Numer Algor (2018) 78:929–956
DOI 10.1007/s11075-017-0407-9

ORIGINAL PAPER

A prewavelet-based algorithm for the solution of second-order elliptic differential equations with variable coefficients on sparse grids

Rainer Hartmann[1] · **Christoph Pflaum**[1]

Received: 12 April 2017 / Accepted: 22 August 2017 / Published online: 2 September 2017

Abstract We present a Ritz-Galerkin discretization on sparse grids using prewavelets, which allows us to solve elliptic differential equations with variable coefficients for dimensions $d \geq 2$. The method applies multilinear finite elements. We introduce an efficient algorithm for matrix vector multiplication using a Ritz-Galerkin discretization and semi-orthogonality. This algorithm is based on standard *1*-dimensional restrictions and prolongations, a simple prewavelet stencil, and the classical operator-dependent stencil for multilinear finite elements. Numerical simulation results are presented for a three-dimensional problem on a curvilinear bounded domain and for a six-dimensional problem with variable coefficients. Simulation results show a convergence of the discretization according to the approximation properties of the finite element space. The condition number of the stiffness matrix can be bounded below 10 using a standard diagonal preconditioner.

Keywords Sparse grid · Prewavelets · Semi-orthogonality · Variable coefficients · Conjugate gradient method · Finite element method

1 Introduction

A finite element discretization of an elliptic symmetric partial differential equation (PDE) calculates the best approximation with respect to the energy norm. Since a

✉ Rainer Hartmann
rainer.hartmann@fau.de

1 Department of Computer Science, System Simulation, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany

finite element method uses polynomials to construct a finite element space, the convergence of such a method can be proven using Strang's lemma or the Lax-Milgram theorem.

However, difficulties arise in the application to problems of a high dimensionality. Then, the computational amount increases by $O(N^d)$, where N is the number of grid points in one direction and d is the dimension of the space. This exponential growth of the computational amount restricts the application of the finite element method to dimensions $d \leq 3$.

One approach to solve this problem is to use sparse grids (see [21]). With sparse grids, one can construct a subspace of the classical finite element spaces on full grids. The dimension of this subspace reduces to $O(N(\log N)^{d-1})$.

There exist several methods to solve partial differential equations on sparse grids. Here, we restrict ourselves to elliptic PDEs and Ritz-Galerkin discretizations. Such discretizations lead to optimal convergence in the energy norm and allow extensions to adaptive grids, which are needed in case of singularities.

However, it is difficult to solve iteratively the system of linear equations resulting from a Ritz-Galerkin discretization on sparse grids. An efficient algorithm for this problem has the following properties:

– a memory complexity of $O(N(\log N)^{d-1})$,
– requires only $O(N(\log N)^{d-1})$ computations,
– a convergence rate of $O(1)$.

This algorithm may be based on the unidirectional principle (see [2, 4] and [20]) for constant coefficients and cubical domains. An extension of this algorithm for tensor product variable coefficients is presented in [11]. These algorithms evaluate the matrix vector multiplication in $O(N(\log N)^{d-1})$ operations. However, these algorithms cannot be applied to arbitrary variable coefficients. Furthermore, it is difficult to obtain an iterative solver with convergence rate $O(1)$. In order to obtain fast convergence, one can apply a multigrid approach or prewavelets (see [14] and [10]). The algorithms described in these publications are restricted to constant coefficients as well.

One interesting application of sparse grids is the solution of the electronic Schrödinger equation, since this is an elliptic PDE of a high dimensionality. In order to obtain high accuracy, a wavelet-based sparse grid method was proposed in [9]. The resulting linear equation system was solved in $O((N(\log N)^{d-1})^2)$ operations instead of $O((N(\log N)^{d-1}))$, since the large support of the wavelet functions leads to a complex structure of the stiffness matrix. It is evident that this high complexity of the computational amount limits the application of the method described in [9].

So far, the Ritz-Galerkin discretization of elliptic equations on sparse grids has had limited range of application since most partial differential equations in natural science or engineering include variable coefficients.

The first Ritz-Galerkin discretization on sparse grid with variable coefficients is presented in [16]. This discretization applies the semi-orthogonality property of standard hierarchical basis functions (see Section 2). A complete convergence theory is given in [15] for the two-dimensional case. The discretization leads to a symmetric stiffness matrix in the case of a symmetric bilinear form. Nevertheless, an extension to higher-dimensional problems is not possible for standard hierarchical basis functions since hierarchical basis functions do not satisfy a semi-orthogonality property for $d \geq 3$.

Other discretizations of PDEs with variable coefficients are presented in [1] and [8]. The discretization in [1] can be treated as a finite element discretization, while the discretization in [8] is a finite difference discretization. For symmetric problems, both discretizations lead to a non-symmetric linear equation system for symmetric problems, which is an undesired property.

Additionally, a convergence proof is missing for both discretizations. Therefore, convergence of these methods is not guaranteed in higher dimensions. Furthermore, the discretization in [1] requires high-order interpolation operators, which increases the computational load. However, simulation results presented in literature show an optimal convergence for certain two-dimensional and three-dimensional problems.

In this paper, we present a new method to discretize elliptic partial differential equations on sparse grids (see Section 2). This discretization uses prewavelets and their semi-orthogonality property (see [16]). It is well known that prewavelets and wavelets can be used to discretize partial differential equations (see [3, 13, 19], and [17]). In the context of sparse grids, they can even lead to natural discretizations of elliptic partial differential equations with variable coefficients. The elementary convergence theory of such discretizations is presented for a Helmholtz problem in [18]. In Section 6, we present an algorithm that efficiently evaluates the matrix vector multiplication with the discretization matrix. The algorithm applies only standard *1*-dimensional restriction and prolongation operators, a simple prewavelet stencil of size 5, and the classical stencil operator for multilinear finite elements. This operator-dependent stencil is a 9-point stencil for bilinear elements and a 27-point stencil for trilinear elements. However, in the six-dimensional case, the size of this stencil increases to $729 = 3^6$. The difficulty of this algorithm is to apply all operators in the correct sequential ordering.

In Section 8, simulation results are presented for the three-dimensional Poisson's problem on a curvilinear bounded domain and for a six-dimensional Helmholtz problem with a variable coefficient. The simulation result for Poisson's problem implies that sparse grids are not restricted to cubical domains. To our knowledge, the numerical result for the six-dimensional Helmholtz equation is the first simulation result for a six-dimensional Ritz-Galerkin finite element discretization of a elliptic PDE with variable coefficients.

This paper is restricted to non-adaptive grids. However, the algorithm presented in this paper can certainly be extended to adaptive sparse grids using concepts and ideas presented in [6, 16], and [12].

2 Sparse grid discretization

Let $d \geq 2$ be the dimension of space and $\Omega = [0, 1]^d$. Consider an elliptic differential equation:

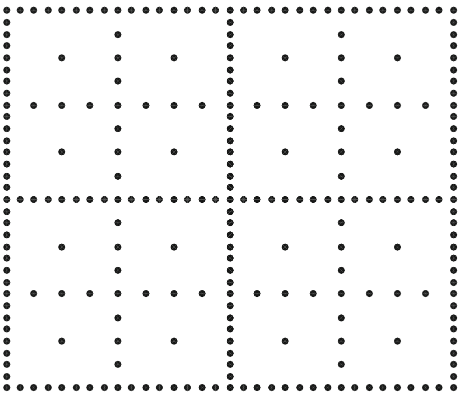


Fig. 1 Example of a two-dimensional sparse grid $\mathcal{D}_n$

Problem Let $f \in L^2(\Omega)$, $A \in (L^\infty(\Omega))^{d\times d}$ and $\kappa \in L^\infty(\Omega), \kappa \geq 0$ be given. Furthermore, assume that A is symmetric and uniformly positive definite. This means that there is a $\alpha > 0$ such that $v^T A(\mathbf{x})v > \alpha v^T v$ for almost every $\mathbf{x} \in \Omega$ and every vector $v \in \mathbb{R}^d$. Find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega (\nabla u)^T A(\mathbf{x})\nabla v + \kappa(\mathbf{x})uv \, d\mathbf{x} = \int_\Omega f v_h \, d\mathbf{x} \quad \forall v \in H_0^1(\Omega). \tag{1}$$

Our aim is to find an efficient sparse grid finite element discretization that can be used even for large dimension d. A typical two-dimensional sparse grid is depicted in Fig. 1 and a three-dimensional sparse grid in Fig. 11.

Finite elements on sparse grids are constructed by tensor products of one-dimensional finite elements. Here, we apply piecewise linear elements in 1D. Let us explain the construction of the sparse grid finite element space in more detail. To this end, define the one-dimensional grid

$$\begin{aligned} \Omega_k &= \{2^{-t-1} i \mid i \in I_t\}, \\ I_k &= \{i \mid i = 1, ..., 2^{t+1} - 1\} \text{ for } t \in \mathbb{N}_0, \quad I_{-1} = \emptyset. \end{aligned}$$

I_t is the index set of Ω_t. Observe that $\Omega_0 \subset \Omega_1 \subset \Omega_2 \subset ...$. The complementary index set is defined by

$$\Xi_t = I_t \backslash (2I_{t-1}) .$$

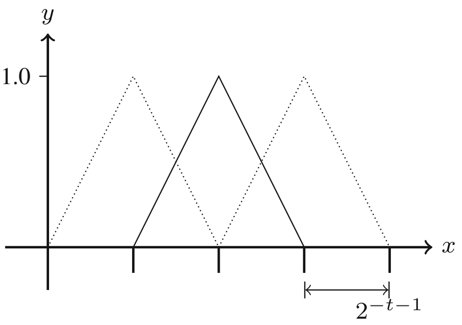


Fig. 2 One-dimensional nodal basis functions

Now, let V_t be the space of piecewise linear functions of mesh size 2^{-t-1} and $v_{t,i}^{\text{lin}}$ the corresponding nodal basis function at point $2^{-t-1}i \in \Omega_t$ (see Fig. 2).

Using these functions, we define prewavelets $\varphi_{t,i}$, $i \in \Xi_t$ by (see Fig. 3)

$$\varphi_{t,i} = \begin{cases} \frac{9}{10} v_{t,i}^{\text{lin}} - \frac{3}{5} v_{t,i+1}^{\text{lin}} + \frac{1}{10} v_{t,i+2}^{\text{lin}} & \text{if } 1 = i \\ v_{t,i}^{\text{lin}} - \frac{3}{5}(v_{t,i+1}^{\text{lin}} + v_{t,i-1}^{\text{lin}}) + \frac{1}{10}(v_{t,i+2}^{\text{lin}} + v_{t,i-2}^{\text{lin}}) & \text{if } 3 \le i \le 2^t - 3 \\ \frac{9}{10} v_{t,i}^{\text{lin}} - \frac{3}{5} v_{t,i-1}^{\text{lin}} + \frac{1}{10} v_{t,i-2}^{\text{lin}} & \text{if } i = 2^t - 1 \end{cases},$$

for $t \in \mathbb{N}$ and $\varphi_{0,1} = v_{0,1}^{\text{lin}}$. An important property of these functions is the L^2-orthogonality for prewavelets of different levels

$$\int_0^1 \varphi_{t,i}\, \varphi_{t',i'}\, dx = 0 \quad \text{if } t \neq t'. \tag{2}$$

Let us introduce the following abbreviations for a multi-index $\mathbf{t} = (t_1, ..., t_d) \in \mathbb{N}_0^d$:

$$\begin{aligned} |\mathbf{t}|_\delta &:= \sum_{i=1}^{\delta} |t_i|, \quad \text{for } 1 \le \delta \le d, \\ \mathbf{t} \le \mathbf{t}' &\text{ if } \; t_i \le t_i' \quad \forall i = 1, ..., d, \\ \max(\mathbf{t}, \mathbf{t}') &:= (\max(t_1, t_1'), ..., \max(t_d, t_d')), \quad \text{and} \\ \max(\mathbf{t}) &:= \max(t_1, ..., t_d). \end{aligned}$$

Using these abbreviations, the sets of tensor product indices are defined

$$\begin{aligned} \mathbf{I_t} &= \left\{(i_1, ..., i_d) \,\middle|\, i_s \in I_{t_s}, s = 1, ..., d\right\}, \\ \Xi_{\mathbf{t}} &= \left\{(\xi_1, ..., \xi_d) \,\middle|\, \xi_s \in \Xi_{t_s}, s = 1, ..., d\right\} \end{aligned}$$

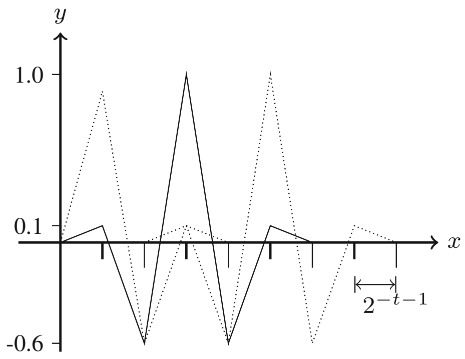


Fig. 3 One-dimensional prewavelet functions

and the tensor product functions

$$v^{\text{lin}}_{\mathbf{t},\mathbf{i}}(\mathbf{x}) := \prod_{s=1}^{d} v^{\text{lin}}_{t_s,i_s}(x_s), \quad \mathbf{i} \in \mathbf{I}_{\mathbf{t}},$$

$$\varphi_{\mathbf{t},\mathbf{i}}(\mathbf{x}) := \prod_{s=1}^{d} \varphi_{t_s,i_s}(x_s), \quad \mathbf{i} \in \Xi_{\mathbf{t}},$$

where $\mathbf{x} = (x_1, ..., x_d)$. These constructions allow to define the tensor product vector spaces (see Fig. 4)

$$\begin{aligned} V^{\text{full}}_{n,d} &:= \text{span}\left\{v^{\text{lin}}_{\mathbf{t},\mathbf{i}} \;\middle|\; \max(\mathbf{t}) \le n, \mathbf{i} \in \Xi_{\mathbf{t}}\right\}, \\ V_{\mathbf{t}} &:= \text{span}\left\{v^{\text{lin}}_{\mathbf{t}',\mathbf{i}} \;\middle|\; \mathbf{t}' \le \mathbf{t}, \mathbf{i} \in \Xi_{\mathbf{t}'}\right\}, \\ W_{\mathbf{t}} &:= \text{span}\left\{\varphi_{\mathbf{t},\mathbf{i}} \;\middle|\; \mathbf{i} \in \Xi_{\mathbf{t}}\right\}, \\ V^{\text{lin}}_{\mathcal{D}_n} &:= \text{span}\left\{v^{\text{lin}}_{\mathbf{t},\mathbf{i}} \;\middle|\; |\mathbf{t}|_d \le n, \mathbf{i} \in \Xi_{\mathbf{t}}\right\}, \\ V^{\text{prew}}_{\mathcal{D}_n} &:= \text{span}\left\{\varphi_{\mathbf{t},\mathbf{i}} \;\middle|\; |\mathbf{t}|_d \le n, \mathbf{i} \in \Xi_{\mathbf{t}}\right\}. \end{aligned}$$

Obviously, this results in $W_{\mathbf{t}} \subset V_{\mathbf{t}}$.

$V^{\text{full}}_{n,d}$ is the well-known standard space of multilinear finite element functions which can be written as follows:

$$V^{\text{full}}_{n,d} = \text{span}\left\{v^{\text{lin}}_{(n,\ldots,n),\mathbf{i}} \;\middle|\; \mathbf{i} \in \mathbf{I}_{(n,\ldots,n)}\right\}.$$

The sparse grid spaces $V^{\text{lin}}_{\mathcal{D}_n}$ and $V^{\text{prew}}_{\mathcal{D}_n}$ are equal (see [18]). However, the adaptive versions of these spaces are not equal. For reasons of simplicity, only the non-adaptive case is considered. In the case of smooth functions, sparse and full grid have similar approximation properties

$$\min_{v \in V^{\text{full}}_{n,d}} \|u - v\|_{H^1} \le C 2^{-n} \|u\|_{H^2} \quad \text{and}$$

$$\min_{v \in V^{\text{prew}}_{\mathcal{D}_n}} \|u - v\|_{H^1} \le C\, n\, 2^{-n} \left\| \frac{\partial^{2d} u}{\partial x_1^2 \ldots \partial x_d^2} \right\|_{L^2},$$

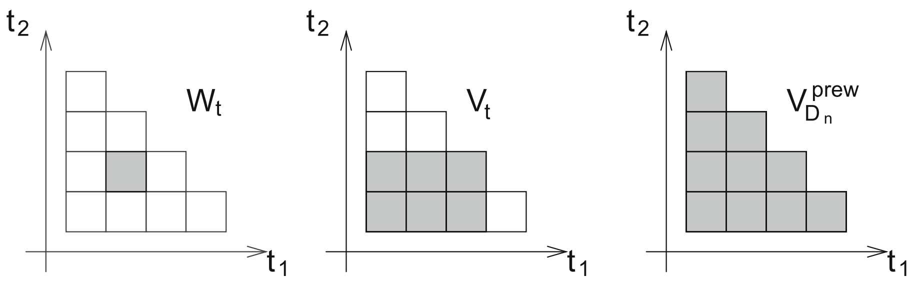


Fig. 4 Example of spaces $W_{\mathbf{t}} = W_{1,1}$, $V^{\text{full}}_{\mathbf{t}} = V^{\text{full}}_{2,1}$, and $V^{\text{prew}}_{\mathcal{D}_n} = V^{\text{prew}}_{\mathcal{D}_3}$

where C is a constant independent of n and u. However, the dimensions of the corresponding spaces are completely different:

$$\dim(V_{n,d}^{\text{full}}) = O(2^{nd}) \quad \text{and} \quad \dim(V_{\mathcal{D}_n}^{\text{prew}}) = O(n^{d-1}2^n).$$

Therefore, the aim of this paper is to find an efficient Galerkin discretization of Problem (2) using the sparse grid space $V_{\mathcal{D}_n}^{\text{prew}}$. To this end, the following lemma and corollary is an important observation:

Lemma 1 *Let κ be constant and $A = diag(\alpha_1, ..., \alpha_d)$ a constant diagonal matrix. Then, for all indices $\mathbf{t}$, $\mathbf{t}'$ such that $t_s \neq t_s'$ for at least two indexes $s = 1, ..., d$ the following equation holds:*

$$\int_\Omega (\nabla\varphi_{\mathbf{t},\mathbf{i}})^T A \nabla\varphi_{\mathbf{t}',\mathbf{i}'} + \kappa\varphi_{\mathbf{t},\mathbf{i}}\varphi_{\mathbf{t}',\mathbf{i}'} \, d\mathbf{x} = 0. \tag{3}$$

Proof By tensor product construction of the functions $\varphi_{\mathbf{t},\mathbf{i}}$ and $\varphi_{\mathbf{t}',\mathbf{i}'}$, the integral of (3) can be written as a product of one-dimensional integrals for each term in (3), since the coefficients A and κ are constant. Here, the orthogonality property (2) implies that the term containing κ is zero. Next, let us consider the term

$$\alpha_k \int_\Omega \frac{\partial\varphi_{\mathbf{t},\mathbf{i}}}{\partial x_k}\frac{\partial\varphi_{\mathbf{t}',\mathbf{i}'}}{\partial x_k}\, d\mathbf{x} \;=\; \alpha_k \int_0^1 \frac{\partial\varphi_{t_k,i_k}}{\partial x_k}\frac{\partial\varphi_{t_k',i_k'}}{\partial x_k}\, dx_k \;\prod_{s\neq k}\int_0^1 \varphi_{t_s,i_s}\varphi_{t_s',i_s'}\, dx_s.$$

Since $t_s \neq t_s'$ for at least two indexes $s = 1, ..., d$, there is an index $s \neq k$ such that $t_s \neq t_s'$. Thus, (2) completes the proof. □

Corollary 1 (**Semi-orthogonality property**) *Let κ be constant and $A = diag(\alpha_1, ..., \alpha_d)$ a constant diagonal matrix. Then, for all indices $\mathbf{t}$, $\mathbf{t}'$, $\mathbf{i} \in \Xi_{\mathbf{t}}$, and $\mathbf{i}' \in \Xi_{\mathbf{t}'}$ such that*

$$|\max(\mathbf{t},\mathbf{t}')|_d > n \text{ and } |\mathbf{t}|_d \leq n, \quad |\mathbf{t}'|_d \leq n, \tag{4}$$

the following equation holds:

$$\int_\Omega (\nabla\varphi_{\mathbf{t},\mathbf{i}})^T A \nabla\varphi_{\mathbf{t}',\mathbf{i}'} + \kappa\varphi_{\mathbf{t},\mathbf{i}}\varphi_{\mathbf{t}',\mathbf{i}'} \, d\mathbf{x} = 0.$$

The consequence of this corollary is that prewavelet basis functions with overlapping support are orthogonal to each other (see Fig. 5). This orthogonality property is the motivation behind the following discretization:

Discretization 1 (**Semi-orthogonality**) *Let $f \in L^2(\Omega)$ and $\kappa \in L^\infty(\Omega)$, $\kappa \geq 0$ be given. Then, let us define*

$$a(u, v) := \int_\Omega (\nabla u)^T A(\mathbf{x}) \nabla v + \kappa(\mathbf{x}) uv \, d\mathbf{x}$$

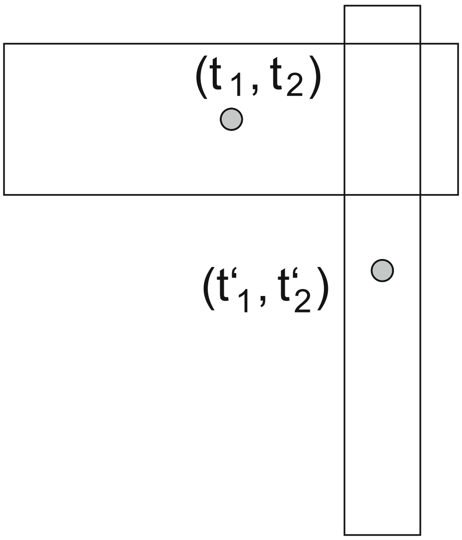


Fig. 5 Example of the support of two basis functions that satisfy $|\max(\mathbf{t}, \mathbf{t}')|_d > n$ and $|\mathbf{t}|_d \leq n$, $|\mathbf{t}'|_d \leq n$

and

$$a_n^{\textit{semi-ortho}} : V_{\mathcal{D}_n}^{\textit{prew}} \times V_{\mathcal{D}_n}^{\textit{prew}} \to \mathbb{R}$$

$$a_n^{\textit{semi-ortho}}(\varphi_{\mathbf{t},\mathbf{i}}, \varphi_{\mathbf{t}',\mathbf{i}'}) := \begin{cases} a(\varphi_{\mathbf{t},\mathbf{i}}, \varphi_{\mathbf{t}',\mathbf{i}'}) & \textit{if } |\max(\mathbf{t}, \mathbf{t}')|_d \leq n \\ 0 & \textit{if } |\max(\mathbf{t}, \mathbf{t}')|_d > n \end{cases}.$$

Find $u_{\mathcal{D}_n}^{\textit{prew}} \in V_{\mathcal{D}_n}^{\textit{prew}}$ *such that*

$$a_n^{\textit{semi-ortho}}(u_{\mathcal{D}_n}^{\textit{prew}}, v_h) = \int_\Omega f v_h \, d\mathbf{x} \quad \forall v_h \in V_{\mathcal{D}_n}^{\textit{prew}}. \tag{5}$$

In [18], we analyzed the convergence of this discretization for the Helmholtz problem with variable coefficients with respect to the H^1-norm. This paper shows how to obtain an efficient algorithm for solving the corresponding linear equation.

3 Basic notation

The difficulty in explaining sparse grid algorithms is that the matrices in these algorithms are applied to vectors with varying size. Thus, describing these matrices in a mathematically correct form leads to a non-trivial notation. A second problem appears in the case of adaptive grids. All sparse grid algorithms have a recursive structure that use a tree data structure. Explaining such algorithms in a mathematical and clear notation is difficult. Therefore, we restrict ourselves to non-adaptive sparse grids and assume that a sparse grid is a union of semi-coarsened full grids.

Furthermore, we introduce a notation that is based on operators on vector spaces and its dual space. Assume that the finite element solution $u_\mathbf{t} \in V_\mathbf{t}$ is searched for, such that

$$a_n^{\text{semi-ortho}}(u_{\mathbf{t}}, v_{\mathbf{t}}) = \int_\Omega f v_{\mathbf{t}} \, d\mathbf{x} \quad \forall v_{\mathbf{t}} \in V_{\mathbf{t}}.$$

Then, $u_{\mathbf{t}}$ is contained in the vector space $V_{\mathbf{t}}$, but the mappings

$$v \mapsto \int_\Omega f v \, d\mathbf{x}, \quad v \in V_{\mathbf{t}}$$

and

$$v \mapsto a_n^{\text{semi-ortho}}(w, v), \quad v \in V_{\mathbf{t}}$$

are contained in the dual space $V_{\mathbf{t}}'$. To store an element in $V_{\mathbf{t}}$ or a functional in $V_{\mathbf{t}}'$ as a vector, assume that $n_{\mathbf{t}}$ is the number of grid points on level $\mathbf{t}$

$$n_{\mathbf{t}} = \dim(V_{\mathbf{t}})$$

Moreover, assume that the data of a sparse grid algorithm is stored on various suitable full grids. A corresponding global array is as follows:

$$\begin{aligned} \mathbf{U}_n &:= (U_{\mathbf{t}})_{|\mathbf{t}|_d \le n} \\ U_{\mathbf{t}} &\in \mathbb{R}^{n_{\mathbf{t}}} \end{aligned}$$

The vector $U_{\mathbf{t}}$ is used to store data of different mathematical objects. One possibility is to describe a function in $V_{\mathbf{t}}$ by the vector $U_{\mathbf{t}}$. Another possibility is to describe a functional of $V_{\mathbf{t}}'$ by $U_{\mathbf{t}}$. The notation for a corresponding assignment operator is as follows:

$$\left(U_{\mathbf{t}} \overset{\text{set}}{\longleftarrow} u\right) \quad \text{for } u \in V_{\mathbf{t}}$$

and

$$\left(U_{\mathbf{t}} \overset{\text{set}}{\longleftarrow} f\right) \quad \text{for } f \in V_{\mathbf{t}}'$$

However, the assignment operator $\overset{\text{set}}{\longleftarrow}$ depends on the basis, which is used to represent u or f, respectively. Let us explain this with the following examples:

Example 1 Let $u = \sum_{\mathbf{i} \in \mathbf{I}_{\mathbf{t}}} c_{\mathbf{t},\mathbf{i}} v_{\mathbf{t},\mathbf{i}}^{\text{lin}} \in V_{\mathbf{t}}$.
Then,

$$\left(U_{\mathbf{t}} \overset{\text{set}}{\longleftarrow} u\right) \quad :\Leftrightarrow \quad U_{\mathbf{t}} = (c_{\mathbf{t},\mathbf{i}})_{\mathbf{i} \in \mathbf{I}_{\mathbf{t}}}.$$

Example 2 Let $u = \sum_{\mathbf{i} \in \Xi_{\mathbf{t}}} c_{\mathbf{t},\mathbf{i}} \varphi_{\mathbf{t},\mathbf{i}} \in W_{\mathbf{t}} \subset V_{\mathbf{t}}$.
Then,

$$\left(U_{\mathbf{t}} \overset{\text{set}}{\longleftarrow} u\right) \quad :\Leftrightarrow \quad U_{\mathbf{t}} = \begin{pmatrix} c_{\mathbf{t},\mathbf{i}} & \text{if } \mathbf{i} \in \Xi_{\mathbf{t}} \\ 0 & \text{else.} \end{pmatrix}.$$

Example 3 Let

$$v_{\mathbf{t},\mathbf{i}} := \left(\bigotimes_{s \in S} \varphi_{t_s, i_s}\right) \bigotimes \left(\bigotimes_{s \notin S} v_{t_s, i_s}^{\text{lin}}\right)$$

be the basis function which is a prewavelet function in directions $s \in S \subset \{1, ..., d\}$ and a nodal basis function for all other directions. Now, let $u = \sum_{\mathbf{i}} c_{\mathbf{t},\mathbf{i}} v_{\mathbf{t},\mathbf{i}}$. Then, $U_{\mathbf{t}}$ stores the coefficients $c_{\mathbf{t},\mathbf{i}}$ after evaluation of

$$\left(U_{\mathbf{t}} \quad \overset{\text{set}}{\longleftarrow} \quad u \right).$$

Example 4 Let $f \in V_{\mathbf{t}}'$. Then, we write $\left(U_{\mathbf{t}} \quad \overset{\text{set}}{\longleftarrow} \quad f \right) \quad :\Leftrightarrow \quad U_{\mathbf{t}} = (f(v_{\mathbf{t},\mathbf{i}}^{\text{lin}}))_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}}}$.

Example 5 Let $f \in W_{\mathbf{t}}'$. Then, we write $\left(U_{\mathbf{t}} \quad \overset{\text{set}}{\longleftarrow} \quad f \right) \quad :\Leftrightarrow \quad U_{\mathbf{t}} = \begin{pmatrix} f(\varphi_{\mathbf{t},\mathbf{i}}) & \text{if } \mathbf{i} \in \Xi_{\mathbf{t}} \\ 0 & \text{else.} \end{pmatrix}$.

For describing our algorithms, we introduce a special operator $\mathcal{B}$, which we will call *back construction operator*. This operator reconstructs the mathematical object u which was used to set values in a vector $U_{\mathbf{t}}$. This implies the following property of the back construction operator $\mathcal{B}$:

$$\left(U_{\mathbf{t}} \quad \overset{\text{set}}{\longleftarrow} \quad u \right) \quad \Rightarrow \quad u = \mathcal{B}(U_{\mathbf{t}}).$$

Therefore, if $U_{\mathbf{t}}$ was set by u in an assignment $\left(U_{\mathbf{t}} \quad \overset{\text{set}}{\longleftarrow} \quad u \right)$ during execution of an algorithm, then $\mathcal{B}$ reconstructs u in a later execution of this algorithm. This notation avoids describing which basis was exactly used to define $U_{\mathbf{t}}$.

4 Basic operators

The algorithms in this paper are mainly based on well-known one-dimensional operators. Briefly recall these operators:

1. Prolongation

A prolongation in direction s can be described by

$$W_{\mathbf{t}} \quad \overset{\text{set}}{\longleftarrow} \quad I_{\mathbf{t}}^{s} \left(\mathcal{B} \left(W_{\mathbf{t}-e_s} \right) \right),$$

where $\mathcal{B}(W_{\mathbf{t}-e_s}) \in V_{\mathbf{t}-e_s}$ is given and e_s is the unit vector in direction s. We want to describe the operator $I_{\mathbf{t}}^{s}$ in matrix format. To this end, let

$$(c_{\mathbf{i}}^{\text{co}})_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}-e_s}} \in \mathbb{R}^{|\mathbf{I}_{\mathbf{t}-e_s}|}, \qquad \mathcal{B}(W_{\mathbf{t}-e_s}) = \sum_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}-e_s}} c_{\mathbf{i}}^{\text{co}} v_{\mathbf{t}-e_s,\mathbf{i}},$$

$$(c_{\mathbf{i}})_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}}} \in \mathbb{R}^{|\mathbf{I}_{\mathbf{t}}|}, \qquad \mathcal{B}(W_{\mathbf{t}}) = \sum_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}}} c_{\mathbf{i}} v_{\mathbf{t},\mathbf{i}},$$

where the basis function $v_{\mathbf{t},\mathbf{i}}$ is a nodal basis function in direction s (see Example 3). The prolongation in direction s acts on the above vectors as follows:

$$(c_{\mathbf{i}})_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}}} = M_s^{\text{prol}} (c_{\mathbf{i}}^{\text{co}})_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}-e_s}}, \qquad M_s^{\text{prol}} = \otimes_{i=1}^{s-1} \text{Id} \otimes M^{\text{prol}} \otimes \otimes_{i=s+1}^{d} \text{Id}, \quad (6)$$

where Id is the identity matrix and M^{prol} is the matrix

$$M^{\text{prol}} = \begin{pmatrix} 1 & & & & & \\ \frac{1}{2} & \frac{1}{2} & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & \frac{1}{2} & \frac{1}{2} & \\ & & & & 1 & \end{pmatrix}.$$

Obviously, the operator $I_{\mathbf{t}}^{s}$ can be generalized to an operator $I_{\mathbf{t}}^{\boldsymbol{\alpha}}$, which interpolates in directions $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} \in \{0, 1\}^d$.

2. Coarse grid interpolation (or coarse injection)

Let $\mathcal{B}(W_{\mathbf{t}}) \in V_{\mathbf{t}}$. Then, the operator $I_{\mathbf{t}-e_s}^{s}$ used in

$$W_{\mathbf{t}-e_s} \overset{\text{set}}{\longleftarrow} I_{\mathbf{t}-e_s}^{s}\left(\mathcal{B}(W_{\mathbf{t}})\right),$$

is defined to be the operator, which interpolates $\mathcal{B}(W_{\mathbf{t}})$ on a coarse grid in direction s. $I_{\mathbf{t}-e_s}^{s}\left(\mathcal{B}(W_{\mathbf{t}})\right)$ applies nodal basis functions in direction s.

3. Restriction of the right-hand side

A restriction of the right-hand side in direction s can be described by

$$G_{\mathbf{t}-e_s} \overset{\text{set}}{\longleftarrow} \big(v \mapsto \mathcal{B}(G_{\mathbf{t}})(v),\ v \in V_{\mathbf{t}-e_s}\big),$$

where $\mathcal{B}(G_{\mathbf{t}}) \in V_{\mathbf{t}}'$ is given. To describe the matrix form of this operator, let

$$\begin{aligned} (g_{\mathbf{i}})_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}}} &\in \mathbb{R}^{|\mathbf{I}_{\mathbf{t}}|}, \qquad & g_{\mathbf{i}} &= \mathcal{B}(G_{\mathbf{t}})(v_{\mathbf{t},\mathbf{i}}^{\text{lin}}), \mathbf{i} \in \mathbf{I}_{\mathbf{t}}, \\ (g_{\mathbf{i}}^{\text{co}})_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}-e_s}} &\in \mathbb{R}^{|\mathbf{I}_{\mathbf{t}-e_s}|}, \qquad & g_{\mathbf{i}} &= \mathcal{B}(G_{\mathbf{t}-e_s})(v_{\mathbf{t}-e_s,\mathbf{i}}^{\text{lin}}), \mathbf{i} \in \mathbf{I}_{\mathbf{t}-e_s}. \end{aligned}$$

The restriction in direction s acts on these vectors as follows:

$$(g_{\mathbf{i}}^{\text{co}})_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}-e_s}} = M_s^{\text{res}}(g_{\mathbf{i}})_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}}}, \qquad M_s^{\text{res}} = \otimes_{i=1}^{s-1}\text{Id} \otimes M^{\text{res}} \otimes_{i=s+1}^{d} \text{Id}, \tag{7}$$

where Id is the identity matrix and M^{res} is the matrix

$$M^{\text{res}} = \begin{pmatrix} \frac{1}{2} & 1 & \frac{1}{2} & & & & & & & \\ & & & \ddots & & & & & & \\ & & & & \frac{1}{2} & 1 & \frac{1}{2} & & & \\ & & & & & & \frac{1}{2} & 1 & \frac{1}{2} & \\ & & & & & & & & \ddots & \\ & & & & & & & & \frac{1}{2} & 1 & \frac{1}{2} \end{pmatrix}.$$

The next operators require the application of matrices including prewavelet coefficients. These are as follows:

$$M = \begin{pmatrix} \frac{9}{10} & \frac{1}{2} & \frac{1}{10} & & & \\ -\frac{3}{5} & 1 & -\frac{3}{5} & & & \\ \frac{1}{10} & \frac{1}{2} & 1 & \ddots & & \\ & & -\frac{3}{5} & \ddots & \frac{1}{2} & \frac{1}{10} \\ & & \frac{1}{10} & \ddots & 1 & -\frac{3}{5} \\ & & & & \frac{1}{2} & \frac{9}{10} \end{pmatrix} \quad \text{and} \quad M^{\text{nodal}} = \begin{pmatrix} \frac{9}{10} & \frac{1}{10} & & \\ -\frac{3}{5} & -\frac{3}{5} & & \\ \frac{1}{10} & 1 & \ddots & \\ & -\frac{3}{5} & \ddots & \frac{1}{10} \\ & \frac{1}{10} & \ddots & -\frac{3}{5} \\ & & & \frac{9}{10} \end{pmatrix} \tag{8}$$

M contains prewavelet coefficients and coarse to fine interpolation coefficients whereas M^{nodal} contains only prewavelet coefficients. Observe that M is a square matrix and M^{nodal} a rectangular matrix.

4. Transformation to a prewavelet basis

Let s be a direction such that $1 \leq s \leq d$. Furthermore, assume that $U_{\mathbf{t}}$ is given such that $\mathcal{B}(U_{\mathbf{t}}) \in V_{\mathbf{t}}$ and

- $U_{\mathbf{t}}$ stores the coefficients of $\mathcal{B}(U_{\mathbf{t}})$ in nodal basis form in the direction s and
- $U_{\mathbf{t}}$ stores the coefficients of $\mathcal{B}(U_{\mathbf{t}})$ in prewavelet form or nodal basis form in the directions $\tilde{d} \neq s$.

Now assume that the prewavelet coefficients are calculated in direction s with respect to level t_s of $\mathcal{B}(U_{\mathbf{t}})$ and the resulting vector is stored in $H_{\mathbf{t}}$. The corresponding assignment can be written as follows:

$$H_{\mathbf{t}} \quad \overset{\text{set}}{\longleftarrow} \quad Q^s_{\mathbf{t}}(\mathcal{B}(U_{\mathbf{t}})). \tag{9}$$

Here, $Q^s_{\mathbf{t}}$ is the L^2 projection operator onto the space

$$\otimes_{i=1}^{s-1} V_{t_i} \otimes W_{t_s} \otimes \otimes_{i=s+1}^{d} V_{t_i}.$$

In matrix form, assignment (9) can be written as follows:

$$\otimes_{i=1}^{s-1}\text{Id} \otimes M^{\text{prew}} \otimes \otimes_{i=s+1}^{d}\text{Id},$$

where Id are suitable identity matrices and M^{prew} is the matrix of the one-dimensional case. To describe the matrix M^{prew} of the one-dimensional case, let

$$(u_i)_{i \in I_t} \in \mathbb{R}^{|I_t|}, \qquad u_i = \mathcal{B}(U_t) = \sum_{i \in I_t} v_{t,i},$$

$$(h_i)_{i \in \Xi_t} \in \mathbb{R}^{|\Xi_t|}, \qquad h_i = \mathcal{B}(H_t) = \sum_{i \in \Xi_t} \varphi_{t,i},$$

where $t = t_s$. The matrix M^{prew} performs the following mapping:

$$(h_i)_{i\in\Xi_t} = M^{\text{prew}}(u_i)_{i\in I_t}.$$

Let $\mathcal{R}^{\text{prew}}$ be the following restriction operator which takes only the prewavelet coefficients:

$$\mathcal{R}^{\text{prew}}\left((h_i)_{i\in I_t}\right) := (h_i)_{i\in\Xi_t}.$$

Then, $M^{\text{prew}} = \mathcal{R}^{\text{prew}}M^{-1}$.

This means that the assignment (9) has to be implemented by inverting the matrix M in direction s and taking only the resulting prewavelet coefficients (see (8)).

5. Transformation to a nodal basis

Let $\mathcal{B}(H_{\mathbf{t}}) \in V_{\mathbf{t}}$ and assume that $H_{\mathbf{t}}$ stores the coefficients of $\mathcal{B}(H_{\mathbf{t}})$ in the prewavelet format in the directions $S \subset \{1, ..., d\}$ and in the nodal basis format for all other directions. Then, writing $I_{\mathbf{t}}(\mathcal{B}(H_{\mathbf{t}}))$ means that we decompose $\mathcal{B}(H_{\mathbf{t}})$ only by nodal basis functions:

$$\mathcal{B}(H_{\mathbf{t}}) = \sum_{\mathbf{i}\in\mathbf{I_t}} c_{\mathbf{t},\mathbf{i}} v_{\mathbf{t},\mathbf{i}}^{\text{lin}} \in V_{\mathbf{t}}.$$

This implies that after evaluating $Q_{\mathbf{t}} \overset{\text{set}}{\longleftarrow} T_{\mathbf{t}}(\mathcal{B}(H_{\mathbf{t}}))$, $Q_{\mathbf{t}}$ stores the coefficients $c_{\mathbf{t},\mathbf{i}}$. Obviously, the matrix of this complete transformation to a nodal basis includes a tensor product of prewavelet coefficients in directions S (see matrix M^{nodal} in (8)). Analogously, we define the operator $T_{\mathbf{t}}^{\delta}$, which performs a transformation to a nodal basis only in the direction δ.

5. Transformation of functionals to a prewavelet basis

Let $\mathcal{B}(Z_{\mathbf{t}}) \in V_{\mathbf{t}}'$ and assume that $Z_{\mathbf{t}}$ represents $\mathcal{B}(Z_{\mathbf{t}})$ with respect to nodal functions (see Example 4). Then, $F_{\mathbf{t}}$ represents $\mathcal{B}(Z_{\mathbf{t}})$ with respect to prewavelet functions (see Example 5) after evaluation of

$$F_{\mathbf{t}} \overset{\text{set}}{\longleftarrow} (\varphi \mapsto \mathcal{B}(Z_{\mathbf{t}})(\varphi), \varphi \in W_{\mathbf{t}}).$$

Obviously, the matrix of this complete transformation of functionals to a prewavelet basis is a tensor product of prewavelet coefficients in directions $s = 1, ..., d$ (see matrix M^{nodal} in (8)).

6. Discretization stencil

Let $U_{\mathbf{t}}$ be a vector on a semi-coarsened full subgrid such that $\mathcal{B}(U_{\mathbf{t}}) \in V_{\mathbf{t}}$. Then, an assignment involving the bilinear form of the operator is as follows:

$$Z_{\mathbf{t}} \overset{\text{set}}{\longleftarrow} \left(v \mapsto a(\mathcal{B}(U_{\mathbf{t}}), v),\ v \in V_{\mathbf{t}}\right). \tag{10}$$

The computation corresponding to this assignment requires a 9-point stencil in the two-dimensional case and a 27-point stencil in the three-dimensional case.

As an example, let us consider the two-dimensional case and the bilinear form corresponding to Poisson's equation:

$$a(u, v) = \int_\Omega \nabla u \nabla v \, d(x, y).$$

Then, the discretization stencil for bilinear finite elements is as follows:

$$S_{h_x,hy} = \frac{h_y}{h_x}\frac{1}{6}\begin{bmatrix} -1 & 2 & -1 \\ -4 & 8 & -4 \\ -1 & 2 & -1 \end{bmatrix} + \frac{h_x}{h_y}\frac{1}{6}\begin{bmatrix} -1 & -4 & -1 \\ 2 & 8 & 2 \\ -1 & -4 & -1 \end{bmatrix},$$

where $h_x = 2^{t_1}$, $h_y = 2^{t_2}$. Then, $Z_{\mathbf{t}}$ in (10) is $U_{\mathbf{t}}$ applied to the stencil $S_{h_x,hy}$.

Remark In this paper, we assume that the 3^d-stencils corresponding to the bilinear form a are given for each depth $\mathbf{t}$. This means that these stencils are the only description of a which is used in the subsequent algorithms of this paper. In particular, Algorithm 3 does not need any tensor product construction of a. However, an accurate computation of the 3^d-stencils is a non-trivial task, since the meshsizes $h_i = 2^{t_i}$ can be large. Therefore, the simulation results in Section 8 are restricted to cases, where the stencils can be obtained by analytic computations. If an analytic integration of the local stiffness matrices is not possible, then one has to apply a piecewise constant interpolation of the variable coefficients on the sparse grid in order to compute the the 3^d-stencils approximatively as in [5]. We will explain the corresponding algorithm in a subsequent paper.

5 Calculation of prewavelet coefficients

Let $(F_{\mathbf{t}})_{|\mathbf{t}|_d \le n}$ be a function evaluated on the sparse grid of depth n such that $F_{\mathbf{t}} = \left(f\left(x_{\mathbf{t},\mathbf{i}}\right)\right)_{\mathbf{i}\in\mathbf{I_t}}$ where $x_{\mathbf{t},\mathbf{i}}$ is the grid point on level $\mathbf{t}$ with index $\mathbf{i}$. Algorithm 1 calculates the prewavelet decomposition with coefficients $(C_{\mathbf{t}})_{|\mathbf{t}|_d \le n} = \left(c_{\mathbf{t},\mathbf{i}}\right)_{|\mathbf{t}|_d \le n, \mathbf{i}\in\mathbf{I_t}}$ given by

$$f\left(x_{\mathbf{t},\mathbf{i}}\right) = \sum_{|\mathbf{t}'|_d \le n, \mathbf{i}\in\mathbf{I_t}} c_{\mathbf{t}',\mathbf{i}} \varphi_{\mathbf{t}',\mathbf{i}}\left(x_{\mathbf{t},\mathbf{i}}\right) \tag{11}$$

for every sparse grid point $x_{\mathbf{t},\mathbf{i}} \in \mathcal{D}_n$, where $\mathcal{D}_n$ is the sparse grid of depth n.

The calculation of the wavelet coefficients works recursively through all the dimensions. For this purpose, the coefficients of the highest dimension are calculated first, starting from the grid with maximum depth down to the coarsest grid. After this, the algorithm continues with lower dimensions.

The calculation of the prewavelet coefficients on depth t in one dimension requires one to solve a system of linear equations with $2^{t+1} + 1$ unknowns. The inverse matrix M^{-1} can efficiently be computed using an LU decomposition since M is a five-diagonal matrix (see (8)).

Now, let us explain the basic idea of the algorithm for dimension $0 \leq \tilde{d} < d$ on level $t_{\tilde{d}}$. First, it calculates the prewavelet decomposition. Then, the local hierarchical surplus is subtracted from all low-order levels $\tilde{t}_{\tilde{d}} < t_{\tilde{d}}$. In the case of sparse grids, this requires the interpolation of the the hierarchical surplus for grids which have no direct predecessor. To this end, the combination technique is applied to all direct neighbors in directions with dimensions larger than $\tilde{d}$.

The same algorithm is used in reverse order to calculate a point-wise evaluation $F_{\mathbf{t}} = \left(f\left(x_{\mathbf{t},\mathbf{i}}\right)\right)_{\mathbf{i}\in\mathbf{I}_{\mathbf{t}}}$ for a given set of prewavelet coefficients $(C_{\mathbf{t}})_{|\mathbf{t}|_d\leq n}$. Algorithm 2 starts on the coarsest level of the smallest dimension and accumulates the surpluses over all dimensions.

Let us explain the structure of Algorithm 2 and the subsequent algorithms Algorithm 1 and Algorithm 3. First observe, that we used the following notation for the upper part of $\mathbf{t}$:

$$\mathcal{U}_\delta(\mathbf{t}) = (t_{\delta+1}, t_{\delta+2}, ..., t_d).$$

Then, Algorithms 1, 2, and 3 perform a recursive call for $\tau = m, ..., 0$ and $\tau = 0, ..., m$, respectively. In the recursive functions the dimension δ decreases from d to 0. Furthermore, the upper part of the depth vector $\mathbf{t}$ is fixed from $\delta + 1, \cdots, d$. A recursive call increases this fixed part of $\mathbf{t}$:

$$\begin{array}{rl}\text{input of recursion:} & (\overbrace{t_1, t_2, \cdots, t_\delta}^{\substack{\text{variable part}\\ \sum_{i=1}^{\delta} t_i \leq m}}, \overbrace{t_{\delta+1}, \cdots, t_d}^{\substack{\text{fixed part}\\ \mathcal{U}_\delta(\mathbf{t})=}}) \\ & \Downarrow \\ \text{recursion call for } \tau: & (\underbrace{t_1, t_2, \cdots, t_{\delta-1}}_{\sum_{i=1}^{\delta-1} t_i \leq m-\tau}, \underbrace{\tau, t_{\delta+1}, \cdots, t_d}_{\text{new fixed part}}).\end{array}$$

6 Matrix multiplication

Let

$$\mathcal{A}_n = \left(a_n^{\text{semi-ortho}}\left(\varphi_{\mathbf{t},\mathbf{i}}, \varphi_{\mathbf{t}',\mathbf{i}'}\right)\right)_{\substack{|\mathbf{t}|_d\leq n, \mathbf{i}\in\Xi_{\mathbf{t}}\\ |\mathbf{t}'|_d\leq n, \mathbf{i}'\in\Xi_{\mathbf{t}'}}}. \tag{12}$$

be the stiffness matrix of Discretization 1. The main difficulty is to construct an algorithm that efficiently evaluates a matrix vector multiplication with matrix $\mathcal{A}_n$. To construct such an algorithm, the notation introduced in Section 3 is applied. The recursive Algorithm 3 is obtained.

To explain the concept of this algorithm, we start with a one-dimensional observation. Assume that the following prewavelet decomposition is given as follows:

$$u = \sum_{t'=1}^{n} w_{t'} \quad \text{where} \quad w_{t'} = \sum_{i\in\Xi_{t'}} c_{t',i}\varphi_{t',i}.$$

Algorithm 1 Calculate prewavelet decomposition

```
Input: Let $\mathbf{U}_n = (U_\mathbf{t})_{|\mathbf{t}|_d \le n}$ be given in nodal format.
 Call PREWAVELET ALGORITHM ( $\mathbf{U}_n, d$).
Output: $\mathbf{U}_n$ in prewavelet format until dimension $d$.

Function PREWAVELET ALGORITHM ( $\mathbf{U}_m := (U_\mathbf{t})_{|\mathbf{t}|_\delta \le m, \mathcal{U}_\delta(\mathbf{t})=\mathbf{t}_u}, \delta$) {
 iterate for $\tau = m, ..., 0$ {
  1. Calculate prewavelet coefficients in direction $\delta$:
  for every $\mathbf{t}$ with $t_\delta = \tau$ and $\mathcal{U}_\delta(\mathbf{t}) = \mathbf{t}_u$ do:
   $U_\mathbf{t} \overset{\text{set}}{\longleftarrow} Q_\mathbf{t}^\delta(\mathcal{B}(U_\mathbf{t}))$   // transform to prewavelets in direction $\delta$
   $W_\mathbf{t} = U_\mathbf{t}$
  2. Subtract interpolated prewavelets on coarse grid
  iterate for $t' = \tau, ..., 1$ {
   2.1 for every $\mathbf{t}$ with $t_\delta = t'$ and $\mathcal{U}_\delta(\mathbf{t}) = \mathbf{t}_u$ do:
    $W_{\mathbf{t}-e_\delta} \overset{\text{set}}{\longleftarrow} I_{\mathbf{t}-e_\delta}^\delta(\mathcal{B}(W_\mathbf{t}))$   // coarse interpolation in direction $\delta$
    $U_{\mathbf{t}-e_\delta} \overset{\text{set}}{\longleftarrow} \mathcal{B}(U_{\mathbf{t}-e_\delta}) - (\mathcal{B}(W_{\mathbf{t}-e_\delta}))$
   2.2 if $\delta > 1$ then {
    for every $\mathbf{t}$ with $|\mathbf{t}|_\delta = m$ and $t_\delta = t' - 1$ and $\mathcal{U}_\delta(\mathbf{t}) = \mathbf{t}_u$ do:
     $W_\mathbf{t} \overset{\text{set}}{\longleftarrow} \sum_{\substack{(\alpha_1,...,\alpha_{\delta-1}) \in \{0,1\}^{\delta-1} \wedge \\ \boldsymbol{\alpha}=(\alpha_1,...,\alpha_{\delta-1},0...,0) \wedge \boldsymbol{\alpha} \neq (0,...,0)}} (-1)^{|\boldsymbol{\alpha}|_d+1} I_\mathbf{t}(\mathcal{B}(W_{\mathbf{t}-\boldsymbol{\alpha}}))$
     $U_\mathbf{t} \overset{\text{set}}{\longleftarrow} \mathcal{B}(U_\mathbf{t}) - \mathcal{B}(W_\mathbf{t})$  // subtract local hierarchical surplus
   }
  }
  3. Recursion
  if $\delta > 1$ then {
   Define $m^{\text{low}} := m - \tau$. Define $\mathbf{U}_{m^{\text{low}}}^{\text{low}} := (U_\mathbf{t})_{|\mathbf{t}|_{\delta-1} \le m^{\text{low}}, t_\delta = \tau, \mathcal{U}_\delta(\mathbf{t})=\mathbf{t}_u}$
   If $m^{\text{low}} > 0$ call PREWAVELET ALGORITHM ( $\mathbf{U}_{m^{\text{low}}}^{\text{low}}, \delta - 1$)
  }
 }
}
End of Algorithm.
```

Then, we can split the matrix vector multiplication in two parts:

$$a(u, v_t) = \sum_{t'>t} a(w_{t'}, v_t) \quad + \quad a(\sum_{t' \le t} w_{t'}, v_t).$$

Algorithm 3 calculates these two parts separately. We call these two parts *restriction part* and *prolongation part*.

The restriction part $\sum_{t'>t} a(w_{t'}, v_t)$ is calculated in the function SPARSE GRID MATRIX MULTIPLICATION with input parameter $D = 1$. The essential calculations are performed as follows in Step 3.2 and Step 1. The recursive call of the algorithm applies the discretization stencil on $H_\mathbf{t}$ (see Step 1) first and then performs several restrictions (see Step 3.2).

Algorithm 2 Calculate back prewavelet decomposition

```
Input: Let $\mathbf{U}_n = (U_{\mathbf{t}})_{|\mathbf{t}|_d \le n}$ be given in prewavelet format.
 Call BACK PREWAVELET ALGORITHM ( $\mathbf{U}_n, d$).
Output: $\mathbf{U}_n$ in nodal format until dimension $d$.

Function BACK PREWAVELET ALGORITHM ( $\mathbf{U}_m := (U_{\mathbf{t}})_{|\mathbf{t}|_\delta \le m, \mathcal{U}_\delta(\mathbf{t})=\mathbf{t}_u}, \delta$) {
 iterate for $\tau = 0, ..., m$ {
  1. Recursion
  if $\delta > 1$ then {
   Define $m^{\text{low}} := m - \tau$. Define $\mathbf{U}^{\text{low}}_{m^{\text{low}}} := (U_{\mathbf{t}})_{|\mathbf{t}|_{\delta-1} \le m^{\text{low}}, t_\delta = \tau, \mathcal{U}_\delta(\mathbf{t})=\mathbf{t}_u}$
   If $m^{\text{low}} > 0$ call BACK PREWAVELET ALGORITHM ( $\mathbf{U}^{\text{low}}_{m^{\text{low}}}, \delta - 1$)
  }
  2. Transform back in direction $\delta$:
  if $\tau > 0$ then {
   for every $\mathbf{t}$ with $t_\delta = \tau$ and $\mathcal{U}_\delta(\mathbf{t}) = \mathbf{t}_u$ do:
    $W_{\mathbf{t}} \xleftarrow{\text{set}} T^\delta_{\mathbf{t}}(\mathcal{B}(U_{\mathbf{t}}))$   // transform to nodal in direction $\delta$
    $U_{\mathbf{t}} \xleftarrow{\text{set}} \mathcal{B}(W_{\mathbf{t}}) + I^\delta_{\mathbf{t}}(\mathcal{B}(U_{\mathbf{t}-e_\delta}))$
  }
  3. Add interpolated prewavelets on coarse grid
  iterate for $t' = \tau, ..., 1$ {
   3.1. for every $\mathbf{t}$ with $t_\delta = t'$ and $\mathcal{U}_\delta(\mathbf{t}) = \mathbf{t}_u$ do:
    $W_{\mathbf{t}-e_\delta} \xleftarrow{\text{set}} I^\delta_{\mathbf{t}-e_\delta}(\mathcal{B}(W_{\mathbf{t}}))$   // coarse interpolation in direction $\delta$
    $U_{\mathbf{t}-e_\delta} \xleftarrow{\text{set}} \mathcal{B}(U_{\mathbf{t}-e_\delta}) + (\mathcal{B}(W_{\mathbf{t}-e_\delta}))$
   3.2. if $\delta > 1$ then {
    for every $\mathbf{t}$ with $|\mathbf{t}|_\delta = m$ and $t_\delta = t' - 1$ and $\mathcal{U}_\delta(\mathbf{t}) = \mathbf{t}_u$ do:
     $W_{\mathbf{t}} \xleftarrow{\text{set}} \sum_{\substack{(\alpha_1,...,\alpha_{\delta-1}) \in \{0,1\}^{\delta-1} \wedge \\ \boldsymbol{\alpha}=(\alpha_1,...,\alpha_{\delta-1},0...,0) \wedge \boldsymbol{\alpha} \neq (0,...,0)}} (-1)^{|\boldsymbol{\alpha}|_d+1} I_{\mathbf{t}}(\mathcal{B}(W_{\mathbf{t}-\boldsymbol{\alpha}}))$
     $U_{\mathbf{t}} \xleftarrow{\text{set}} \mathcal{B}(U_{\mathbf{t}}) + \mathcal{B}(W_{\mathbf{t}})$   // add local hierarchical surplus
    }
   }
  }
}
End of Algorithm.
```

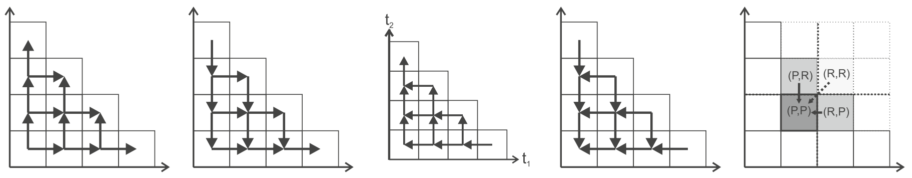
The prolongation part $a(\sum_{t' \le t} w_{t'}, v_t)$ is calculated in the function SPARSE GRID MATRIX MULTIPLICATION with input parameter $D = 0$. Here the essential calculations are done in Step 2.2 and Step 1. First $H_{\mathbf{t}}$ is prolongated on finer grids (see Step 2.2) and then the discretization stencil is applied (see Step 1).

The d-dimensional case is a combination of these two cases in all dimensional directions. In each direction $1 \le \delta \le d$, we either can perform a restriction or prolongation. All directions corresponding to restrictions are denoted by

$$R \subset \{1, ..., d\}.$$

Algorithm 3 Sparse grid matrix multiplication

```
Input: Let $\mathbf{U}_n = (U_{\mathbf{t}})_{|\mathbf{t}|_d \le n}$ be given in prewavelet format.
 $F_{\mathbf{t}} \xleftarrow{\text{set}} (\varphi_{\mathbf{t}} \mapsto 0,\ \varphi_{\mathbf{t}} \in W_{\mathbf{t}})\ \ \forall \mathbf{t}$    // set to zero
 for every $R \subset \{1, 2, ..., d\}$ do {    // $2^d$ cases
  Reorder the directions 1, ..., $d$ such that $R = \{1, 2, ..., D\}$,  $0 \le D \le d$.
  $\mathbf{H}_n := (H_{\mathbf{t}})_{|\mathbf{t}|_d \le n} = \mathbf{U}_n$
  Call SPARSE GRID MATRIX MULTIPLICATION ( $\mathbf{H}_n$, $d$, $D$).
 }
Output: $\mathbf{F}_n$ in complete prewavelet format.
Function SPARSE GRID MATRIX MULTIPLICATION ( $\mathbf{H}_m := (H_{\mathbf{t}})_{|\mathbf{t}|_\delta \le m, \mathcal{U}_\delta(\mathbf{t})=\mathbf{t}_u}$, $\delta$ , $D$)
if ($\delta == 0$) { // 1. Deepest point of recursion
 if ($D == 0$) do  $Z_{\mathbf{t}} \xleftarrow{\text{set}} (v \mapsto a(\mathcal{B}(H_{\mathbf{t}}), v),\ v \in V_{\mathbf{t}})$  // application of $3^d$-stencil of A
 else  $Z_{\mathbf{t}} \xleftarrow{\text{set}} \mathcal{B}(G_{\mathbf{t}})$  // cases that include restrictions
 $N_{\mathbf{t}} \xleftarrow{\text{set}} (\varphi \mapsto \mathcal{B}(Z_{\mathbf{t}})(\varphi),\ \varphi \in W_{\mathbf{t}})$    // transformation to prewavelets
 $F_{\mathbf{t}} \xleftarrow{\text{set}} \mathcal{B}(F_{\mathbf{t}}) + \mathcal{B}(N_{\mathbf{t}})$    // add up results of $2^d$ cases
} else if $\delta > D$ { // 2. Interpolate from coarser grids
 2.1. Recursive call on coarsest grid.
  Define $\mathbf{H}_m^{low} := (H_{\mathbf{t}})_{|\mathbf{t}|_{\delta-1} \le m, t_\delta = 0, \mathcal{U}_\delta(\mathbf{t})=\mathbf{t}_u}$
  call SPARSE GRID MATRIX MULTIPLICATION ( $\mathbf{H}_m^{low}$, $\delta - 1$, $D$)
 iterate for $\tau = 1, ..., m$ {
  2.2. Prolongate in direction $\delta$
  for every $\mathbf{t}$ with $t_\delta = \tau$ and $\mathcal{U}_\delta(\mathbf{t}) = \mathbf{t}_u$ do
   $H_{\mathbf{t}} \xleftarrow{\text{set}} I_{\mathbf{t}}^\delta(\mathcal{B}(H_{\mathbf{t}})) + I_{\mathbf{t}}^\delta(\mathcal{B}(H_{\mathbf{t}-e_\delta}))$
  2.3. Recursive call on finer grids.
  Define $m^{\text{low}} := m - \tau$. Define $\mathbf{H}_{m^{\text{low}}}^{\text{low}} := (H_{\mathbf{t}})_{|\mathbf{t}|_{\delta-1} \le m, t_\delta = \tau, \mathcal{U}_\delta(\mathbf{t})=\mathbf{t}_u}$
  if $m^{\text{low}} > 0$ call SPARSE GRID MATRIX MULTIPLICATION ( $\mathbf{H}_{m^{\text{low}}}^{\text{low}}$, $\delta - 1$, $D$)
 }
} else if $\delta \le D$ { // 3. Restrict fine prewavelet functions
 3.1. Recursive call on finest grid.
  $G_{\mathbf{t}} \xleftarrow{\text{set}} (v_{\mathbf{t}} \mapsto 0,\ v_{\mathbf{t}} \in V_{\mathbf{t}})\ \ \forall \mathbf{t}, t_\delta = m, \mathcal{U}_\delta(\mathbf{t}) = \mathbf{t}_u$    // set to zero
  Define $\mathbf{H}_0^{\text{low}} := (H_{\mathbf{t}})_{|\mathbf{t}|_{\delta-1} = 0, t_\delta = m, \mathcal{U}_\delta(\mathbf{t})=\mathbf{t}_u}$
  call SPARSE GRID MATRIX MULTIPLICATION ( $\mathbf{H}_0^{\text{low}}$, $\delta - 1$, $D$)
 iterate for $\tau = m - 1, ..., 0$ {
  3.2. Restrict right hand side.
  for every $\mathbf{t}$ with $t_\delta = \tau + 1$ and $\mathcal{U}_\delta(\mathbf{t}) = \mathbf{t}_u$ do:
   $Q_{\mathbf{t}} \xleftarrow{\text{set}} T_{\mathbf{t}}(\mathcal{B}(H_{\mathbf{t}}))$    // transformation to nodal basis
   $Z_{\mathbf{t}} \xleftarrow{\text{set}} (v \mapsto a(\mathcal{B}(Q_{\mathbf{t}}), v),\ v \in V_{\mathbf{t}})$    // application of $3^d$-stencil of A
   $N_{\mathbf{t}} \xleftarrow{\text{set}} \mathcal{B}(G_{\mathbf{t}}) + \mathcal{B}(Z_{\mathbf{t}})$    // add up functionals
   $G_{\mathbf{t}-e_\delta} \xleftarrow{\text{set}} (v \mapsto \mathcal{B}(N_{\mathbf{t}})(v),\ v \in V_{\mathbf{t}-e_\delta})$    // restriction
  Define $m^{\text{low}} := m - \tau$. Define $\mathbf{H}_{m^{\text{low}}}^{\text{low}} := (H_{\mathbf{t}})_{|\mathbf{t}|_{\delta-1} \le m^{\text{low}}, t_\delta = \tau, \mathcal{U}_\delta(\mathbf{t})=\mathbf{t}_u}$
  if $m^{\text{low}} > 0$ call SPARSE GRID MATRIX MULTIPLICATION ( $\mathbf{H}_{m^{\text{low}}}^{\text{low}}$, $\delta - 1$, $D$)
} }
End of Algorithm.
```



(a) (pro, pro) (b) (pro, res) (c) (res, pro) (d) (res, res) (e) Splitting

Fig. 6 Matrix vector multiplication on sparse grids in the two-dimensional case. **a**–**d** Calculation of part $P = \{1, 2\}$, $P = \{1\}$, $P = \{2\}$, $P = \emptyset$. **e** Summation of the result of the four parts

Now, the recursive structure of Algorithm 3 can be explained. Assume that u is given in the following form:

$$u = \sum_{|\mathbf{t}'|_d \leq n} w_{\mathbf{t}'}, \qquad \text{where } w_{\mathbf{t}'} = \sum_{\mathbf{i} \in \Xi_{\mathbf{t}}} \varphi_{\mathbf{t},\mathbf{i}} c_{\mathbf{t},\mathbf{i}}.$$

This form corresponds to a decomposition of u in its prewavelets. Then, a short calculation shows the following:

$$a(u, \varphi_{\mathbf{t},\mathbf{i}}) = \sum_{R \subset \{1,2,\ldots,d\}} \quad \sum_{s \in R} \sum_{t'_s > t_s} a \left(\sum_{s \notin R} \sum_{t'_s \leq t_s} w_{\mathbf{t}'}, \varphi_{\mathbf{t},\mathbf{i}} \right). \tag{13}$$

In the two-dimensional case, this sum leads to four parts:

$$\begin{aligned} a(u, \varphi_{(t_1,t_2),(i_1,i_2)}) &= \sum_{t'_1 > t_1,\ t'_2 > t_2} a\left(w_{(t'_1,t'_2)}, \varphi_{(t_1,t_2),(i_1,i_2)} \right) \\ &+ \sum_{t'_1 > t_1} a \left(\sum_{t'_2 \leq t_2} w_{(t'_1,t'_2)}, \varphi_{(t_1,t_2),(i_1,i_2)} \right) + \sum_{t'_2 > t_2} a \left(\sum_{t'_1 \leq t_1} w_{(t'_1,t'_2)}, \varphi_{(t_1,t_2),(i_1,i_2)} \right) \\ &+ a \left(\sum_{t'_1 \leq t_1,\ t'_2 \leq t_2} w_{(t'_1,t'_2)}, \varphi_{(t_1,t_2),(i_1,i_2)} \right). \end{aligned}$$

Figure 6 depicts the calculation of these four parts.

The d-dimensional case (13) consists of 2^d cases. Each is calculated by calling the function Sparse Grid Matrix Multiplication with input parameter d and D. Here, D is the number of restriction directions. In order to simplify the notation, the restriction directions are reordered such that

$$R = \{1, \ldots, D\}.$$

However, this is only a notational trick. An actual implementation of the algorithm should not perform such a reordering.

Observe that the recursive structure of Algorithm 3 leads to the following sequential calculations:

1. Prolongation in directions $D + 1, \ldots, d$. This is depicted in Fig. 7.
2. Application of discretization stencil on level $\mathbf{t}$ in Step 1 or Step 3.2.

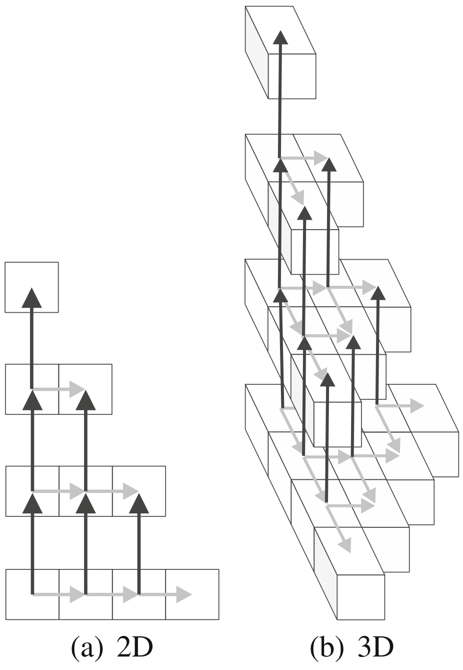


Fig. 7 Prolongation step. **a** Two-dimensional case. **b** Three-dimensional case

3. Restriction in direction $1, \ldots, D$, if $D \geq 1$

Now, two important questions are as follows:

Does Algorithm 3 correctly perform a matrix vector multiplication?
How is semi-orthogonality involved in this algorithm?

To answer these two questions, we restrict ourselves to the two-dimensional case and the computation of the part

$$\sum_{t_1' > t_1} a \left(\sum_{t_2' \leq t_2} w_{(t_1', t_2')}, \varphi_{(t_1, t_2), (i_1, i_2)} \right).$$

This means that $D = 1$ and $d = 2$. Clearly, the algorithm would be correct if we prolongated all of the data to a full grid, applied the discretization stencil, and then applied restrictions. Such a prolongation restriction scheme is depicted in Fig. 8a. Yet, this would lead to a computational inefficient algorithm since it requires the full grid to perform calculations. Instead, all computations are omitted which are not needed due to the semi-orthogonality property. To explain this in more detail, consider two overlapping basis functions as in Fig. 8d. Computations along the algorithmic path depicted in Fig. 8c are needed to take into account the corresponding value in the stiffness matrix $a_n^{\text{semi-ortho}}(\varphi_{\mathbf{t},\mathbf{i}}, \varphi_{\mathbf{t}',\mathbf{i}'})$. However, by using semi-orthogonality property, the result of these computations is zero. Therefore, this algorithmic path can be omitted. This shows that the remaining algorithmic paths depicted in Fig. 8b are

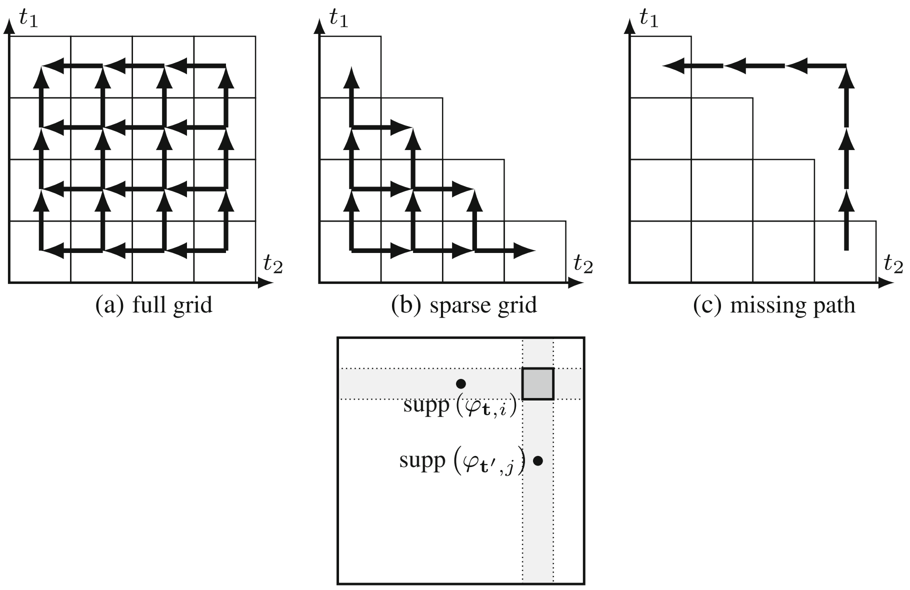


Fig. 8 Prolongation-restriction in 2D on a sparse grid ignores the overlapping of **t** and **t**′. Calculation would require a path over the full grid but can be ignored due to semi-orthogonality

enough for obtaining a correct computation. This proves that Algorithm 3 correctly performs the matrix vector multiplication using semi-orthogonality.

Performance

Finally, the computational costs of the algorithms Algorithm 1–Algorithm 3 are analyzed. Let $\mathrm{Op}(\delta, m)$ be the computational amount of one recursive call. By Fig. 9, this computational amount satisfies the following recursive formula:

$$\begin{aligned}\mathrm{Op}(\delta, m) &= \sum_{\tau=0}^{m} \mathrm{Op}(\delta-1, m-\tau) + O(2^{m-\tau}(m-\tau)^{\delta-2} \cdot 2^{n-(m-\tau)}) \\ &= \sum_{\tau=0}^{m} \mathrm{Op}(\delta-1, m-\tau) + O(2^{n}(m-\tau)^{\delta-2}).\end{aligned}$$

The factor $2^{m-\tau}(m-\tau)^{\delta-2}$ follows from the fact that for each τ, operations on a sparse grid of dimension $\delta - 1$ and depth $(m-\tau)$ are performed in each recursive call. Furthermore, the factor $2^{n-(m-\tau)}$ arises from the number of operations on a full grid of dimension $d-(\delta-1)$. This full grid has depth $(t_\delta, \cdots, t_d)$, where $\sum_{i=\delta}^{d} t_i = n-(m-\tau)$ (see Fig. 9) .

Now, a proof by induction shows that

$$\mathrm{Op}(\delta, m) = O(2^n m^{\delta-1}).$$

This implies that Algorithms 1–3 have the optimal complexity

$$O(2^n n^{d-1}).$$

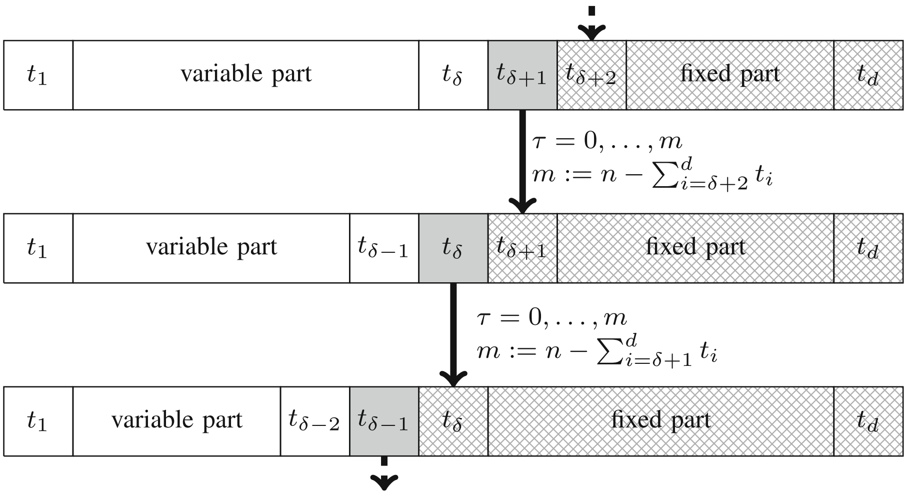


Fig. 9 Change of the depth vector **t** during a recursive call

Observe, that this estimate does not show how the computational amount increases with respect to the dimension d.

7 The complete iterative solver

Discretization 1 leads to the linear equation system

$$\mathcal{A}_n \mathbf{U} = \mathcal{M}\mathbf{F}, \tag{14}$$

where $\mathcal{A}_n$ is the stiffness matrix (12), $\mathcal{M}$ the mass matrix, **F** the right-hand side in prewavelet format, and **U** the solution vector in prewavelet format. By using the orthogonality property (2), $\mathcal{M}$ is reduced to a diagonal matrix.

For solving (14), the conjugate gradient method with a simple diagonal Jacobi precoditioner is applied. The condition number of $\mathcal{A}_n$, including this simple preconditioner, is $O(1)$. This follows from the multilevel theory in [13] and the following equivalence of norms:

$$\|u\|_{H^1}^2 \cong \sum_{|\mathbf{t}|_d \le n, \mathbf{i} \in \Xi_\mathbf{t}} \left|c_{\mathbf{t},\mathbf{i}}\right|^2 a(\varphi_{\mathbf{t},\mathbf{i}}, \varphi_{\mathbf{t},\mathbf{i}}),$$

where $u = \sum_{|\mathbf{t}|_d \le n, \mathbf{i} \in \Xi_\mathbf{t}} c_{\mathbf{t},\mathbf{i}} \varphi_{\mathbf{t},\mathbf{i}}$

Algorithm 1 is used to calculate the right-hand side vector **F** for a given right-hand side vector $(f(p))_{p \in \mathcal{D}_n}$. The conjugate gradient algorithm requires the application of the stiffness matrix multiplication Algorithm 3. The resulting vector **U** applied to Algorithm 2 leads to an approximation of the finite element solution $u_{\mathcal{D}_n}^{\text{prew}} \in V_{\mathcal{D}_n}^{\text{prew}}$ of Discretization 1. The total algorithm is depicted in Fig. 10.

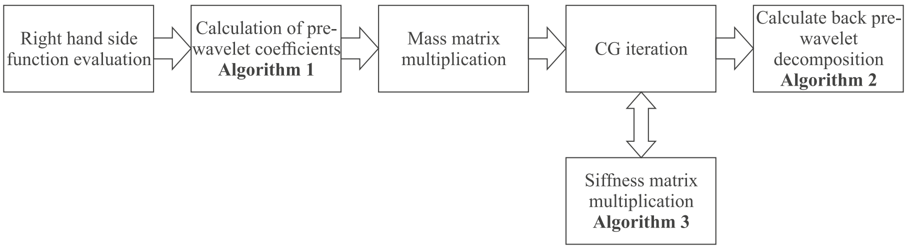

Right hand side function evaluation
Calculation of pre-wavelet coefficients Algorithm 1
Mass matrix multiplication
CG iteration
Calculate back pre-wavelet decomposition Algorithm 2
Siffness matrix multiplication Algorithm 3

Fig. 10 Preconditioned CG-solver using prewavelet decomposition

8 Numerical results

To show the efficiency of the discretization in this paper, two numerical results are presented. The first example shows that sparse grids can be used to discretize elliptic partial differential equations on curvilinear bounded domains in 3D. The second example is a six-dimensional Helmholtz problem with a variable coefficient. To our knowledge, this is the first Ritz-Galerkin finite element discretization of an elliptic PDE with variable coefficients in a higher-dimensional space.

In this paper, the discretization stencils were obtained by analytic calculations.

8.1 Poisson's equation

We want to show that sparse grids can be applied to discretize partial differential equations on curvilinear bounded domains. To this end, consider Poisson's equation

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega \subset \mathbb{R}^3 \\ u &= g \text{ on } \partial\Omega. \end{aligned} \tag{15}$$

In order to discretize this problem on a curvilinear bounded domain, one has to subdivide the domain Ω into several blocks, such that each of these blocks can smoothly be transformed to a unit cube. By these transformations to a unit cube, one obtains partial differential equations with variable coefficients on the unit cube. The basic idea of this concept is explained in [5, 7] for two-dimensional domains. To show that this concept can be extended to a three-dimensional domain, it is applied to the curvilinear bounded domain depicted in Fig. 11b. Simulation results are compared with simulations obtained on a simple cubical domain $[0, 1]^3$ (see Fig. 11a). The right-hand side f and inhomogeneous Dirichlet boundary condition g are chosen such that

$$u = \sin(x\pi)\sin(y\pi)\sin(z\pi)$$

is the exact solution of (15). The curvilinear bounded domain is obtained by the transformation of the x-coordinate and y-coordinate accordingly:

$$\begin{aligned} \tilde{x} &= \frac{1}{2}\cos\left(\frac{\pi}{2}\left(x + \frac{1}{2}\right)\right)(y+1) \\ \tilde{y} &= \frac{1}{2}\sin\left(\frac{\pi}{2}\left(x + \frac{1}{2}\right)\right)(y+1) \end{aligned} \tag{16}$$

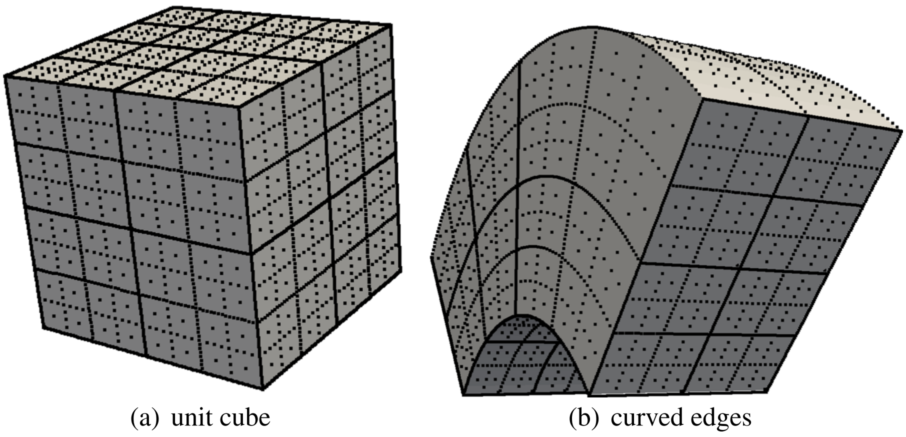


Fig. 11 Sparse grid with depth $n = 7$ on three-dimensional unit cube (**a**) and domain with curved edges (**b**)

Now, let u be the exact solution of Poisson's problem (15) and $u_{\mathcal{D}_n}^{\text{prew}}$ the finite element discretization (5) on the sparse grid $\mathcal{D}_n$ of depth n. Furthermore, let $e_{n,\infty} = \|u - u_{\mathcal{D}_n}^{\text{prew}}\|_\infty$ be the error in the maximum norm and $e_{n,2} = \|u - u_{\mathcal{D}_n}^{\text{prew}}\|_2$ the error in the L_2-norm. Tables 1, 2, 3, 4, and Fig. 12 show that the discretization with semi-orthogonality and prewavelets leads to an optimal convergence according the approximation properties of sparse grids. These approximation properties of sparse grids are described in detail in [4]. In particular, the convergence of the discretization error is of order $\mathcal{O}(h)$ with respect to the H_1-norm (see Table 4).

Moreover, the condition number of the stiffness matrix using a simple diagonal preconditioner stays far below 10 for $n = 2, ..., 9$ (see Table 5). Therefore, only a few cg-iterations are needed to obtain a small algebraic error.

8.2 Helmholtz equation with variable coefficients in PDE of a high dimensionality

Consider the six-dimensional Helmholtz problem

$$\begin{aligned} -\Delta u + cu &= f \text{ in } \Omega := [0,1]^6 \\ u &= 0 \text{ on } \partial\Omega \end{aligned} \tag{17}$$

Table 1 Discretization error for variable coefficient $c(x, y, z) = 1/(1 + x^2) + 1/(1 + y^2) + 1/(1 + y^2)$

n	DOF	$e_{n,\infty}$	$\frac{e_{n,\infty}}{e_{n-1,\infty}}$	$e_{n,2}$	$\frac{e_{n,2}}{e_{n-1,2}}$	e_{n,H^1}	$\frac{e_{n,H^1}}{e_{n-1,H^1}}$
4	111	3.04^{-2}		2.27^{-3}		6.79^{-2}	
5	351	1.03^{-2}	2.95	7.65^{-4}	2.96	3.80^{-2}	1.78
6	1023	3.24^{-3}	3.17	2.47^{-4}	3.10	2.00^{-2}	1.9
7	2815	9.74^{-4}	3.32	7.69^{-5}	3.21	1.02^{-2}	1.96
8	7423	2.96^{-4}	3.29	2.33^{-5}	3.30	5.15^{-3}	1.98
9	18,943	8.83^{-5}	3.35	6.92^{-6}	3.37	2.59^{-3}	1.99
10	47,103	2.61^{-5}	3.38	2.02^{-6}	3.43	1.29^{-3}	2.0

Table 2 L_∞-norm of discretization error

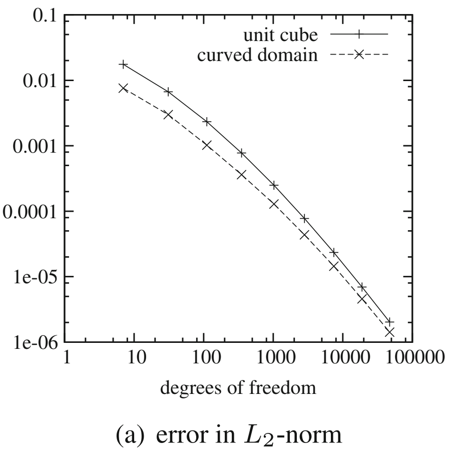
		Unit cube		Curved domain	
n	DOF	$e_{n,\infty}$	$\frac{e_{n,\infty}}{e_{n-1,\infty}}$	$e_{n,\infty}$	$\frac{e_{n,\infty}}{e_{n-1,\infty}}$
2	7	1.41^{-1}		3.94^{-2}	
3	31	7.84^{-2}	1.80	2.23^{-2}	1.77
4	111	3.08^{-2}	2.54	1.04^{-2}	2.14
5	351	1.04^{-2}	2.96	4.02^{-3}	2.59
6	1023	3.27^{-3}	3.19	1.34^{-3}	3.00
7	2815	9.83^{-4}	3.33	4.23^{-4}	3.17
8	7423	2.98^{-4}	3.29	1.28^{-4}	3.30
9	18,943	8.87^{-5}	3.37	3.77^{-5}	3.40
10	47,103	2.62^{-5}	3.38	1.08^{-5}	3.48

Table 3 L_2-norm of discretization error

		Unit cube		Curved domain	
n	DOF	$e_{n,2}$	$\frac{e_{n,2}}{e_{n-1,2}}$	$e_{n,2}$	$\frac{e_{n,2}}{e_{n-1,2}}$
2	7	1.75^{-2}		7.58^{-3}	
3	31	6.66^{-3}	2.63	2.99^{-3}	2.53
4	111	2.33^{-3}	2.86	1.02^{-3}	2.92
5	351	7.78^{-4}	3.00	3.61^{-4}	2.83
6	1023	2.50^{-4}	3.12	1.29^{-4}	2.81
7	2815	7.75^{-5}	3.22	4.36^{-5}	2.95
8	7423	2.35^{-5}	3.30	1.45^{-5}	3.02
9	18,943	6.95^{-6}	3.37	4.59^{-6}	3.15
10	47,103	2.02^{-6}	3.43	1.42^{-6}	3.23

Table 4 H_1-norm of discretization error

		Unit cube		Curved domain	
n	DOF	e_{n,H^1}	$\frac{e_{n,H^1}}{e_{n-1,H^1}}$	e_{n,H^1}	$\frac{e_{n,H^1}}{e_{n-1,H^1}}$
2	7	1.36^{-1}		6.43^{-2}	
3	31	1.07^{-1}	1.27	3.44^{-2}	1.87
4	111	6.81^{-2}	1.57	2.54^{-2}	1.35
5	351	3.81^{-2}	1.79	1.60^{-2}	1.59
6	1023	2.00^{-2}	1.90	9.23^{-3}	1.73
7	2815	1.02^{-2}	1.96	4.87^{-3}	1.89
8	7423	5.16^{-3}	1.98	2.51^{-3}	1.94
9	18,943	2.59^{-3}	1.99	1.27^{-3}	1.97
10	47,103	1.29^{-3}	2.00	6.41^{-4}	1.99



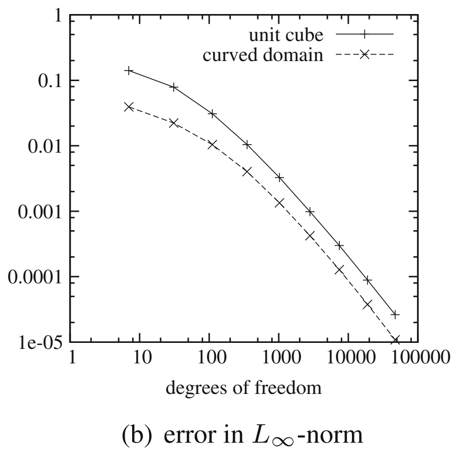


Fig. 12 Convergence of the error norm measured by L_2 (**a**) and L_∞ (**b**)

Table 5 Condition number of prewavelet discretization and diagonal preconditioning

		Unit cube		Curved domain	
n	DOF	$\kappa(A)$	$\kappa\left(C^T AC\right)$	$\kappa(A)$	$\kappa\left(C^T AC\right)$
2	7	2.96	1.62	7.70	1.63
3	31	8.48	2.87	33.07	2.59
4	111	18.92	3.39	156.75	2.85
6	1023	150.08	4.01	454.82	3.14
7	2815	411.77	4.59	673.84	3.42
8	7423	670.33	5.11	836.71	3.61
9	18,943	780.90	5.47	880.15	3.79

Table 6 Error convergence for a six-dimensional Helmholtz problem

		Constant coefficient		Variable coefficient	
n	DOF	$e_{n,\infty}$	$\frac{e_{n,\infty}}{e_{n-1,\infty}}$	$e_{n,\infty}$	$\frac{e_{n,\infty}}{e_{n-1,\infty}}$
2	13	0.40427		0.404078	
3	97	0.28396	1.42	0.283793	1,42
4	545	0.10686	2.65	0.107563	2.64
5	2561	0.04028	2.65	0.040271	2.67
6	10,625	0.01533	2.63	0.015336	2.63
7	40,193	0.00566	2.71	0.005662	2.71
8	141,569	0.002129	2.66	0.002129	2.66
9	471,041	0.000788	2.70	0.000788	2.70
10	1,496,065	0.000279	2.82	0.000279	2.82

Table 7 Error convergence for a six-dimensional Helmholtz problem

		Constant coefficient		Variable coefficient	
n	DOF	$e_{n,2}$	$\frac{e_{n,2}}{e_{n-1,2}}$	$e_{n,2}$	$\frac{e_{n,2}}{e_{n-1,2}}$
2	13	0.006371		0.017708	
3	97	0.004383	1.45	0.013738	1.29
4	545	0.002518	1.74	0.005889	2.33
5	2561	0.001265	1.99	0.002259	2.61
6	10,625	0.000576	2.20	0.000878	2.57
7	40,193	0.000243	2.37	0.000347	2.53
8	141,569	0.000097	2.50	0.000137	2.54
9	471,041	0.000037	2.62	0.000053	2.60
10	1,496,065	0.000014	2.72	0.000020	2.66

with variable coefficient

$$c\,(x, y, z, u, v, w) := \left(1 - x^2\right)\left(1 - y^2\right)\left(1 - z^2\right)\left(1 - u^2\right)\left(1 - v^2\right)\left(1 - w^2\right). \tag{18}$$

The right-hand side f is chosen such that

$$u = \sin(x\pi)\sin(y\pi)\sin(z\pi)\sin(u\pi)\sin(v\pi)\sin(w\pi)$$

is a solution of (17).

Table 6 shows the discretization error for constant coefficients ($c = 1$) as well as for the variable coefficient (18) using the maximum norm and Table 7 for the L_2-norm. The convergence rate for the problem with constant coefficients is similar to the reported convergence behavior in [21]. In addition, the solution of the discretization with semi-orthogonality and prewavelet convergences in the case of variable coefficients is as fast as in the case of constant coefficients. This shows that the discretization with semi-orthogonality does not introduce any remarkable additional errors.

Acknowledgments The authors gratefully acknowledge funding of the Erlangen Graduate School in Advanced Optical Technologies (SAOT) by the German Research Foundation (DFG) in the framework of the German excellence initiative.

References

1. Achatz, S.: Higher order sparse grid methods for elliptic partial differential equations with variable coefficients. Computing **71**(1), 1–15 (2003)
2. Balder, R., Zenger, C.: The solution of multidimensional real Helmholtz equations of sparse grids. SIAM J. Sci. Comp. **17**(3), 631–646 (1996)
3. Barinka, A., Barsch, T., Charton, P., Cohen, A., Dahlke, S., Dahmen, W., Urban, K.: Adaptive wavelet schemes for elliptic problems—implementation and numerical experiments. SIAM J. Sci. Comput. **23**(3), 910–939 (2002)

4. Bungartz, H.-J.: Dùnne Gitter und deren Anwendung bei der adaptiven Lòsung der dreidimensionalen Poisson-Gleichung. Dissertation, Fakultàt für Informatik, Technische Universitàt Mùnchen (1992)
5. Dornseifer, T., Pflaum, C.: Discretization of elliptic differential equations on curvilinear bounded domains with sparse grids. Computing **56**(3), 197–213 (1996)
6. Feuersànger, C.: Sparse grid methods for higher dimensional approximation. PhD thesis, University of Bonn (2010)
7. Gordon, W.J., Hall, C.A.: Construction of curvilinear co-ordinate systems and applications to mesh generation. Int. J. Numer. Methods Eng. **7**(4), 461–477 (1973)
8. Griebel, M.: Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. Computing **61**, 151–179 (1998)
9. Griebel, M., Hamaekers, J.: A wavelet based sparse grid method for electronic Schrödinger equation. In: Sanz-Solë, M., Soria, J., Varona, J.L., Verdera, J. (eds.) Proceedings of the International Congress of Mathematicians Madrid, August 22–30, 2006, vol. 1, pp. 1473–1506. European Mathematical Society, Publishing House (2007)
10. Griebel, M., Hullmann, A.: On a multilevel preconditioner and its condition numbers for the discretized laplacian on full and sparse grids in higher dimensions. In: Singular Phenomena and Scaling in Mathematical Models, pp. 263–296. Springer International Publishing (2014)
11. Kestler, S., Stevenson, R.: Fast evaluation of system matrices w.r.t. multi-tree collections of tensor product refinable basis functions. J. Comput. Appl Math. **260**, 103–116 (2014)
12. Niedermeier, A., Zimmer, S.: Implementational aspects of prewavelet sparse grid methods. In: Lai, C.-H., Bjoerstad, P., Cross, M., Widlund, O. (eds.) Eleventh International Conference of Domain Decomposition Methods (1999)
13. Oswald, P.: Multilevel finite element approximation. Teubner Skripten zur Numerik. Teubner, Stuttgart (1994)
14. Peherstorfer, B., Zimmer, S., Zenger, C., Bungartz, H.-J.: A multigrid method for adaptive sparse grids. SIAM J. Sci. Comput. **37**(5), 51–70 (2015)
15. Pflaum, C.: Diskretisierung elliptischer Differentialgleichungen mit dünnen Gittern. Dissertation, Technische Universität München (1995)
16. Pflaum, C.: A multilevel algorithm for the solution of second order elliptic differential equations on sparse grids. Numer. Math. **79**, 141–155 (1998)
17. Pflaum, C.: Robust convergence of multilevel algorithms for convection-diffusion equations. SIAM J. Numer. Anal. **37**(2), 443–469 (2000)
18. Pflaum, C., Hartmann, R.: A sparse grid discretization of the helmholtz equation with variable coefficients in high dimensions. SIAM J. Numer. Anal. **54**(4), 2707–2727 (2016)
19. Vassilevski, P.S., Wang, J.: Stabilizing the hierarchical basis by approximate wavelets II implementation and numerical results. SIAM J. Sci. Comput. **20**(2), 490–514 (1998)
20. Zeiser, A.: Fast matrix-vector multiplication in the sparse-grid Galerkin method. J. Sci. Comput. **47**(3), 328–346 (2011)
21. Zenger, C.: Sparse grids. In: Hackbusch, W. (ed.) Proceedings of the Sixth GAMM-Seminar Parallel Algorithms for Partial Differential Equations, Kiel, 1990, vol. 31 of Notes on Numerical Fluid Mechanics, pp. 240–251. Kiel, Vieweg (1991)