Noname manuscript No.
(will be inserted by the editor)

On the algebraic construction of sparse multilevel approximations of elliptic tensor product problems

Helmut Harbrecht · Peter Zaspel

Received: date / Accepted: date

Abstract We consider the solution of elliptic problems on the tensor product of two physical domains as e.g. present in the approximation of the solution covariance of elliptic partial differential equations with random input. Previous sparse approximation approaches used a geometrically constructed multilevel hierarchy. Instead, we construct this hierarchy for a given discretized problem by means of the algebraic multigrid method (AMG). Thereby, we are able to apply the sparse grid combination technique to problems given on complex geometries and for discretizations arising from unstructured grids, which was not feasible before. Numerical results show that our algebraic construction exhibits the same convergence behaviour as the geometric construction, while being applicable even in black-box type PDE solvers.

Keywords Elliptic boundary value problem · Sparse tensor product approximation · Combination technique · Algebraic multigrid · Uncertainty quantification

Mathematics Subject Classification (2000) 65N30 · 65N22 · · 65N55 · 65N50 · 65F10 · 65Y20 · 65Y05

1 Introduction

The solution of elliptic problems on tensor products of a polygonally bounded domain $\Omega \subset \mathbb{R}^d$ with e.g. $d = 2, 3$ given by

$$\begin{aligned} (\Delta \otimes \Delta) u &= f \quad \text{on } \Omega \times \Omega\,, \\ u &= 0 \quad \text{on } \partial(\Omega \times \Omega)\,, \end{aligned}$$

This work is funded by the Swiss National Science Foundation (SNF) under project number 407540_167186.

H. Harbrecht, P. Zaspel
Departement für Mathematik und Informatik, Unversität Basel
Spiegelgasse 1, 4051 Basel, Schweiz
E-mail: {helmut.harbrecht,peter.zaspel}@unibas.ch

is an important high-dimensional problem. As an example, this problem shows up in the estimation of the output covariance of an elliptic partial differential equation with random input data that is given on a domain Ω, see [13,15,20,21] for example. The problem becomes high-dimensional since the dimensionality of the elliptic problem on Ω is doubled. In case of real-world problems in $d = 3$, we end up solving a six-dimensional problem, which might become prohibitively expensive.

Recently, there have been developments to overcome this strong limitation. These developments are based on the introduction of a *geometrically constructed multilevel frame* to solve the elliptic problem on Ω. Standard Galerkin discretizations of this problem approximate the solution with respect to a *basis* of a finite-dimensional trial and test space V_J associated to a triangulation $\mathcal{T}_J$ of the domain Ω. A multi-level frame discretization uses more functions to construct the trial and test space. In fact, it uses all basis functions of a (nested) hierarchy of subspaces $V_0 \subset V_1 \subset \ldots \subset V_J$, which are associated to a (nested) *geometric* hierarchy of triangulations $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_J$ with an increasing number of nodes $|\mathcal{T}_0| < |\mathcal{T}_1| < \ldots < |\mathcal{T}_J|$. This set with many redundant basis functions is no longer a basis for V_J, but a *frame*.

The multilevel frame gives rise to a sparse approximation with respect to the interaction of the involved domains in $\Omega \times \Omega$ [16,20,21]. It has been shown that the sparse approximation, i.e. using the trial and test space $\bigcup_{0 \leq j+j' \leq J} V_j \otimes V_{j'}$ instead of $V_J \otimes V_J$, allows to solve the tensor product problem at a computational complexity that stays essentially (i.e. up to a poly-logarithmic factor) proportional to the number of degrees of freedom to discretize a function on the single domain Ω with respect to the trial space V_J. In a more recent work by one of the authors [15], it has been shown that the sparse approximation can equivalently be replaced by the sparse grid combination technique [4,10,12,17], which combines cheap anisotropic full-grid solutions of the tensor-product elliptic problem. This further reduces the computational work and facilitates the implementation.

However, the currently available geometric construction of the multilevel hierarchy imposes limitations on the discretization for real-world problems. First, the coarsest triangulation $\mathcal{T}_0$ in the geometrical hierarchy of triangulations has to fully represent the boundary of the geometry Ω. This either limits the types of geometry to consider or the computational efficiency (in case even the coarsest mesh has to be fine at the boundary). Second, the use of a fully unstructured mesh $\mathcal{T}_J$ becomes barely possible, since we are missing a coarsening strategy for such a mesh.

This work introduces *algebraically* constructed multilevel hierarchies [8,11,25] for the solution of elliptic problems on tensor product domains. While previous works [15,16] first constructed the multilevel hierarchy of meshes or triangulations and then discretized the problem by finite elements, the new approach first discretizes the problem on Ω on the finest (potentially unstructured) mesh $\mathcal{T}_J$ and then constructs coarser versions of the linear system resulting from the fine discretization. The coarser problems are generated using algebraic coarsening known from the classical *Ruge-Stüben algebraic multigrid* (AMG) [19,22]. The algebraic construction of multilevel hierarchies for frames has been previously discussed in context of optimal complexity solvers for elliptic problems in [25]. However, it has not been applied in the context of sparse approximation yet. Note that, by construction, our new approach allows us to overcome both the limitations in pres-

ence of complex geometries and the requirements on the structure of the mesh. Moreover, it perfectly fits into the context of black-box type PDE solvers.

As it is well-known, a full theory for algebraic multigrid methods, especially in the multilevel context and on unstructured grids, is still to be developed. Nevertheless, this technique is extremely popular as solver in real-world applications and, usually, empirically shows the same performance as geometric multigrid. This work follows the same spirit and focuses on the formal construction and the empirical analysis of the resulting numerical method. Thereby, we are able to match the convergence results available for geometrically constructed sparse approximations, while being able to apply this approach to complex geometries and unstructured grids in a black-box fashion.

In Section 2, the algebraic multilevel construction is outlined. This construction is introduced to the tensor product problem with sparse approximation and the sparse grid combination technique in Section 3. Section 4 briefly discusses the implementation. In Section 5, we give a series of numerical examples with empirical error analysis. Finally, Section 6 summarizes this work.

2 Algebraic multilevel constructions

In our algebraic construction, we aim at replacing classical multilevel discretizations for elliptic partial differential equations by a purely matrix-based construction. That is, we consider an elliptic partial differential equation

$$\begin{aligned} -\Delta u &= f \quad \text{on } \Omega \\ u &= 0 \quad \text{on } \partial\Omega \end{aligned} \tag{1}$$

on a polygonally bounded domain $\Omega \subset \mathbb{R}^d$. This problem has been discretized by some method on a discretization level J, leading to a system of linear equations

$$\boldsymbol{A}_J \boldsymbol{u}_J = \boldsymbol{f}_J \,, \tag{2}$$

where $\boldsymbol{A}_J \in \mathbb{R}^{N_J \times N_J}$ is an *M-matrix* and $\boldsymbol{u}_J, \boldsymbol{f}_J \in \mathbb{R}^{N_J}$. Note that an M-matrix has positive diagonal entries, non-positive non-diagonal entries, is non-singular and the entries of its inverse are non-negative. In case of the discretization by finite elements, $\boldsymbol{A}_J$ corresponds to the stiffness matrix and $\boldsymbol{f}_J$ is the load vector, obtained by, for example, using the mass matrix $\boldsymbol{M}_J$ and interpolation. Moreover, we identify each variable $u_{J,i}$ in $\boldsymbol{u}_J = (u_{J,1} \dots u_{J,N_J})^\top$ by its index i and introduce the corresponding index set $\mathcal{D}_J := \{1, \dots, N_J\}$ for discretization level J.

2.1 Multilevel hierarchy of discretized problems

The objective is to construct from (2) a hierarchy of systems of linear equations

$$\boldsymbol{A}_j \boldsymbol{u}_j = \boldsymbol{f}_j \,, \quad j = 0, \dots, J \,, \tag{3}$$

which are similar to discretizations on different geometric refinement levels. Especially, we intend to do this in a purely matrix-based, i.e. algebraic, way by using

Algorithm 1 Standard coarsening algorithm [23]

Require: level j
function AMGstandardCoarsening
  $\mathcal{F}_j := \emptyset, \mathcal{D}_{j-1} := \emptyset, \mathcal{U}_j := \mathcal{D}_j$
  for $i \in \mathcal{U}_j$ **do**
    $\lambda_j(i) := \left|\mathcal{S}_j(i)^\top \cap \mathcal{U}_j\right| + 2\left|\mathcal{S}_j(i)^\top \cap \mathcal{F}_j\right|$
  while $\exists i$ s.th. $\lambda_j(i) \neq 0$ **do**
    find $i_{\max} := \arg\max_i \lambda_j(i)$
    $\mathcal{D}_{j-1} := \mathcal{D}_{j-1} \cup \{i_{\max}\}$
    $\mathcal{U}_j := \mathcal{U}_j \setminus \{i_{\max}\}$
    for $k \in (\mathcal{S}_j(i_{\max})^\top \cap \mathcal{U}_j)$ **do**
      $\mathcal{F}_j := \mathcal{F}_j \cup \{k\}$
      $\mathcal{U}_j := \mathcal{U}_j \setminus \{k\}$
    for $i \in U_j$ **do**
      $\lambda_j(i) := \left|\mathcal{S}_j(i)^\top \cap \mathcal{U}_j\right| + 2\left|\mathcal{S}_j(i)^\top \cap \mathcal{F}_j\right|$
  return $\mathcal{D}_{j-1}, \mathcal{F}_j$

coarsening and transfer operators from algebraic multigrid (AMG) [22]. To this end, we first introduce a construction method for a hierarchy of variable sets

$$\mathcal{D}_0 \subset \mathcal{D}_1 \subset \ldots \subset \mathcal{D}_J \tag{4}$$

of sizes

$$N_0 \leq N_1 \leq \ldots \leq N_J\,.$$

In classical Ruge-Stüben AMG [19,23], this is achieved by recursively splitting the set of variables $\mathcal{D}_j$ on level j into a set of coarse and fine grid variables

$$\mathcal{D}_j = \mathcal{D}_{j-1} \uplus \mathcal{F}_j\,,$$

where "$\uplus$" is the union of two disjoint sets. Each fine grid variable is supposed to be in the neighborhood of an appropriate amount of strongly negatively coupled coarse grid variables, where we define the *neighborhood* of a variable $i \in \mathcal{D}_j$ by

$$\mathcal{N}_j(i) := \{i' \in \mathcal{D}_j : i' \neq i,\ a_{j,ii'} \neq 0\}\,,$$

where $\boldsymbol{A}_j = (a_{j,ii'})_{i,i'=1}^{N_j}$. That is, we consider neighborhoods between variables by reinterpreting the system matrix $\boldsymbol{A}_j$ as the adjacency matrix of a graph with edges between nodes for each non-zero matrix entry. Moreover, the set of neighboring strongly negatively coupled variables of a variable i is

$$\mathcal{S}_j(i) := \left\{i' \in \mathcal{N}_j(i) : -a_{j,ii'} \geq \epsilon_{str} \max_k |a_{j,ik}|\right\}$$

with a strength measure $0 < \epsilon_{str} < 1$. The *standard coarsening* procedure, cf. Algorithm 1 [23], builds an appropriate splitting $\mathcal{D}_j = \mathcal{D}_{j-1} \uplus \mathcal{F}_j$ based on these considerations. It also involves the sets $\mathcal{S}_j(i)^\top$, which are given by

$$\mathcal{S}_j(i)^\top := \{i' \in \mathcal{D}_j : i \in \mathcal{S}_j(i')\}\,.$$

Algorithm 1 uses the notation $|\cdot|$ to express the cardinality of a set.

In order to define the hierarchy of linear systems (3), we further need a means to transfer information between two consecutive levels j and $j+1$. This is done by prolongation operators $\boldsymbol{P}_j^{j+1} \in \mathbb{R}^{N_{j+1} \times N_j}$ and restriction operators $\boldsymbol{P}_{j+1}^{j} \in \mathbb{R}^{N_j \times N_{j+1}}$. Prolongation and restriction are done in a purely algebraic way based on AMG. In *standard interpolation* [23], which is one possible type of algebraic prolongation, data given on a fine grid node $i \in \mathcal{F}_j$ is interpolated from the set of interpolatory variables

$$\mathcal{I}_j(i) := (\mathcal{D}_{j-1} \cap \mathcal{S}_j(i)) \cap \left(\bigcup_{i' \in \mathcal{F}_j \cap \mathcal{S}_j(i)} \left(\mathcal{D}_{j-1} \cap \mathcal{S}_j(i') \right) \right) .$$

Thus, it is interpolated from strongly negatively coupled coarse grid points and all coarse grid points that are strongly negatively coupled to strongly negatively coupled fine grid points. The exact choice of prolongation / interpolation weights is known from literature [23]. If the quality of the resulting algebraic interpolation is not good enough, one might also apply one or several steps of *Jacobi interpolation* [23]. This, roughly speaking, extends the whole set of interpolatory variables $\mathcal{I}_j(i)$ of a node i by one layer of additional neighboring nodes. *Truncation* allows to drop some interpolatory variables based on a threshold [23]. *Restriction* is given as the transpose of the prolongation, i.e. $\boldsymbol{P}_{j+1}^{j} = \boldsymbol{P}_j^{j+1\top}$.

Finally, we recursively define for $j = J-1, \ldots, 0$ the matrices and the right-hand sides involved in the hierarchy of linear systems (3) as

$$\boldsymbol{A}_j := \boldsymbol{P}_{j+1}^{j} \boldsymbol{A}_{j+1} \boldsymbol{P}_j^{j+1}, \quad \boldsymbol{f}_j := \boldsymbol{P}_{j+1}^{j} \boldsymbol{f}_{j+1} ,$$

which can also be directly expressed in terms of prolongations and restrictions from $\boldsymbol{A}_J$ and $\boldsymbol{f}_J$ as

$$\boldsymbol{A}_j := \boldsymbol{P}_{j+1}^{j} \cdots \boldsymbol{P}_J^{J-1} \boldsymbol{A}_J \boldsymbol{P}_{J-1}^{J} \cdots \boldsymbol{P}_j^{j+1}, \quad \boldsymbol{f}_j := \boldsymbol{P}_{j+1}^{j} \cdots \boldsymbol{P}_J^{J-1} \boldsymbol{f}_J .$$

Later on, we will also use the abbreviations

$$\boldsymbol{P}_J^{j} := \boldsymbol{P}_{j+1}^{j} \cdots \boldsymbol{P}_J^{J-1}, \quad \boldsymbol{P}_j^{J} = \boldsymbol{P}_{J-1}^{J} \cdots \boldsymbol{P}_j^{j+1} . \tag{5}$$

Optimal complexity in AMG can be achieved, if coarser levels are constructed such that the *operator complexity*

$$C_{\boldsymbol{A}} := \sum_j \frac{\eta(\boldsymbol{A}_j)}{\eta(\boldsymbol{A}_J)} ,$$

where $\eta(\boldsymbol{A}_J)$ is the number of non-zeros in $\boldsymbol{A}_J$, stays bounded by some constant independent of J. Standard coarsening together with standard interpolation empirically fulfill this property for model problems discretized on simple geometries. However, in more complex situations, it might happen that standard interpolation and standard coarsening fail in achieving this. Then, stronger or more aggressive versions such as *extended / multi-pass interpolation* and *aggressive coarsening* on some levels are applied to keep this empirical property [24]. In fact, it might become necessary to use the operator complexity as indicator function in a manual optimization process in which several combinations of coarsenings and interpolation schemes are tried until an acceptable operator complexity is reached. Unfortunately, to the best of the authors' knowledge, there is for now no theory on the

Algorithm 2 V-cycle in a multigrid scheme

Require: $\boldsymbol{A}_0, \ldots, \boldsymbol{A}_J, \boldsymbol{P}_0^1, \ldots, \boldsymbol{P}_{J-1}^J, \boldsymbol{P}_J^{J-1}, \ldots, \boldsymbol{P}_1^0$

function VCycle($\boldsymbol{u}_j$, $\boldsymbol{b}_j$, j)
  if j=0 **then**
    return $\boldsymbol{A}_j^{-1}\boldsymbol{b}_j$ ▷ direct solve on coarsest level
  else
    $\boldsymbol{u}_j = \text{SMOOTHER}(\boldsymbol{u}_j, \boldsymbol{b}_j)$ ▷ pre-smoothing
    $\boldsymbol{r}_{j-1} = \boldsymbol{P}_j^{j-1}(\boldsymbol{b} - \boldsymbol{A}_j\boldsymbol{u}_j)$ ▷ restriction
    $\boldsymbol{u}_{j-1} = \text{VCycle}(\boldsymbol{0}, \boldsymbol{r}_{j-1}, j-1)$ ▷ coarse grid correction
    $\boldsymbol{u}_j = \boldsymbol{u}_j + \boldsymbol{P}_{j-1}^j \boldsymbol{u}_{j-1}$ ▷ prolongation
    $\boldsymbol{u}_j = \text{SMOOTHER}(\boldsymbol{u}_j, \boldsymbol{b}_j)$ ▷ post-smoothing
    return $\boldsymbol{u}_j$

decay of the number of non-zeros in the coarse grid matrices $\boldsymbol{A}_j$ constructed by classical Ruge-Stüben AMG on multiple levels and for general M matrices $\boldsymbol{A}_J$, which could simplify this process.

In classical literature on algebraic multigrid, the hierarchy of system matrices, prolongation operators, and restriction operators

$$\boldsymbol{A}_0, \ldots, \boldsymbol{A}_J, \quad \boldsymbol{P}_0^1, \ldots, \boldsymbol{P}_{J-1}^J, \quad \boldsymbol{P}_J^{J-1}, \ldots, \boldsymbol{P}_1^0,$$

are used in an iterative method with, e.g., a V-cycle, cf. Algorithm 2.1, in order to solve the linear system (2) with optimal constant or logarithmically growing number of iterations. Instead, we will use it for the construction of a multi-level hierarchy of problems as required by sparse multilevel approximations.

2.2 Multilevel frames

Let us note here that the above algebraic construction naturally leads to *algebraic multilevel frames*, cf. [25], for the elliptic problem on Ω. That is, we formally introduce the system of linear equations

$$\boldsymbol{A_J u_J} = \boldsymbol{f_J} \tag{6}$$

with

$$\boldsymbol{A_J} := \begin{pmatrix} \boldsymbol{A}_{11} & \cdots & \boldsymbol{A}_{1J} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{J1} & \cdots & \boldsymbol{A}_{JJ} \end{pmatrix}, \quad \boldsymbol{u_J} := \begin{pmatrix} \boldsymbol{u}_0 \\ \vdots \\ \boldsymbol{u}_J \end{pmatrix}, \quad \boldsymbol{f_J} := \begin{pmatrix} \boldsymbol{f}_0 \\ \vdots \\ \boldsymbol{f}_J \end{pmatrix}$$

and set

$$\boldsymbol{A}_{j_1 j_2} = \boldsymbol{P}_{j_1}^J \boldsymbol{A}_J \boldsymbol{P}_J^{j_2}.$$

The diagonal matrices $\boldsymbol{A}_{jj}$ are the system matrices $\boldsymbol{A}_j$ from the previous section. Moreover, we use (5) to extend prolongation / restriction to arbitrary levels. We further introduce the multi-index $\boldsymbol{j} = (j_1, j_2)$ allowing the abbreviated notation

$$\boldsymbol{A_J} = [\boldsymbol{A_j}]_{\|\boldsymbol{j}\|_{\ell^\infty} \le J}, \quad \boldsymbol{u_J} = [\boldsymbol{u}_j]_{|j| \le J}, \quad \boldsymbol{f_J} = [\boldsymbol{f}_j]_{|j| \le J}.$$

Note that matrix $\boldsymbol{A_J}$ has a large kernel. However, it can be ignored when solving (6) by using appropriate iterative linear solvers. The projection matrix

$$\mathcal{P}_{\boldsymbol{J}} = \left[\boldsymbol{P}_0^J, \boldsymbol{P}_1^J, \ldots, \boldsymbol{P}_J^J\right]$$

can be used to transfer the right-hand side $\boldsymbol{f}_J$ from (2) to the multi-level representation $\boldsymbol{f}_{\boldsymbol{J}}$ and to project back solutions $\boldsymbol{u}_{\boldsymbol{J}}$ to the single-level solutions $\boldsymbol{u}_J$. This is done by

$$\boldsymbol{u}_J = \mathcal{P}_{\boldsymbol{J}} \boldsymbol{u}_{\boldsymbol{J}} = \sum_{j=0}^{J} \boldsymbol{P}_j^J \boldsymbol{u}_j, \qquad \boldsymbol{f}_{\boldsymbol{J}} = \mathcal{P}_{\boldsymbol{J}}^\top \boldsymbol{f}_J = \left[\boldsymbol{P}_J^0 \boldsymbol{f}_J, \ldots, \boldsymbol{P}_J^J \boldsymbol{f}_J \right]^\top . \tag{7}$$

Using the linear system (6) together with the transfer operations from (7) instead of using linear system (2) conceptually corresponds to replacing a single-level discretization by a multi-level frame discretization. As in multilevel frame discretizations based on geometric refinements / coarsening, cf. [16], the above system of linear equations is much larger, since it encodes the full information of the hierarchy of systems in (3). However, it has the big advantage that the application of standard iterative solvers such as Jacobi, Gauss-Seidel or CG to (6) immediately leads to the same convergence behavior (in terms of the number of iterations) as if these solvers were applied with a BPX-preconditioner to (2). A Gauss-Seidel method applied to (6) could e.g. converge as fast as a multigrid method with Gauss-Seidel smoother applied to (2).

From a theoretical point of view, it has been formally shown for *geometric* multi-level constructions, that (6) is equivalent to the linear system of equations (2), if the BPX-preconditioner is applied in the solution process, cf. [2,5,7,18]. In [25], it has been further shown by numerical experiments that the application of specific iterative solvers to (6) leads to problem-size independent convergence rates for the here discussed case of *algebraically* constructed multilevel frames.

3 Sparse algebraic tensor product approach

Next, we like to consider elliptic problems on tensor products $\Omega \times \Omega$ of the polygonally bounded domain Ω. That is, we consider problems of the form

$$\begin{aligned} (\Delta \otimes \Delta) u &= f \quad \text{on } \Omega \times \Omega\,, \\ u &= 0 \quad \text{on } \partial(\Omega \times \Omega)\,. \end{aligned} \tag{8}$$

As in Section 2, we assume to have a discretization (e.g. by finite elements) for the problem on a level J resulting in the system of linear equations

$$(\boldsymbol{A}_J \otimes \boldsymbol{A}_J) \boldsymbol{U}_J = \boldsymbol{F}_J\,. \tag{9}$$

Here, $\boldsymbol{A}_J \in \mathbb{R}^{N_J \times N_J}$ is the system matrix from (2). The operator $\otimes$ is the Kronecker product operator for matrices. For matrices $\boldsymbol{S} \in \mathbb{R}^{n_1 \times n_2}$, $\boldsymbol{T} \in \mathbb{R}^{m_1 \times m_2}$, it computes the Kronecker product

$$\boldsymbol{S} \otimes \boldsymbol{T} := \begin{pmatrix} s_{11}\boldsymbol{T} & \ldots & s_{1n_2}\boldsymbol{T} \\ \vdots & \ddots & \vdots \\ s_{n_1 1}\boldsymbol{T} & \ldots & s_{n_1 n_2}\boldsymbol{T} \end{pmatrix}.$$

Consequently, $\boldsymbol{A}_J \otimes \boldsymbol{A}_J$ becomes a matrix of size ${N_J}^2 \times {N_J}^2$. Moreover, $\boldsymbol{U}_J, \boldsymbol{F}_J \in \mathbb{R}^{N_J \cdot N_J}$ are the solution and the right-hand side, respectively.

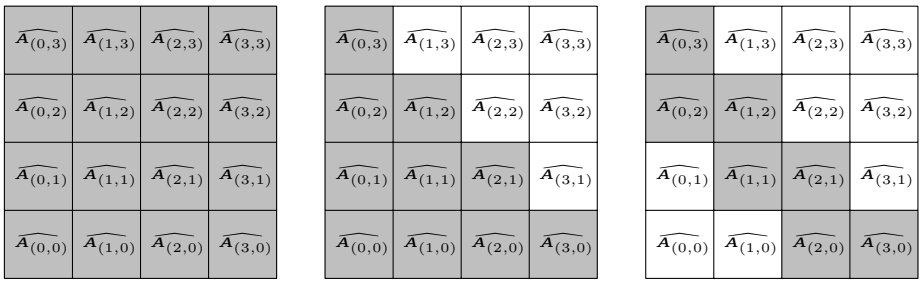

Fig. 1 For discretization level $J = 3$, multilevel frames on the full tensor product space require a very densely populated system matrix $\widehat{\boldsymbol{A}_J}$ *(left)*, while sparse approximation leads to the system matrix $\widehat{\boldsymbol{A}_J}$ *(center)* with smaller size due to fewer active (i.e. gray) matrix subblocks. The sparse grid combination technique *(right)* leads to the most efficient approximation.

By assuming an underlying d-dimensional finite element discretization with mesh width h and a multigrid-type linear solver, solving the linear system in (9) would require at least $O\left(h^{-2d}\right)$ operations, in contrast to $O\left(h^{-d}\right)$ for the problem given by (2). This amount of computational work is prohibitively large, especially for larger d. Therefore, we shall find a way to reduce the amount of work to solve this problem. Before we do that, we change the problem discretization to a multilevel discretization, which is the basis for the subsequent sparse approaches.

3.1 Multilevel frames for tensor product constructions

To extend the solution approach from Section 2.2 to tensor product problems, we first recall that we had in the *univariate* multilevel frame case matrix blocks of the form

$$\boldsymbol{A}_{\boldsymbol{j}} = \boldsymbol{A}_{j_1 j_2} := \boldsymbol{P}_{j_1}^{J} \boldsymbol{A}_J \boldsymbol{P}_J^{j_2} \,,$$

with $\boldsymbol{P}_J^j$, $\boldsymbol{P}_j^J$ as defined in (5) by applying coarsening and the transfer operators of algebraic multigrid. In the univariate case, the multilevel frame linear system of equations was

$$\boldsymbol{A}_{\boldsymbol{J}} \boldsymbol{u}_{\boldsymbol{J}} = \boldsymbol{f}_{\boldsymbol{J}} \,,$$

$$\boldsymbol{A}_{\boldsymbol{J}} = [\boldsymbol{A}_{\boldsymbol{j}}]_{\|\boldsymbol{j}\|_{\ell^\infty} \leq J}, \quad \boldsymbol{u}_{\boldsymbol{J}} = [\boldsymbol{u}_j]_{|j| \leq J}, \quad \boldsymbol{f}_{\boldsymbol{J}} = [\boldsymbol{f}_j]_{|j| \leq J}.$$

By tensorizing this problem, we naturally get the tensor-product frame linear system of equations

$$\widehat{\boldsymbol{A}_{\boldsymbol{J}}} \boldsymbol{U}_{\boldsymbol{J}} = \boldsymbol{F}_{\boldsymbol{J}} \,, \tag{10}$$

with

$$\widehat{\boldsymbol{A}_{\boldsymbol{J}}} = [\boldsymbol{A}_{j_1 j_2} \otimes \boldsymbol{A}_{j_1' j_2'}]_{\|(j_1, j_1')\|_{\ell^\infty}, \|(j_2, j_2')\|_{\ell^\infty} \leq J} \,,$$

and

$$\boldsymbol{U}_{\boldsymbol{J}} = [\boldsymbol{U}_{\boldsymbol{j}}]_{\|\boldsymbol{j}\|_{\ell^\infty} \leq J}, \quad \boldsymbol{F}_{\boldsymbol{J}} = [\boldsymbol{F}_{\boldsymbol{j}}]_{\|\boldsymbol{j}\|_{\ell^\infty} \leq J} \,.$$

For a given right-hand side $\boldsymbol{F}_J$, we can construct the corresponding blocks $\boldsymbol{F}_{\boldsymbol{j}}$ by

$$\boldsymbol{F}_{\boldsymbol{j}} = \boldsymbol{F}_{j_1 j_2} := \left(\boldsymbol{P}_J^{j_1} \otimes \boldsymbol{P}_J^{j_2} \right) \boldsymbol{F}_J \,.$$

The corresponding vectors and matrices are (using $\boldsymbol{j} = (j_1, j_2)$) of the dimensionalities

$$\boldsymbol{A_j} \otimes \boldsymbol{A_{j'}} \in \mathbb{R}^{N_{j_1} N_{j_1'} \times N_{j_2} N_{j_2'}} \quad \text{and} \quad \boldsymbol{U_j}, \boldsymbol{F_j} \in \mathbb{R}^{N_{j_1} N_{j_2}} .$$

In order to characterize the computational complexity for the solution of (10), we recall that we assume to have a constant operator complexity for the sequence of matrices $\boldsymbol{A}_{jj} = \boldsymbol{A}_j$, i.e. $\sum \eta(\boldsymbol{A}_j) \leq c\,\eta(\boldsymbol{A}_J)$. Moreover, by definition of the Kronecker product, we have the number of non-zeros in each block of $\widehat{\boldsymbol{A_J}}$ given by

$$\eta(\boldsymbol{A_j} \otimes \boldsymbol{A_{j'}}) = \eta(\boldsymbol{A_j})\eta(\boldsymbol{A_{j'}}) ,$$

from which it is easy to verify that we have

$$\eta(\widehat{\boldsymbol{A_J}}) = \eta(\boldsymbol{A_J})\eta(\boldsymbol{A_J}) ,$$

with $\boldsymbol{A_J}$ from Section 2.2. It remains to find an upper bound to the number of non-zeros of the univariate multilevel frame system matrix. Here, we compute

$$\begin{aligned}\eta(\boldsymbol{A_J}) = \eta\left([A_{\boldsymbol{j}}]_{\|\boldsymbol{j}\|_{\ell^\infty} \leq J}\right) = \sum_{j_1=1}^{J}\sum_{j_2=1}^{J} \eta(\boldsymbol{A}_{j_1 j_2}) &\leq \sum_{j_1=1}^{J}\sum_{j_2=1}^{J} \eta(\boldsymbol{A}_{\max(j_1,j_2),\max(j_1,j_2)}) \\ &= J\sum_{j} \eta(\boldsymbol{A}_{jj}) \leq C_{\boldsymbol{A}} J \eta(\boldsymbol{A}_J) .\end{aligned}$$

In the last equality, we used that we have $\eta(\boldsymbol{A}_{j_1 j_2}) = \eta(\boldsymbol{A}_{j_2 j_1})$. The last inequality corresponds to our assumption on the operator complexity. Since we have $J \sim O(|\log h|)$, we finally get

$$\eta(\widehat{\boldsymbol{A_J}}) \leq c\, {C_{\boldsymbol{A}}}^2 |\log h|^2 \eta(\boldsymbol{A}_J)^2 .$$

This means that the computational work to solve (10) is asymptotically identical to a solve of (9), up to a logarithmic term. Moreover, by using recursive techniques known from the BPX-preconditioner [2], we could even avoid the logarithmic term.

Figure 1 displays the matrix blocks $\widehat{\boldsymbol{A_{(j,j')}}} := \boldsymbol{A}_{\max(j_1,j_2),\max(j_1',j_2')}$ that are used by the tensor product multi-level frame system. We limit ourselves to this subset of matrices for the ease of visualization. However, following [16], we in fact only need these matrices to construct $\widehat{\boldsymbol{A}}_J$, if appropriate prolongation and restriction operators are considered.

3.2 Sparse tensor product construction

Solving (9) or (10) would be prohibitively expensive, cf. Figure 1. As in the geometric multilevel case, we now assume that the solution of the elliptic problem (1) on Ω is H^s regular. Therefore, the solution of the tensor product problem (8) becomes H^s_{mix}-regular, see [20]. This allows to follow, for example, the lines of [16] to introduce a sparse, however now algebraically constructed, version of the discretized problem. Instead of using all sub-problems for multi-indices $\|\boldsymbol{j}\|_{\ell^\infty} \leq J$, the sparse approximation is reduced to multi-indices $\|\boldsymbol{j}\|_{\ell^1} \leq J$. Thereby, we obtain a new system of linear equations

$$\widetilde{\boldsymbol{A_J}}\widetilde{\boldsymbol{U_J}} = \widetilde{\boldsymbol{F_J}}$$

with

$$\widetilde{\boldsymbol{A_J}} := [\boldsymbol{A}_{j_1 j_2} \otimes \boldsymbol{A}_{j_1' j_2'}]_{\|(j_1,j_1')\|_{\ell^1},\|(j_2,j_2')\|_{\ell^1} \le J},$$
$$\widetilde{\boldsymbol{U_J}} = [\boldsymbol{U_j}]_{\|\boldsymbol{j}\|_{\ell^1} \le J}, \quad \widetilde{\boldsymbol{F_J}} = [\boldsymbol{F_j}]_{\|\boldsymbol{j}\|_{\ell^1} \le J}.$$

Figure 1 compares both choices in the plots on the left-hand side and the center, recalling that we use only matrices $\widehat{\boldsymbol{A_{(j,j')}}} := \boldsymbol{A}_{\max(j_1,j_2),\max(j_1',j_2')}$ in this figure, see last section. It is easy to see, that this choice should be much more efficient.

To show that it is actually more efficient, we now discuss the number of non-zeros in $\widetilde{\boldsymbol{A}}_J$. Similar to the extimate of the number of non-zeros in the *univariate* multi-level system matrix, we now compute

$$\begin{aligned}
\eta(\widetilde{\boldsymbol{A}}_J) &= \sum_{0\le j_1+j_1'\le J} \sum_{0\le j_2+j_2'\le J} \eta(\boldsymbol{A}_{j_1 j_2} \otimes \boldsymbol{A}_{j_1' j_2'}) \\
&= \sum_{j_1=0}^{J} \sum_{j_1'=0}^{J-j_1} \sum_{j_2=0}^{J} \sum_{j_2'=0}^{J-j_2} \eta(\boldsymbol{A}_{j_1 j_2}) \eta(\boldsymbol{A}_{j_1' j_2'}) \\
&\le \sum_{j_1=0}^{J} \sum_{j_1'=0}^{J-j_1} \sum_{j_2=0}^{J} \sum_{j_2'=0}^{J-j_2} \eta(\boldsymbol{A}_{\max(j_1,j_2),\max(j_1,j_2)}) \eta(\boldsymbol{A}_{\max(j_1',j_2'),\max(j_1',j_2')}) \\
&= J^2 \sum_{j=0}^{J} \sum_{j'=0}^{J-j} \eta(\boldsymbol{A}_{jj}) \eta(\boldsymbol{A}_{j'j'})
\end{aligned}$$

As discussed before, there is not much theory on the size of the levels in the algebraic multilevel construction. The only available information is the assumed bound on the operator complexity. However, this does not give enough information to finish the above estimate. Nevertheless, the bound on the operator complexity implies a similar scaling of the non-zeros with level j as in the geometric multilevel construction. Therefore, we here assume to have the same number of non-zeros for each matrix $\boldsymbol{A}_{jj}$ as in the geometric construction, to give a hint towards the possible performance improvement by the algebraic sparse construction.

With this in mind, we follow the previous example of (linear) finite elements on a mesh with mesh width h. The number of non-zero entries for matrix $\boldsymbol{A}_j$ is proportional to the number of elements and therefore

$$\eta(\boldsymbol{A}_{jj}) = O(2^{d\,j}).$$

By extending the above estimate, we get

$$\begin{aligned}
\eta(\widetilde{\boldsymbol{A}}_J) = J^2 \sum_{j=0}^{J} \sum_{j'=0}^{J-j} \eta(\boldsymbol{A}_{jj}) \eta(\boldsymbol{A}_{j'j'}) &= c\,J^2 \sum_{j=0}^{J} \sum_{j'=0}^{J-j} 2^{d\,j} 2^{d\,j'} \\
= c\,J^2 \sum_{j=0}^{J} \sum_{k=j}^{J} 2^{d\,j} 2^{d\,(k-j)} = c\,J^2 \sum_{j=0}^{J} \sum_{k=j}^{J} 2^{d\,k} &= O\left(J^3 2^{d\,J}\right)
\end{aligned}$$

Moreover, we have $J = O(|\log h|)$. That is, the number of non-zeros in the system matrix in $\widetilde{\boldsymbol{A_J}}$ is asymptotically

$$\eta\left(\widetilde{\boldsymbol{A_J}}\right) = O\left(|\log h|^3\, h^{-d}\right).$$

That is, in case a BPX-type preconditioner [2,5,7,18] is used, the computational complexity of the problem on the tensor product domain $\Omega\times\Omega$ is (up to a logarithmic factor) reduced to the computational complexity of the problem on domain Ω. Moreover, by applying an optimal approach for the construction of the subproblem matrices $\boldsymbol{A_j}\otimes\boldsymbol{A_{j'}}$ [1,3,26], the remaining logarithmic factors might eben be dropped.

3.3 Sparse grid combination technique

It has been shown in [15] that the previous sparse approximation is equivalent to the so-called sparse grid combination technique. The latter one starts approximating tensor product problems from a sequence of finite dimensional function spaces

$$V_0^{(i)} \subset V_1^{(i)} \subset \ldots \subset V_J^{(i)} \subset \ldots \subset V^{(i)}$$

of increasing accuracy, where i indicates the domain to which the function space is associated. Since we operate on $\Omega\times\Omega$, we have $i=1,2$. As next step, hierarchical increment spaces $W_j^{(i)}$ are considered such that

$$V_j^{(i)} := W_j^{(i)} \oplus V_{j-1}^{(i)}\,,$$

where $W_0^{(i)} := V_0^{(i)}$. As usual in sparse (grid) approximation, the (two-dimensional) sparse approximation space $\widehat{V_J}$ is then, cf. [9], defined as

$$\begin{aligned}\widehat{V_J} &:= \bigoplus_{j'=0}^{J} W_{J-j'}^{(1)}\otimes V_{j'}^{(2)} = \bigoplus_{j'=0}^{J}\left(V_{J-j'}^{(1)}\ominus V_{J-1-j'}^{(1)}\right)\otimes V_{j'}^{(2)}\\ &= \bigoplus_{j'=0}^{J}\left[\left(V_{J-j'}^{(1)}\otimes V_{j'}^{(2)}\right)\ominus\left(V_{J-1-j'}^{(1)}\otimes V_{j'}^{(2)}\right)\right]\,. \end{aligned} \tag{11}$$

The *combination technique* computes (anisotropic) full-grid solutions on the subspaces involved in equation (11) and combines them using appropriate projection. Translated to our problem setting, this approximation is given as

$$\begin{aligned}\widehat{\boldsymbol{U_J}} &= \sum_{j'=0}^{J}\left[\left(\boldsymbol{P}_{J-j'}^{J}\otimes\boldsymbol{P}_{j'}^{J}\right)\boldsymbol{U}_{J-j',j'} - \left(\boldsymbol{P}_{J-1-j'}^{J}\otimes\boldsymbol{P}_{j'}^{J}\right)\boldsymbol{U}_{J-1-j',j'}\right]\\ &= \sum_{\|\boldsymbol{j}\|_{\ell^1}=J}(\boldsymbol{P}_j^J\otimes\boldsymbol{P}_{j'}^J)\boldsymbol{U_j} \;-\; \sum_{\|\boldsymbol{j}\|_{\ell^1}=J-1}(\boldsymbol{P}_j^J\otimes\boldsymbol{P}_{j'}^J)\boldsymbol{U_j}\,. \end{aligned} \tag{12}$$

To compute it, we have to solve the decoupled problems

$$\widehat{\boldsymbol{A_j}}\boldsymbol{U_j} = (\boldsymbol{A}_{j_1j_1}\otimes\boldsymbol{A}_{j_2j_2})\boldsymbol{U_j} = \boldsymbol{F_j},\quad \text{where } \|\boldsymbol{j}\|_{\ell^1}\in\{J,J-1\}\,. \tag{13}$$

On the right-hand side of Figure 1, the sub-matrices $\widehat{\boldsymbol{A_j}}$ used in this approximation have been marked gray. As before, one can easily verify that the total number of non-zeros of the matrices in (13) is asymptotically $O\left(|\log h|\,h^{-d}\right)$ for the case of linear finite elements on a tetrahedral mesh with mesh width h in

d dimensions and a geometrically constructed multilevel structure. However, Figure (1) easily clarifies that the pre-asymptotic number of non-zeros in the matrices involved in the combination technique is much smaller than the non-zeros in the sparse approximation discussed before.

In terms of computational complexity of the combination technique, let us remind that the (approximate) solution of each sub-problem in (13) can be realized by an iterative linear solver with matrix-vector products. To be more specific, tensor product versions of standard iterative solvers can be constructed, by reshaping a given iterate $\boldsymbol{U}_{\boldsymbol{j}=(j,j')} \in \mathbb{R}^{N_j \cdot N'_j}$ (and the appropriate right-hand side) to a matrix of size $N_j \times N_{j'}$. Then, the action of one step of an iterative solver for matrix $\widehat{\boldsymbol{A}_{\boldsymbol{j}}} = \boldsymbol{A}_j \otimes \boldsymbol{A}_{j'}$ is done by first applying the iterative solver step for $\boldsymbol{A}_j$ to all $N_{j'}$ columns of the reshaped matrix and by second applying the iterative solver step for $\boldsymbol{A}_{j'}$ to all N_j rows of the reshaped matrix. One easily verifies that the Kronecker product of two matrices $\boldsymbol{A}_j$, $\boldsymbol{A}_{j'}$ with $O(N_j)$, $O(N_{j'})$ non-zeros has $O(N_j N_{j'})$ non-zeros. This leads to a computational complexity of $O(N_j N_{j'})$ for a single matrix-vector product.

Next, we observe that we actually need only a problem-size independent constant number of iterations, if we choose an appropriate solver. Since we have all prolongation and restriction operators from AMG at our disposal, we can actually build a tensor product version of algebraic multigrid. The construction of a tensor-product AMG follows the idea outlined above, i.e. we apply univariate versions of AMG to the columns and rows of a reshaped iterate $\boldsymbol{U}_{\boldsymbol{j}=(j,j')}$ of size $N_j \times N_{j'}$. The tensor-product AMG gives us the property of problem-size independent convergence for each sub-problem in (13), i.e. we need $O(N_j N_{j'})$ operations for each sub-problem.

While we have no theory on the number of unknowns on each level of our algebraically constructed combination technique, we can still give an analogy from the geometric setting, in order to predict the overall complexity of the method. In case our algebraical construction would behave exactly as a geometrically constructed multilevel hierarchy, we would have the relation $N_j = O(2^{dj})$. Thereby, the solution of each sub-problem would require $O(2^{d(j+j')})$ operations. Since it holds $\|\boldsymbol{j}\|_{\ell^1} \in \{J, J-1\}$, we can compute

$$\begin{aligned}\sum_{\|\boldsymbol{j}\|_{\ell^1} \in \{J,J-1\}} 2^{d(j+j')} &= \sum_{\|\boldsymbol{j}\|_{\ell^1}=J} 2^{d(j+j')} + \sum_{\|\boldsymbol{j}\|_{\ell^1}=J-1} 2^{d(j+j')} \\ &= (J+1)2^{dJ} + J2^{d(J-1)}\,.\end{aligned}$$

Hence, we would finally end up with a computational complexity of $O(J2^{dJ})$ or $O(N_J \log N_J)$.

4 Implementation

In our numerical results, we approximate solutions for tensor product finite element discretizations of elliptic problems based on the combination technique with $\Omega \subset \mathbb{R}^{2,3}$. To this end, we assemble system matrices for a given problem, construct the multilevel hierarchies, solve the decoupled, anisotropic problems in (13) and combine the solutions following the combination rule (12).

Assembly of system matrices. The discretization by the finite element method is done with the *Matlab PDE Toolbox* of *Matlab 2017a*. We use linear finite elements and construct meshes with maximum element size `Hmax`$= 2^{-J}$. Furthermore, we use the option `Jiggle` to optimize the mesh in quality. The stiffness matrix (incorporating boundary conditions) is constructed by using the *Matlab* command `assembleFEMatrices` with option `nullspace`. In a similar way, we extract the mass matrix. Afterwards, both matrices and the mesh node coordinates are stored to files.

Construction of the multilevel hierarchy. From within *Matlab* we call an in-house ad-hoc code that uses the parallel linear solver library *hypre* [6] in version 2.11.1. This library contains the implementation *BoomerAMG* of classical Ruge-Stüben AMG. The code reads the stiffness matrix from file and creates the AMG multilevel hierarchy by using *hypre*. In addition to *standard coarsening* with strength measure $\epsilon_{str} = 0.25$ and *standard interpolation*, we use two passes of *Jacobi* interpolation with a truncation of the Jacobi interpolation with a threshold of 0.001 for the two-dimensional problems and 0.01 for the three-dimensional problem (being treated in Section 5). All other parameters are kept as the defaults of *BoomerAMG*. After having created the multigrid hierarchy, the program stores the prolongation matrices of all created levels to files. These are read by *Matlab*.

Solution of the anisotropic tensor product problems. Based on the prolongation matrices and the system matrix $\boldsymbol{A}_J$ on the finest levels, the decoupled problems in (13) can be set up. As discussed before, a tensor product version of AMG is used to solve the systems of linear equations. In our implementation, we construct the sub-problem operators in (13) by individually multiplying the transfer operators between two consecutive levels.

Our tensor product AMG is iterated until the convergence criterion

$$\|\boldsymbol{R}_{\boldsymbol{j}}^{it}\|_{\ell^2} / \|\boldsymbol{F}_{\boldsymbol{j}}\|_{\ell^2} \leq \epsilon_{tol}$$

is fulfilled, where $\boldsymbol{R}_{\boldsymbol{j}}^{it}$ is the residual of the current iterate $\boldsymbol{U}_{\boldsymbol{j}}^{it}$ in the solver. Since the problems in (13) completely decouple, we can easily parallelize their solution process by a `parfor` loop in *Matlab*. In case an individual problem becomes very expensive, we further implemented a distributed memory parallelization for the tensor product AMG based on *Matlab*'s `distributed` function. Thereby, we overcome the limitation of a non-existing multi-core parallelization for sparse matrix-vector products in *Matlab*.

Combination of the solutions. In the combination phase, we avoid to prolongate the full partial solutions to the finest level J. Instead, we randomly chose N_{eval} nodes on the product of the finest meshes on $\Omega \times \Omega$. On these points, we evaluate the combination formula (12) and compute the empirical error measure

$$e(\boldsymbol{U}_{approx}) = \|\boldsymbol{U}_{approx} - \boldsymbol{U}_{ref}\|_{\ell^2} / \|\boldsymbol{U}_{ref}\|_{\ell^2},$$

where $\boldsymbol{U}_{approx}$ is the approximated solution and $\boldsymbol{U}_{ref}$ is an appropriately evaluated reference solution. Note that we do not multiply the tensor product of the prolongation with the solution. Instead, we follow the ideas from Section 3.3 for the construction of the tensor product AMG and apply the prolongations directionwise. The prolongation for each sub-problem is also parallelized by a `parfor` loop.

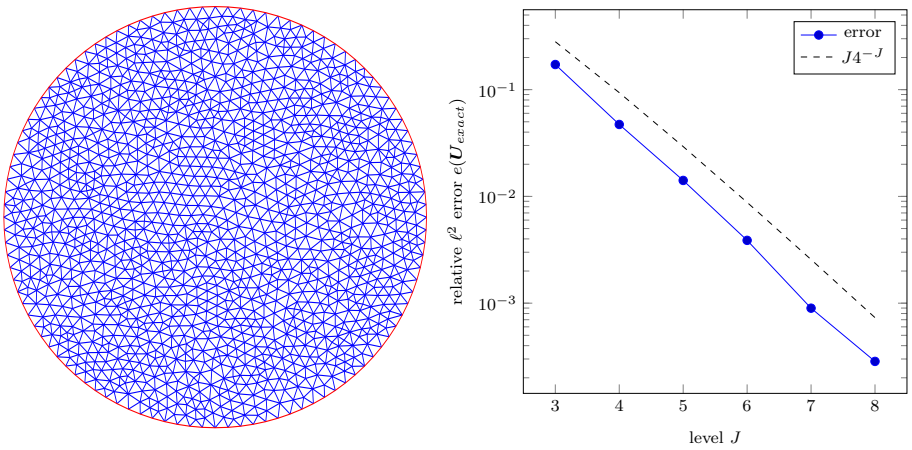

Fig. 2 The combination technique based on our algebraic multilevel hierarchy and applied to the tensor product of a disk geometry with an unstructured mesh (*left, triangulated with* $J = 5$) shows the same convergence as the geometrically constructed combination technique *(right)*.

5 Numerical results

In our empirical studies, we consider the numerical solution of the problem

$$\begin{aligned} (\Delta \otimes \Delta)u &= f \quad \text{on } \Omega \times \Omega\,, \\ u &= 0 \quad \text{on } \partial(\Omega \times \Omega)\,. \end{aligned} \tag{14}$$

by means of the combination technique based on the algebraic multilevel hierarchy. Different choices will be made for the domain Ω and the right-hand side f.

5.1 Analytic example on a disk

The first study is done on a disk domain Ω with center $(0,0)^\top$ and radius 0.5. We set

$$f(\boldsymbol{x},\boldsymbol{y}) = 1\,.$$

The exact solution of the resulting problem is

$$u(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{16}\left(x_1^2 + x_2^2 - 0.5^2\right)\left(y_1^2 + y_2^2 - 0.5^2\right)\,.$$

To approximate the solution u by the combination technique, we follow the methodology discussed in Section 4. As part of this, we triangulate the geometry with a maximum element width of 2^{-J}. Figure 2 shows on the left-hand side the resulting mesh for $J = 5$. It is obvious that the resulting mesh is unstructured. Therefore, classical geometric constructions for the sparse grid combination technique would not be feasible on that mesh. In contrast, our new algebraic approach can solve this problem.

This is shown on the right-hand side of Figure 2, where we compare the numerically approximated solution against the above exact solution. Convergence

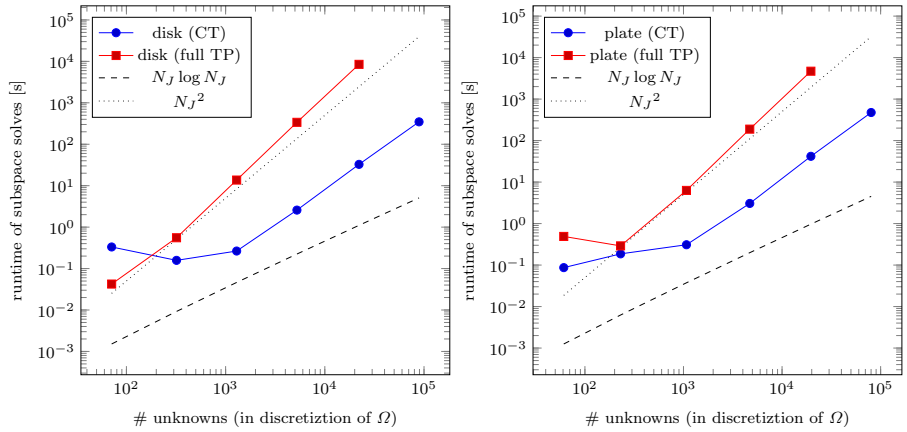

Fig. 3 We compare the runtime of the new combination technique approach (CT) with the runtime of the traditional the full tensor-product approach (full TP) for the solution of the tensor product elliptic problems on the disk geometry *(left)* and the plate geometry *(right)*.

		# dofs on algebraically coarsened level j									
Ω	$J \setminus j$	0	1	2	3	4	5	6	7	8	9
disk	3	3	11	28	71						
	4	8	21	52	119	320					
	5	12	31	84	207	495	1292				
	6	20	51	139	348	852	2009	5234			
	7	35	93	244	606	1473	3510	8415	22118		
	8	46	130	366	978	2469	5983	14480	34081	89097	
	$O(2^{dj})$	1	5	22	87	348	1392	5569	22274	89097	
plate	3	5	20	61							
	4	16	36	90	230						
	5	28	68	168	414	1072					
	6	46	116	297	745	1813	4703				
	7	63	184	515	1272	3117	7491	19611			
	8	103	302	815	2124	5301	12822	30639	80146		
spanner	3	4	10	19	50	117	247				
	4	11	22	59	147	326	689	1454			
	5	40	114	300	689	1516	3216	6484	13939		
	6	210	548	1364	3123	6708	14109	29103	57438	125223	
	7	1386	3120	6627	14016	29533	61150	124921	253291	496614	1082581

Table 1 For a given problem on level J, the algebraic multilevel construction on our example domains Ω constructs coarser levels with a decrease of the number of unknowns roughly similar to geometric multilevel constructions e.g. in the *disk* test case. Above, only those levels j are reported that are used in the convergence study.

results for the choices $J = 3, \ldots, 8$ are given. From literature, compare e.g. [15], we know that the error of the geometrically constructed sparse grid combination technique scales for the problem under consideration like $J4^{-J}$. As we can see from the convergence results in Figure 2, the algebraically constructed combination technique shows the same convergence behavior, while being applicable to unstructured grids.

Figure 3 shows on the left-hand side computing times for growing problem size N_J of the univariate discretization of Ω. We compare the time required for the solution of the combination technique sub-problems with the time required to solve the full tensor-product problem (9) by our tensor-product AMG implementation. Note that we use the coarse grid hierarchies reported in Table 1 for both the

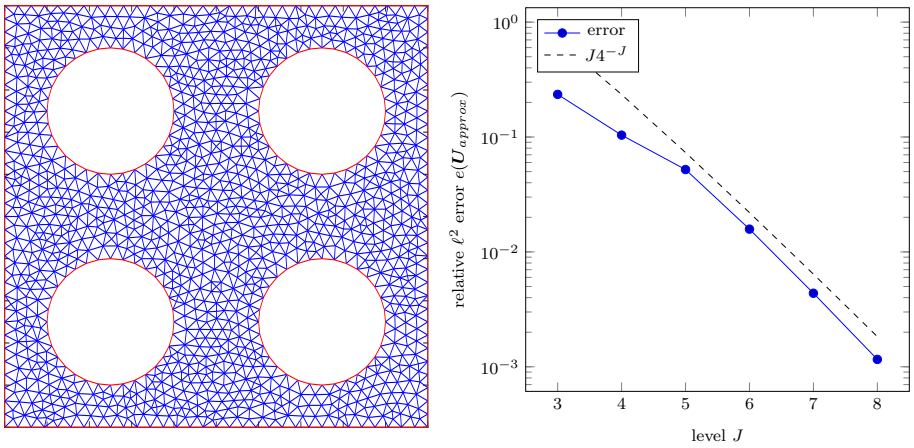

Fig. 4 Even for a covariance load on a complex geometry (*left*, triangulated for $J = 5$), the algebraic construction shows the appropriate convergence rate after a short pre-asymptotic phase *(right)*.

combination technique and the full tensor-product approach. All measurements were done on a compute server with dual 20-core Intel Xeon E5-2698 v4 CPU at 2.2 GHz and 768 GB RAM. It becomes evident that our algebraically constructed combination technique approach beats the full tensor-product approach in both, computational complexity and effective runtime. However, both results do not show the predicted computational complexity of $O(N_J \log N_J)$ and $O({N_J}^2)$. There are several reasons for this behavior.

- First, algebraic multigrid often shows a small, roughly logarithmic, growth in the number of iterations for larger problem sizes, resulting in a slow-down by a logarithmic factor.
- Second, we observe a certain fill-in in the system matrices for coarser problems in the algebraic construction due to our choice of an additional (truncated) Jacobi interpolation. However, this should be pre-asymptotic behavior.
- Third, as can be seen in Table 1, the AMG coarsening approach chosen in our implementation does not show the exact same (asymptotic) decay rate $O(2^{dj})$ in the number of levels as we expect it from the geometric construction. In fact, this leads to a problem-size dependent growth of the coarsest grid. While this growth does not affect the error decay, it shows up in the computational complexity.

Meanwhile, as stated before, we are able to beat the solution approach based on the full tensor-product approach in terms of computational complexity. Even more, if we would use AMG as solver for the anisotropic sub-problems in the *geometric* construction, we would see similar results, anyway. Finally, in terms of runtime, we are by more than two orders of magnitude faster.

5.2 Example on complex geometry with covariance load

The next numerical study is concerned with the solution of the problem (14) with the load

$$f(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(\frac{-\|\boldsymbol{x} - \boldsymbol{y}\|^2}{\ell}\right)$$

that corresponds to an (unscaled) Gaussian covariance kernel with correlation length ℓ. This is a prototype version of the tensor product elliptic problem on $\Omega \times \Omega$ showing up in the computation of the output covariance of an elliptic problem on Ω with random input, cf. [16].

In addition to the more complicated right-hand side, we solve the problem for a rather complex geometry Ω. We choose the geomery of a square plate on $[0,1]^2$ with circular wholes of radius 0.15 which are centered at the points

$$\{(0.25, 0.25), (0.25, 0.75), (0.75, 0.25), (0.75, 0.75)\}.$$

Figure 4 shows its triangulation for $J = 5$ on the left-hand side. Note that it would be almost impossible to solve a problem on such a geometry with the geometrical construction for the sparse grid combination technique. However, with the algebraic construction, a coarsening to very few degrees of freedom becomes easily possible, compare Table 1.

To be able to compare the above problem against a numerically computed reference solution, we replace the (sampled) covariance kernel for $\ell = 1$ by its low-rank approximation computed with the pivoted Cholesky factorization [14], truncated for a trace norm of 10^{-8}. In this case, depending on the problem size, the truncation results in roughly twenty low-rank terms.

On the right-hand side of Figure 4, we show the convergence results with errors computed against the numerically approximated exact solution by use of the low-rank approximation. After a pre-asymptotic phase, we are able to attain an error that scales like $J4^{-J}$ as in the geometric construction.

The problem size dependent runtime to compute the subspace solutions for the plate geometry is given in Figure 3 on the right-hand side. We observe similar compuational complexities and similar runtimes as in the previous example on the disk.

5.3 Large-scale real-world example

Our last numerical study treats a large-scale problem with a complex real-world geometry Ω. We again aim at solving (14) for $f(\boldsymbol{x}, \boldsymbol{y}) = 1$. However, we choose the *three-dimensional* spanner geometry found in Figure 5. In contrast to the previous examples, we set the maximum mesh width to 2^{5-J}, since the geometry is contained in the rather large bounding box $[-5, 5] \times [-12.2, 112] \times [-15.7, 15.7]$. Note that the triangulation of Ω results for level $J = 7$ in a discretization with $1,082,581$ unknowns. That is, if we would want to solve the full tensor product problem on $\Omega \times \Omega$, cf. (8), then we would have to solve a problem with about 10^{12}, that is a *trilion*, unknowns. This would be clearly out of scope even for large parallel clusters. In contrast, the combination technique allows to solve this problem. Nevertheless, we still have to solve, e.g. for level $J = 7$ and the system matrix $\widehat{\boldsymbol{A}_{(0,J)}}$ a problem with $1,082,581 \times 1,386$ unknowns, compare Table 1.

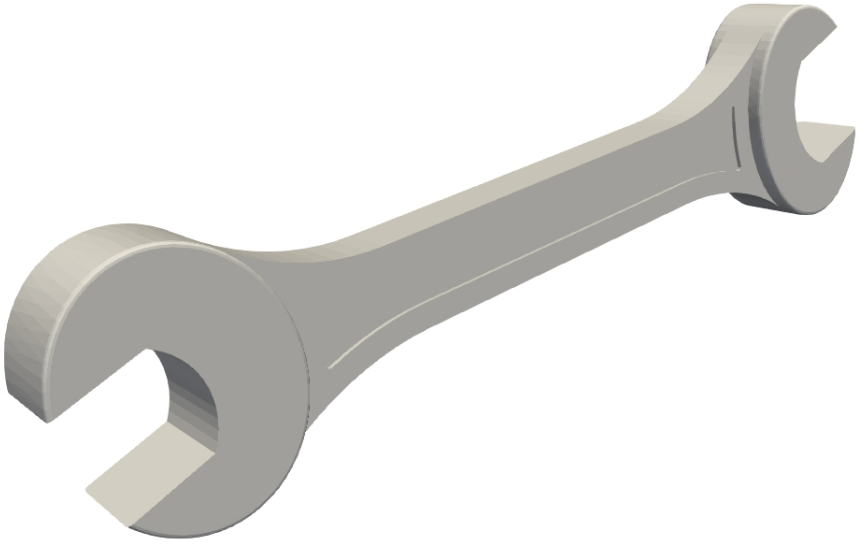

Fig. 5 In our large-scale real world example, we solve an elliptic problem on the tensor product of the three-dimensional geometry of a spanner. For a discretization level of $J = 7$, the discretization of Ω has more than a million unknowns. This would lead to 10^{12}, that is a *trillion*, unknowns in the full tensor product discretization.

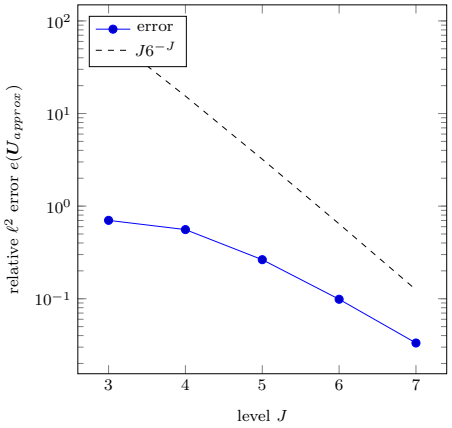

Fig. 6 Our algebraic multilevel construction for the sparse grid combination technique on the large-scale three-dimensional spanner geometry gradually approaches the optimal convergence rate of $J2^{-dJ}$.

In Figure 6, we show the convergence results for this large scale problem relative to a numerical approximation of the solution. Due to the high dimensionality and complexity of the domain Ω, the convergence results in Figure 6 are only gradually approaching the optimal scaling of $J2^{-dJ}$. Nevertheless, we are able to solve this problem up to a certain accuracy. This shows that even very complex problems of large scale can be solved by the proposed approach.

6 Conclusions

In this work, we have introduced an algebraic construction method for the sparse approximation of tensor product elliptic problems by means of the combination technique. While previous approaches were tight to geometric hierarchies of mesh refinements to build the underlying multilevel discretization, we were able to solve the given type of problems on complex geometries and for unstructured grids by an algebraic multilevel hierarchy based on AMG. We could show that our approach has the same convergence rates as the geometric construction. Measurements of the computational complexity were in the linear range with poly-logarithmic factors. Overall, we are now able to apply sparse approximation for elliptic tensor product problems in a black-box fashion.

References

1. Balder, R., Zenger, C.: The solution of multidimensional real Helmholtz equations on sparse grids. SIAM Journal on Scientific Computing **17**(3), 631–646 (1996). DOI 10.1137/S1064827593247035. URL https://doi.org/10.1137/S1064827593247035
2. Bramble, J., Pasciak, J., Xu, J.: Parallel multilevel preconditioners. Mathematics of Computation **55**, 1–22 (1990)
3. Bungartz, H.J.: A multigrid algorithm for higher order finite elements on sparse grids. ETNA. Electronic Transactions on Numerical Analysis **6**, 63–77 (1997). URL http://eudml.org/doc/119649
4. Bungartz, H.J., Griebel, M.: Sparse grids. Acta Numerica **13**, 1–123 (2004)
5. Dahmen, W.: Wavelet and multiscale methods for operator equations. Acta Numerica **6**, 55–228 (1997)
6. Falgout, R.D., Yang, U.M.: hypre: A library of high performance preconditioners. In: P.M.A. Sloot, A.G. Hoekstra, C.J.K. Tan, J.J. Dongarra (eds.) Computational Science — ICCS 2002, pp. 632–641. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
7. Griebel, M.: Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen. Teubner Skripten zur Numerik. B.G. Teubner, Stuttgart (1993)
8. Griebel, M.: Multilevel algorithms considered as iterative methods on semidefinite systems. SIAM International Journal Scientific Statistical Computing **15(3)**, 547–565 (1994)
9. Griebel, M., Harbrecht, H.: On the construction of sparse tensor product spaces. Mathematics of Computation **82**(282), 975–994 (2013). DOI 10.1090/S0025-5718-2012-02638-X
10. Griebel, M., Harbrecht, H.: On the convergence of the combination technique. In: J. Garcke, D. Pflüger (eds.) Sparse grids and Applications – Stuttgart 2014, *Lecture Notes in Computational Science and Engineering*, vol. 97, pp. 55–74. Springer (2014)
11. Griebel, M., Oswald, P.: Greedy and randomized versions of the multiplicative Schwarz method. Linear Algebra and its Applications **7**, 1596–1610 (2012)
12. Griebel, M., Schneider, M., Zenger, C.: A combination technique for the solution of sparse grid problems. In: P. de Groen, R. Beauwens (eds.) Iterative Methods in Linear Algebra, pp. 263–281. IMACS, Elsevier, North Holland (1992)
13. Harbrecht, H.: A finite element method for elliptic problems with stochastic input data. Applied Numerical Mathematics **60**(3), 227–244 (2010). DOI 10.1016/j.apnum.2009.12.002. URL http://dx.doi.org/10.1016/j.apnum.2009.12.002
14. Harbrecht, H., Peters, M., Schneider, R.: On the low-rank approximation by the pivoted Cholesky decomposition. Applied Numerical Mathematics **62**(4), 428–440 (2012). DOI http://dx.doi.org/10.1016/j.apnum.2011.10.001. URL http://www.sciencedirect.com/science/article/pii/S0168927411001814
15. Harbrecht, H., Peters, M., Siebenmorgen, M.: Combination technique based k-th moment analysis of elliptic problems with random diffusion. Journal of Computational Physics **252**(C), 128–141 (2013). DOI 10.1016/j.jcp.2013.06.013. URL http://dx.doi.org/10.1016/j.jcp.2013.06.013
16. Harbrecht, H., Schneider, R., Schwab, C.: Multilevel frames for sparse tensor product spaces. Numerische Mathematik **110**(2), 199–220 (2008). DOI 10.1007/s00211-008-0162-x. URL http://dx.doi.org/10.1007/s00211-008-0162-x

17. Hegland, M., Garcke, J., Challis, V.: The combination technique and some generalisations. Linear Algebra and its Applications **420**(2), 249–275 (2007). DOI https://doi.org/10.1016/j.laa.2006.07.014. URL http://www.sciencedirect.com/science/article/pii/S002437950600334X
18. Oswald, P.: Multilevel finite element approximation. Theory and applications. Teubner Skripten zur Numerik. B.G. Teubner, Stuttgart (1994)
19. Ruge, J., Stüben, K.: Algebraic multigrid (AMG). In: S. McCormick (ed.) Multigrid Methods, Frontiers in Applied Mathematics, vol. 5. SIAM, Philadelphia (1986)
20. Schwab, C., Todor, R.A.: Sparse finite elements for elliptic problems with stochastic loading. Numerische Mathematik **95**(4), 707–734 (2003). DOI http://dx.doi.org/10.1007/s00211-003-0455-z
21. Schwab, C., Todor, R.A.: Sparse finite elements for stochastic elliptic problems: Higher order moments. Computing **71**(1), 43–63 (2003). DOI 10.1007/s00607-003-0024-4. URL http://dx.doi.org/10.1007/s00607-003-0024-4
22. Stüben, K.: A review of algebraic multigrid. Journal of Computational and Applied Mathematics **128**(1-2), 281–309 (2001). Numerical Analysis 2000. Vol. VII: Partial Differential Equations
23. Trottenberg, U., Schuller, A.: Multigrid. Academic Press, Inc., Orlando, FL, USA (2001)
24. Yang, U.M.: On long-range interpolation operators for aggressive coarsening. Numerical Linear Algebra with Applications **17**(2-3), 453–472 (2010). DOI 10.1002/nla.689. URL http://dx.doi.org/10.1002/nla.689
25. Zaspel, P.: Subspace correction methods in algebraic multi-level frames. Linear Algebra and its Applications **488**, 505–521 (2016). DOI https://doi.org/10.1016/j.laa.2015.09.026. URL http://www.sciencedirect.com/science/article/pii/S0024379515005418
26. Zeiser, A.: Fast matrix-vector multiplication in the sparse-grid Galerkin method. SIAM Journal of Scientific Computing **47**(3), 328–346 (2011). DOI 10.1007/s10915-010-9438-2. URL https://doi.org/10.1007/s10915-010-9438-2