacobi-Davidson eigenvalue solver is worse than linear in the number of grid points if the problem is distributed across large numbers of processors [51, 66]. Thus we like to reduce the overall amount of grid points of the five-dimensional problem using the sparse grid combination technique.

6.2. Combination technique for the gyrokinetic eigenvalue problem

The sparse grid combination technique has been applied to eigenvalue problems before in the context of the Schrödinger equation [29]. In the classical combination technique approach, the eigenvalue problem is solved on each of the grids $\mathcal{V}_l$ and the eigenvectors are combined according to (2.11). The corresponding eigenvalues could then be computed using the Rayleigh quotient [27]. Different problems can arise: For example, changing the resolution of the underlying problem can lead to a change in the ordering of the eigenvalues, so that they have to be identified comparing their corresponding eigenvectors [25]. Another problem is that this approach requires a consistent scaling of the eigenvectors in order to apply the combination technique. Furthermore, the gyrokinetic eigenvalue problem is neither symmetric nor Hermitian and thus computing the eigenvalues using the Rayleigh quotient is not possible. A straightforward direct combination is also not possible because the eigenvectors are

[‖]http://www.ipp.mpg.de/˜fsj/gene/

rotated in the complex plane. That is why we consider here the optimized combination technique (Opticom) as introduced in [26, 38].

The Opticom was developed to improve the approximation quality of the classical combination technique. The combination coefficients $\boldsymbol{c}$ in

$$\hat{f}^C = \sum_{\boldsymbol{l}\in\mathcal{L}} c_{\boldsymbol{l}} \hat{f}_{\boldsymbol{l}} \tag{6.5}$$

are computed to suit the underlying problem, cf. Section 2 and (2.11). The active set $\mathcal{L}$ contains all n different $\boldsymbol{l}$, i.e. all n full grids used for combination. In the Opticom, the optimization of $\boldsymbol{c}$ can be done if the problem is accessible in a suited Galerkin formulation and the operator norm is at hand. The method was later extended to solve eigenvalue problems. There, all full grid approximations $\hat{f}_{\boldsymbol{l}}$ are used as a basis, in which a generalized eigenvalue problem is solved in a Galerkin formulation again [27].

We follow here another approach where the minimization problem corresponding to the Opticom is modified to especially take the gyrokinetic eigenvalue problem into account. To find the optimal coefficients, the Opticom minimizes the functional

$$J(\boldsymbol{c},\lambda) = \|\mathbf{L}\boldsymbol{g}^C - \lambda\boldsymbol{g}^C\|^2 = \|(\mathbf{LG} - \lambda\mathbf{G})\boldsymbol{c}\|^2\,, \tag{6.6}$$

with $\boldsymbol{g}^C = \sum_{\boldsymbol{l}\in\mathcal{L}} P_{\boldsymbol{l}} g_{\boldsymbol{l}} = \mathbf{G}\boldsymbol{c}$ being the combined eigenvector and λ being the approximation of the correct eigenvalue λ_0 [48]. The matrix $\mathbf{G}$ is containing the eigenvectors $\boldsymbol{g}_{\boldsymbol{l}}$ in its columns such that

$$\mathbf{G} = \left[P_{\boldsymbol{l}_1}\boldsymbol{g}_{\boldsymbol{l}_1} \;\cdots\; P_{\boldsymbol{l}_n}\boldsymbol{g}_{\boldsymbol{l}_n}\right]\,, \tag{6.7}$$

with P being the operator which projects the n approximations of the eigenvector from the small full grid spaces with level vector $\boldsymbol{l}_i$ into the space corresponding to the grid of the $\boldsymbol{g}^C$. Minimizing the functional J by least-squares minimization leads to

$$[(\mathbf{LG} - \lambda\mathbf{G})^*(\mathbf{LG} - \lambda\mathbf{G})]\boldsymbol{c} = \mathbf{K}(\lambda)\boldsymbol{c} = 0\,, \tag{6.8}$$

which is has to be fulfilled so that the combination coefficients $\boldsymbol{c}$ minimize the functional $J(\boldsymbol{c},\lambda)$ for a fixed λ.

To obtain a λ close to the exact eigenvalue λ_0, (6.8) can be interpreted as an eigenvalue problem again, since a $\lambda = \lambda_0$ results in a singular $\mathbf{K}$. The retrieved eigenvector is containing the sparse grid combination coefficients determining the combined eigenvector $\boldsymbol{g}^C$. A method for solving such eigenvalue problems [36, 52] is to embed the original eigenvalue problem in the slightly enlarged system

$$\begin{pmatrix} \mathbf{K}(\lambda) & \boldsymbol{x} \\ \boldsymbol{s}^* & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{6.9}$$

with nearly arbitrary non-zero vectors $\boldsymbol{x}$ and $\boldsymbol{s}$ of size n, with $\boldsymbol{s}$ giving a scaling of the obtained eigenvector. Solving the system is retrieving $\boldsymbol{c}$ which minimizes J for the given λ. The scalar β is indicating how close the chosen λ is to the actual eigenvalue λ_0, since for a singular $\mathbf{K}$, β is zero. In this case, $\boldsymbol{c}$ represents the combination coefficients to obtain the eigenvector to the eigenvalue λ_0.

Now, to obtain the correct λ, the Newton method can be employed, if a sufficiently close initial guess is at hand. In our case, the initial guess can be obtained by a combination using the classical combination coefficients given by (2.11). From this initial guess the iteration

$$\lambda^{(i+1)} = \lambda^{(i)} - \frac{\boldsymbol{s}^*\boldsymbol{c}}{\boldsymbol{s}^*\mathbf{K}(\lambda_i)^{-1}\frac{\mathrm{d}\mathbf{K}}{\mathrm{d}\lambda}\boldsymbol{c}} \tag{6.10}$$

is used. Solving the linear system in the denominator can be numerically demanding. But since the dimension, i.e., number of unknowns, of this system is the number of full grids used in the combination technique, it is rather small compared to the system matrix $\mathbf{L}$. For our problem the Newton iteration might not always converge, since the $\beta(\lambda)$ might not have a root because the chosen basis $\mathbf{G}$ might not be sufficient to exactly represent the eigenvector of $\mathbf{L}$ and thus $\mathbf{K}$ can not get singular. We then fall back to a gradient descent method using the gradient

$$\frac{\partial\beta}{\partial\lambda} = \beta\frac{\boldsymbol{s}^*\mathbf{K}(\lambda_i)^{-1}\frac{\mathrm{d}\mathbf{K}}{\mathrm{d}\lambda}\boldsymbol{c}}{\boldsymbol{s}^*\boldsymbol{c}}. \tag{6.11}$$

6.3. Results

First studies for the above algorithm have been performed to demonstrate its applicability to the gyrokinetic eigenvalue problem [47,48,51]. It was tested if the classical combination of the eigenvalues $\lambda_{\boldsymbol{l}}$ computed with GENE is retrieving a good initial guess for the application of the Opticom described in the previous section. Combining the computed eigenvalues $\lambda_{\boldsymbol{l}}$ of several full grids of resolution $\boldsymbol{l}$ actually retrieves a rather close approximation of the reference eigenvalue computed by a full resolution computation by GENE [47]. The combination behavior can be seen in Fig. 13. The plots show that the combined eigenvalues give in most cases a better approximation than the approximation of the eigenvalue created by each of the grids used for combination.

The Opticom for the computation of an eigenpair has so far been studied for retrieving the eigenfunction of the ODE [48]

$$\frac{\partial u}{\partial t} = \lambda u \tag{6.12}$$

in the unit interval $\Omega = [0,1]$. It has the eigenfunction and eigenvalues

$$u(t) = \mathrm{e}^{\lambda_0 t}, \quad \lambda_0 = 2\pi k\mathrm{i}, \quad k \in \mathbb{Z}\,. \tag{6.13}$$

Since it contains a first derivative, it can serve as the simplest model for which the method has to work in order to be applicable to the gyrokinetic eigenvalue problem. This eigenfunction can be retrieved using three semi-coarsened approximations [22] u_1, u_2 and u_3 of the eigenfunction, which are depicted in Fig. 14. The Opticom can retrieve the combination coefficients with the Newton iteration (6.10) as well as using

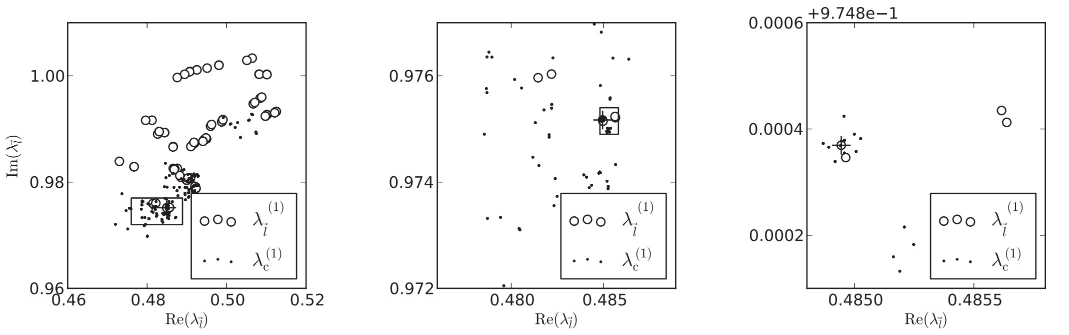


Figure 13: The combined eigenvalues λ^C (*black dots*) for different active sets. Each of the plots shows the section marked in the plot left of it. Combined eigenvalues are usually a better approximation of the correct solution (*cross*) than the eigenvalues $\lambda_{\mathbf{l}}$ which are computed by GENE and used for the combination (*circles*) [47].

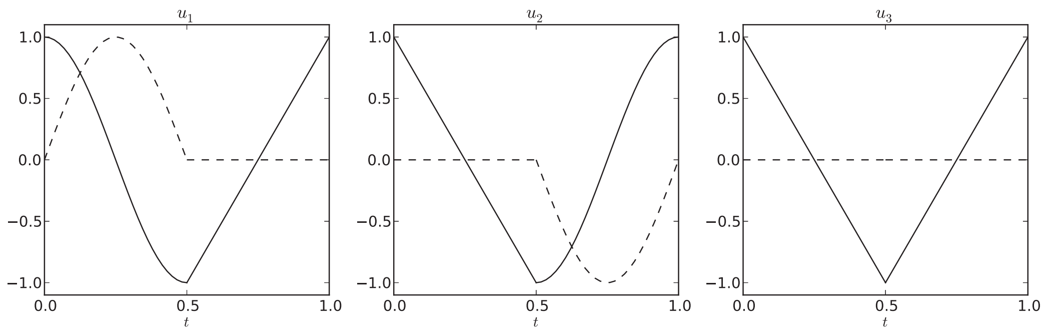


Figure 14: The three semi-coarsened functions used for the combination of the eigenfunction represented by their real (*solid line*) and imaginary (*dashed line*) parts [48].

the gradient-descent approach (6.11). Even when scaled functions are used, the algorithm retrieves the correct eigenfunction. The results so far show that the approach is applicable to other large scale gyrokinetic eigenvalue calculations using GENE, since the computational effort can be regarded as small compared to retrieving the eigenpair using the Jacobi-Davidson algorithm in GENE.

7. Conclusions

In this contribution, we presented four recent real-world applications to show that sparse grids are widely used to tackle today's problems in computational science and engineering: We considered basket option pricing where we discretized the multi-dimensional Black-Scholes equation on a sparse grid. A key ingredient was spatial adaptive refinement based on the hierarchical coefficients which allowed us to place grid points only were they are needed, i.e., near the locally non-smooth region of the payoff function. We then looked into interactive visual exploration with sparse-grid-based surrogate model that also required adaptive refinement to minimize the number of large-scale simulations for the construction of the surrogate model. With a sparse grid clustering method, we analyzed car crash test simulation data and showed that the underlying non-parametric sparse grid density estimation method yields a clustering method that allows us to distinguish between strong and weak clusters. Finally, we considered eigenvalue problems arising in the context of plasma turbulence simulation. There, the combination technique was employed to reuse available plasma problem solvers.

References

[1] S. Achatz. *Adaptive finite Dünngitter-Elemente höherer Ordnung für elliptische partielle Differentialgleichungen mit variablen Koeffizienten*. PhD thesis, Technische Universität München, 2003.

[2] C. Angioni, A. Peeters, F. Jenko, and T. Dannert. Collisionality dependence of density peaking in quasilinear gyrokinetic calculations. *Physics of Plasmas*, 12(11), 2005.

[3] R. Balder. *Adaptive Verfahren für elliptische und parabolische Differentialgleichungen auf dünnen Gittern*. PhD thesis, Technische Universität München, 1994.

[4] J. Benk and D. Pflüger. Hybrid parallel solutions of the Black-Scholes PDE with the truncated combination technique. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 678–683, 2012.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[6] B. Bohn, J. Garcke, R. Iza-Teran, A. Paprotny, B. Peherstorfer, U. Schepsmeier, and C.-A. Thole. Analysis of car crash simulation data with nonlinear machine learning methods. In *Proceedings of the International Conference on Computational Science, ICCS 2013*, 2013.

[7] A. Brizard and T. Hahm. Foundations of nonlinear gyrokinetic theory. *Reviews of Modern Physics*, 79(2):421–468, 2007.

[8] H.-J. Bungartz. *Finite Elements of Higher Order on Sparse Grids*. Habilitationsschrift, Technische Universität München, 1998.

[9] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:1–123, 2004.

[10] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Pointwise convergence of the combination technique for Laplace's equation. *East-West J. Numer. Math*, 2:21–45, 1994.

[11] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Two proofs of convergence for the combination technique for the efficient solution of sparse grid problems. In D. E. Keyes and J. Xu, editors, *Domain Decomposition Methods in Scientific and Engineering Computing, DDM7*, Contemp. Math. 180, pages 15–20. American Mathematical Society, 1994.

[12] H.-J. Bungartz, M. Griebel, and U. Rüde. Extrapolation, combination, and sparse grid techniques for elliptic boundary value problems. *Comput. Methods Appl. Mech. Engrg*, 116:243–252, 1994.

[13] 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*, 236(15):3741–3750, 2011.

[14] H.-J. Bungartz, A. Heinecke, D. Pflüger, and S. Schraufstetter. Parallelizing a Black-Scholes solver based on finite elements and sparse grids. *Concurrency and Computation: Practice and Experience*, 2012.

[15] D. Butnaru. *Computational Steering with Reduced Complexity*. PhD thesis, Technische Universität München, 2013.

[16] D. Butnaru, G. Buse, and D. Pflüger. A parallel and distributed surrogate model implementation for computational steering. In *Proceeding of the 11th International Symposium on Parallel and Distributed Computing - ISPDC 2012*. IEEE, 2012.

[17] D. Butnaru, B. Peherstorfer, D. Pflüger, and H.-J. Bungartz. Fast insight into high-dimensional parametrized simulation data. In *11th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2012.

[18] D. Butnaru, D. Pflüger, and H.-J. Bungartz. Towards high-dimensional computational steering of precomputed simulation data using sparse grids. In *Proceedings of the International Conference on Computational Science (ICCS) 2011*, volume 4 of *Procedia CS*, pages 56–65. Tsukaba, Japan, Springer, 2011.

[19] T. Dijkema, C. Schwab, and R. Stevenson. An adaptive wavelet method for solving high-dimensional elliptic PDEs. *Constructive Approximation*, 30(3):423–455, 2009.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[21] G. Faber. Über stetige Funktionen. *Mathematische Annalen*, 66(1):81–94, 1908.

[22] Y. Fang. *The combination approximation method*. PhD thesis, Australian National University, 2012.

[23] C. Feuersänger. Dünngitterverfahren für hochdimensionale elliptische partielle Differentialgleichungen. Diplomarbeit, Institut für Numerische Simulation, Universität Bonn, 2005.

[24] C. Feuersänger. *Sparse Grid Methods for Higher Dimensional Approximation*. PhD thesis, Institut für Numerische Simulation, Universität Bonn, 2010.

[25] J. Garcke. Berechnung von Eigenwerten der stationären Schrödingergleichung mit der Kombinationstechnik. Diplomarbeit, Institut für Angewandte Mathematik, Universität Bonn, 1998.

[26] J. Garcke. Regression with the optimised combination technique. In W. Cohen and A. Moore, editors, *Proceedings of the 23rd ICML '06*, pages 321–328. ACM Press, 2006.

[27] J. Garcke. An optimised sparse grid combination technique for eigenproblems. *PAMM*, 7(1):1022301–1022302, 2007.

[28] 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.

[29] J. Garcke and M. Griebel. On the parallelization of the sparse grid approach for data mining. In S. Margenov, J. Wasniewski, and P. Yalamov, editors, *Large-Scale Scientific Computations, Third International Conference, LSSC 2001*, volume 2179 of *Lecture Notes in Computer Science*, pages 22–32. Springer, 2001.

[30] J. Garcke, M. Griebel, and M. Thess. Data mining with sparse grids. *Computing*, 67(3):225–253, 2001.

[31] M. Giles. Multilevel Monte Carlo path simulation. *Operations Research*, 56:607–617, 2008.

[32] P. Glasserman. *Monte Carlo methods in financial engineering*. Springer, 2004.

[33] M. Griebel and M. Holtz. Dimension-wise integration of high-dimensional functions with applications to finance. *J. Complexity*, 26:455–489, 2010.

[34] M. Griebel and P. Oswald. On additive Schwarz preconditioners for sparse grid discretiza-

tions. *Numerische Mathematik*, 66(1):449–463, 1993.

[35] 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.

[36] D. Harrar II and M. Osborne. Computing eigenvalues of ordinary differential equations. *ANZIAM Journal*, 44(April):C313–C334, 2003.

[37] M. Hegland. Adaptive sparse grids. In K. Burrage and R. B. Sidje, editors, *Proc. of 10th Computational Techniques and Applications Conference CTAC-2001*, volume 44, pages C335–C353, 2003.

[38] M. Hegland, J. Garcke, and V. Challis. The combination technique and some generalisations. *Linear Algebra and its Applications*, 420(2–3):249–275, 2007.

[39] M. Hegland, G. Hooker, and S. Roberts. Finite element thin plate splines in density estimation. *ANZIAM Journal*, 42:C712–C734, 2000.

[40] A. Heinecke and D. Pflüger. Emerging architectures enable to boost massively parallel data mining using adaptive sparse grids. *International Journal of Parallel Programming*, 41(3):357–399, 2013.

[41] A. Heinecke, S. Schraufstetter, and H.-J. Bungartz. A highly-parallel Black-Scholes solver based on adaptive sparse grids. *International Journal of Computer Mathematics*, 89(9):1212–1238, 2012.

[42] V. Hernandez, J. E. Roman, and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Software*, 31(3):351–362, 2005.

[43] A. Hinneburg and H.-H. Gabriel. Denclue 2.0: Fast clustering based on kernel density estimation. In M. Berthold, J. Shawe-Taylor, and N. Lavrac, editors, *Advances in Intelligent Data Analysis VII*, volume 4723 of *Lecture Notes in Computer Science*, pages 70–80. Springer, 2007.

[44] F. Jenko, D. Told, T. Görler, J. Citrin, A. Navarro, C. Bourdelle, S. Brunner, G. Conway, T. Dannert, H. Doerk, D. Hatch, J. Haverkort, J. Hobirk, G. Hogeweij, P. Mantica, M. Pueschel, O. Sauter, L. Villard, E. Wolfrum, and the ASDEX Upgrade Team. Global and local gyrokinetic simulations of high-performance discharges in view of ITER. *Nuclear Fusion*, 53(7):073003, 2013.

[45] A. Klimke. Sparse Grid Interpolation Toolbox – user's guide. Technical Report IANS report 2007/017, University of Stuttgart, 2007.

[46] A. Klimke and B. Wohlmuth. Algorithm 847: spinterp: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB. *ACM Transactions on Mathematical Software*, 31(4), 2005.

[47] C. Kowitz and M. Hegland. The sparse grid combination technique for computing eigenvalues in linear gyrokinetics. In V. N. Alexandrov, M. Lees, V. V. Krzhizhanovskaya, J. Dongarra, and P. M. A. Sloot, editors, *ICCS*, volume 18 of *Procedia Computer Science*, pages 449–458. Elsevier, 2013.

[48] C. Kowitz and M. Hegland. An opticom method for computing eigenpairs. In J. Garcke and D. Pflüger, editors, *Sparse Grids and Applications - Munich 2012*, volume 97 of *Lecture Notes in Computational Science and Engineering*, pages 239–253. Springer, 2014.

[49] H. Lederer, R. Tisma, and R. Hatzky. Application enabling in deisa: Petascaling of plasma turbulence codes. *Parallel Computing: Architectures, Algorithms and Applications*, 2008.

[50] L. Mei and C.-A. Thole. Data analysis for parallel car-crash simulation results and model optimization. *Sim. Modelling Practice and Theory*, 16(3):329–337, 2008.

[51] F. Merz, C. Kowitz, E. Romero, J. Roman, and F. Jenko. Multi-dimensional gyrokinetic parameter studies based on eigenvalue computations. *Computer Physics Communications*,

183(4):922–930, 2012.

[52] M. R. Osborne. A new method for the solution of eigenvalue problems. *The Computer Journal*, 7(3):228–232, 1964.

[53] B. Peherstorfer. *Model Order Reduction of Parametrized Systems with Sparse Grid Learning Techniques*. PhD thesis, Technische Universität München, 2013.

[54] B. Peherstorfer, D. Pflüge, and H. Bungartz. Density estimation with adaptive sparse grids for large data sets. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 443–451. SIAM, 2014.

[55] B. Peherstorfer, D. Pflüger, and H.-J. Bungartz. A sparse-grid-based out-of-sample extension for dimensionality reduction and clustering with Laplacian eigenmaps. In D. Wang and M. Reynolds, editors, *AI 2011: Advances in Artificial Intelligence*, volume 7106 of *Lecture Notes in Computer Science*, pages 112–121. Springer, 2011.

[56] B. Peherstorfer, D. Pflüger, and H.-J. Bungartz. Clustering based on density estimation with sparse grids. In *KI 2012: Advances in Artificial Intelligence*, volume 7526 of *Lecture Notes in Computer Science*. Springer, 2012.

[57] B. Peherstorfer, S. Zimmer, and H.-J. Bungartz. Model reduction with the reduced basis method and sparse grids. In J. Garcke and M. Griebel, editors, *Sparse Grids and Applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*. Springer, 2013.

[58] T. Petersdorff and C. Schwab. Sparse finite element methods for operator equations with stochastic data. *Applications of Mathematics*, 51(2):145–180, 2006.

[59] C. Pflaum. *Diskretisierung elliptischer Differentialgleichungen mit dünnen Gittern*. PhD thesis, Technische Universität München, 1996.

[60] C. Pflaum and A. Zhou. Error analysis of the combination technique. *Numerische Mathematik*, 84(2):327–350, 1999.

[61] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, 2010.

[62] D. Pflüger. Spatially adaptive refinement. In J. Garcke and M. Griebel, editors, *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering, pages 243–262. Springer, 2012.

[63] 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.

[64] R. Rannacher. Finite element solution of diffusion problems with irregular data. *Numerische Mathematik*, 43:309–327, 1984.

[65] C. Reisinger and G. Wittum. Efficient hierarchical approximation of high-dimensional option pricing problems. *SIAM Journal on Scientific Computing*, 29(1):440–458, 2007.

[66] J. E. Roman, M. Kammerer, F. Merz, and F. Jenko. Fast eigenvalue calculations in a massively parallel plasma turbulence code. *Parallel Computing*, 36(5-6):339–358, 2010.

[67] A. Stork, C.-A. Thole, S. Klimenko, I. Nikitin, L. Nikitina, and Y. Astakhov. Towards interactive simulation in automotive design. *The Visual Computer*, 24(11):947–953, 2008.

[68] J. Wesson. *Tokamaks*. Oxford University Press, third edition, 2004.

[69] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49(4):379–412, 1986.

[70] A. Zeiser. Fast matrix-vector multiplication in the sparse-grid Galerkin method. *Journal of Scientific Computing*, 47(3):328–346, 2011.

[71] 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. Vieweg, 1991.