Adv Comput Math (2020) 46: 59
https://doi.org/10.1007/s10444-020-09797-9

HT-AWGM: a hierarchical Tucker–adaptive wavelet Galerkin method for high-dimensional elliptic problems

Mazen Ali[1] · **Karsten Urban**[1]

Received: 8 August 2019 / Accepted: 30 May 2020 /
Published online: 6 July 2020

Abstract
This paper is concerned with the construction, analysis, and realization of a numerical method to approximate the solution of high-dimensional elliptic partial differential equations. We propose a new combination of an adaptive wavelet Galerkin method (AWGM) and the well-known hierarchical tensor (HT) format. The arising HT-AWGM is adaptive both in the wavelet representation of the low-dimensional factors and in the tensor rank of the HT representation. The point of departure is an adaptive wavelet method for the HT format using approximate Richardson iterations and an AWGM for elliptic problems. HT-AWGM performs a sequence of Galerkin solves based upon a truncated preconditioned conjugate gradient (PCG) algorithm in combination with a tensor-based preconditioner. Our analysis starts by showing convergence of the truncated conjugate gradient method. The next step is to add routines realizing the adaptive refinement. The resulting HT-AWGM is analyzed concerning convergence and complexity. We show that the performance of the scheme asymptotically depends only on the desired tolerance with convergence rates depending on the Besov regularity of low-dimensional quantities and the low-rank tensor structure of the solution. The complexity in the ranks is algebraic with powers of four stemming from the complexity of the tensor truncation. Numerical experiments show the quantitative performance.

Keywords High dimensional · Hierarchical Tucker · Low-rank tensor methods · Adaptive wavelet Galerkin methods · Partial differential equations

Communicated by: Alexander Barnett

✉ Mazen Ali
mazen.ali@uni-ulm.de

Karsten Urban
karsten.urban@uni-ulm.de

[1] Ulm University, Inst. f. Numerical Mathematics, Helmholtzstr. 20, 89081, Ulm, Germany

Mathematics Subject Classification (2010) 65N99

1 Introduction

The increase of available computational power made a variety of complex problems accessible for computer-based simulations. However, the complexity of problems has increased even faster, so that several "real-world" problems will be out of reach even with computers of the next generations. One class of such challenging problems arises from high-dimensional models suffering from the *curse of dimensionality*. This shows the ultimate need to construct and analyze sophisticated numerical methods.

This paper is concerned with high-dimensional systems of elliptic partial differential equations (PDEs). Examples include chemical reactions, financial derivatives, equations depending on a large number of parameters (e.g., material properties), or a large number of independent variables. In general terms, we consider an operator problem $Au = f$, where $A : \mathcal{X} \to \mathcal{X}'$ is elliptic,[1] $f \in \mathcal{X}'$ is given, and $u \in \mathcal{X}$ is the desired solution, which we aim to approximate in a possible "sparse" manner.

Of course, this issue also depends on the specific notion of sparsity, which itself is typically adapted to the problem. In the context of adaptive methods (think of adaptive finite element or wavelet methods), the sparsity benchmark is a *Best N-term* approximation, i.e., a possibly optimal approximation to $u \in \mathcal{X}$ using $N \in \mathbb{N}$ degrees of freedom. In particular for high-dimensional problems, one tries to approximate u in terms of *low-rank* tensor format approximations. We will combine these two notions to be explained next.

1.0.1 Best *N*-term approximation

Given a dictionary (basis, frame) $\Psi := \{\psi_\lambda : \lambda \in \mathcal{J}\} \subset \mathcal{X}$, where the index set $\mathcal{J}$ is typically of infinite cardinality, one seeks an approximate expansion of u in Ψ. A best N-term approximation is of the form $u \approx u_N := \sum_{\lambda\in\Lambda} c_\lambda \psi_\lambda$, $c_\lambda \in \mathbb{R}$ and $\Lambda \subset \mathcal{J}$ is of cardinality $N \in \mathbb{N}$, i.e., $|\Lambda| = N$. The goal of an optimal approximation can also be expressed by determining the minimal number of terms $N(\varepsilon)$ required to achieve a certain accuracy $\varepsilon > 0$: $\|u - u_{N(\varepsilon)}\|_{\mathcal{X}} \le \varepsilon$.

It is known that the optimal speed of convergence of such approximations entirely depends on the properties of the solution u and the chosen basis. In fact, there is an intimate connection between decay of the error of the best N-term approximation and the Besov regularity of u, see [13]. An approximation scheme (or algorithm) is called *quasi-optimal* if it realizes (asymptotically) the same rate as the N-term approximation. Known quasi-optimal methods are adaptive in the sense that approximations are constructed in nonlinear manifolds rather than in linear subspaces.

For adaptive finite element methods (AFEM, [28]) and adaptive wavelet methods (AWM, e.g., [8, 9, 17]), there are quasi-optimal algorithms known, in particular for elliptic problems.

[1] We assume that $\mathcal{X} \hookrightarrow \mathcal{H} \hookrightarrow \mathcal{X}'$ is a Gelfand triple with a pivot Hilbert space $\mathcal{H}$ and $\mathcal{X}'$ is the dual space of $\mathcal{X}$ induced by $\mathcal{H}$.

1.0.2 Low-rank tensor methods

For high-dimensional problems ($d \gg 1$), it is well-known that most algorithms scale exponentially in the dimension and are thus intractable: they suffer from the curse of dimensionality. If the operator A has a tensor structure (or can at least be well-approximated by such), one can try to find an efficient separable approximation

$$u \approx \sum_{i=1}^{r} \bigotimes_{j=1}^{d} v_j^i, \tag{1.1}$$

where r is referred to as the *rank* and $v_1 \otimes \cdots \otimes v_d(x) := v_1(x_1) \cdots v_d(x_d)$ for $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ is a tensor product. Hence, if the rank r is small even for large d, one can try to approximate the univariate factors $v_j^i : \mathbb{R} \to \mathbb{R}$ separately resulting in a tractable algorithm.

A major breakthrough in this area was the development of tensor formats that in fact realized such approximations. We mention *the hierarchical Tucker (HT) format* [20] and the *tensor train format* [29] and refer to [19] for a general overview. Nowadays, there is a whole variety of algorithms that have been developed in these formats, both iterative solvers [6, 24, 26, 27] (using basic arithmetic operations on tensors and truncations to control the rank) and direct methods [14, 21, 25, 30], which work within the tensor structure itself. For a survey on tensor methods for solving high-dimensional PDEs, we refer to [5].

1.0.3 HTucker-adaptive wavelet Galerkin method

In this paper, we consider a combination of best N-term and low-rank approximations in order to obtain a convergent algorithm that is optimal both w.r.t. N and the tensor rank r. To this end, we use appropriate wavelet bases Ψ, i.e., the factors in Eq. 1.1 are approximated by sparse wavelet expansions

$$v_j^i = \sum_{\lambda \in \Lambda_j^i} c_\lambda^{i,j} \psi_\lambda^j, \quad c_\lambda^{i,j} \in \mathbb{R}.$$

To the best of our knowledge, the first such approximation was constructed in [1], where inexact Richardson iterations from [9] were combined with the HT format from [20]. In [4], the authors considered soft thresholding techniques for rank reduction.

The goal of this paper is to extend the AWGM method with coarsening from [8] to the high-dimensional setting using the HT format—resulting in an *HTucker-adaptive wavelet Galerkin method (HT-AWGM)*. In particular, we aim at providing the corresponding convergence analysis. A core ingredient of AWGM is the fact that wavelet bases can be used to rewrite the operator equation $Au = f$ equivalently into an equation $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ in sequence spaces, where $\boldsymbol{A}$ is boundedly invertible. The backbone of that is optimal wavelet preconditioning. Hence, a tensor-based wavelet preconditioner is needed. Luckily, in [3] the problem of separable preconditioning was addressed and the algorithm from [1] was extended to the elliptic case.

1.0.4 Organization of the paper

The remainder of this paper is organized as follows. In Section 2, we collect all required preliminaries. As a core ingredient for the new HT-AWGM, we use a truncated PCG algorithm from [27, Algorithm 2] and analyze its convergence in Section 3. The convergence and complexity analysis of the full HT-AWGM is described in Section 4. We show numerical results in Section 5. We indicate the potential and remaining issues of the method.

2 Preliminaries

We start by briefly reviewing some basic facts on adaptive wavelet methods, low-rank tensor formats, and the preconditioning problem arising in connection with tensor spaces.

2.1 (Quasi-)optimal approximations

For the remainder of this work, we use the shorthand notation

$$A \lesssim B,$$

to indicate there exists a constant $C > 0$ independent of A and B such that $A \leq CB$. The notation $A \gtrsim B$ is defined analogously.

The introduction mainly follows [35]. We seek the solution of the operator equation

$$Au = f, \quad A : \mathcal{X} \to \mathcal{X}', \quad u \in \mathcal{X}, \quad f \in \mathcal{X}', \tag{2.1}$$

where A is a linear boundedly invertible operator and $\mathcal{X}$ is a separable Hilbert Space. Given a Riesz basis $\Psi := \{\psi_\lambda : \lambda \in \mathcal{J}\}$, e.g., a wavelet basis, and the corresponding boundedly invertible analysis and synthesis operators

$$\mathcal{F} : \mathcal{X}' \to \ell_2(\mathcal{J}), \quad f \mapsto \{f(\psi_\lambda)\}_\lambda, \qquad \mathcal{F}' : \ell_2(\mathcal{J}) \to \mathcal{X}, \quad \{c_\lambda\}_\lambda \mapsto \sum_{\lambda \in \mathcal{J}} c_\lambda \psi_\lambda,$$

we can reformulate (2.1) equivalently as a discrete infinite-dimensional linear system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}, \quad \boldsymbol{A} : \ell_2(\mathcal{J}) \to \ell_2(\mathcal{J}), \quad \boldsymbol{u}, \boldsymbol{f} \in \ell_2(\mathcal{J}), \tag{2.2}$$

with $\boldsymbol{A} := \mathcal{F}A\mathcal{F}'$, $\boldsymbol{u} := \mathcal{F}\mathcal{R}u$, and $\boldsymbol{f} := \mathcal{F}f$, where $\mathcal{R} : \mathcal{X} \to \mathcal{X}'$ is the Riesz isomorphism. The operator $\boldsymbol{A}$ inherits the properties of its continuous counterpart A and is in particular boundedly invertible as well.

Next, we introduce the notation for the Galerkin problem. Let $\Lambda \subset \mathcal{J}$ be some finite index subset. We introduce the restriction operator $R_\Lambda : \ell_2(\mathcal{J}) \to \ell_2(\Lambda)$, which simply drops all entries outside Λ. Likewise the extension operator $E_\Lambda : \ell_2(\Lambda) \to \ell_2(\mathcal{J})$ pads all entries outside Λ with zeros. We will sometimes employ the notation $\boldsymbol{A}_\Lambda := R_\Lambda \boldsymbol{A} E_\Lambda$ to denote the discretized wavelet operator.

The benchmark for optimal approximations is the *best N-term approximation*

$$u_N := \arg\min \left\{ \|u - v\|_{\mathcal{X}} : v \in \mathcal{X},\ v = \sum_{\lambda \in \Lambda \subset \mathcal{J}} v_\lambda \psi_\lambda,\ \#\Lambda \leq N \right\},$$

or, equivalently, in $\ell_2(\mathcal{J})$

$$\boldsymbol{u}_N := \arg\min \left\{ \|\boldsymbol{u} - \boldsymbol{v}\|_{\ell_2} : \boldsymbol{v} \in \ell_2(\mathcal{J}),\ \#\operatorname{supp}(\boldsymbol{v}) \leq N \right\},$$

where $\operatorname{supp}(\boldsymbol{v})$ denotes those wavelet indices $\lambda \in \mathcal{J}$, for which $\boldsymbol{v}_\lambda \neq 0$. Note that, as opposed to linear approximation techniques, we seek an approximation in an N-dimensional nonlinear manifold. The approximation class of all best N-term approximations converging with rate s is known as

$$\mathcal{A}_s := \left\{ \boldsymbol{u} \in \ell_2(\mathcal{J}) : \|\boldsymbol{u}\|_{\mathcal{A}_s} := \sup_{\varepsilon > 0} \varepsilon [\min\{N \in \mathbb{N}_0 : \|\boldsymbol{u} - \boldsymbol{u}_N\|_{\ell_2} \leq \varepsilon\}]^s < \infty \right\}. \quad (2.3)$$

It is known that such approximation spaces are interpolation spaces between L_p and certain Besov spaces, which establishes a direct link between regularity and approximation classes, see also [13] for more details.

An adaptive wavelet method is called *(quasi-)optimal* whenever it produces for $\boldsymbol{u} \in \mathcal{A}_s$ an approximation $\boldsymbol{v}$ to $\boldsymbol{u}$ with $\|\boldsymbol{u} - \boldsymbol{v}\|_{\ell_2} \leq \varepsilon$, such that $\#\operatorname{supp}(\boldsymbol{v}) \lesssim \varepsilon^{-1/s} \|\boldsymbol{u}\|_{\mathcal{A}_s}^{1/s}$ and the number of operators is bounded by a multiple of the same quantity. In other words, given that $\boldsymbol{u}$ is in a certain approximation class, an optimal adaptive method achieves the best possible asymptotic rate of convergence in linear computational complexity of the output size.

There are two classical approaches to implementing such an optimal adaptive wavelet method (see [8, 9, 17]). The first[2] applies an *inexact iteration method* such as the Richardson iteration, to the bi-infinite discrete system in Eq. 2.2. The second one, in the spirit of adaptive FEM methods, produces a sequence $\Lambda^{(0)} \to \Lambda^{(1)} \to \cdots$ of finite index sets and solves the finite Galerkin problem on these sets, yielding a sequence of solutions $\boldsymbol{u}^{(0)} \to \boldsymbol{u}^{(1)} \to \cdots$, following the paradigm solve $\to$ estimate $\to$ mark $\to$ refine. The latter one is referred to as an *adaptive wavelet Galerkin method (AWGM)*.

In [8], the authors introduced an AWGM that required additional coarsening to ensure minimality of the active index sets, i.e., solve $\to$ estimate $\to$ mark $\to$ refine $\to$ coarsen. In [17], it was shown that optimality can be guaranteed *without* the additional coarsening step. In this work, we require an additional coarsening step as in [8] for the same purpose. Removing this step as in [17], optimality can no longer be guaranteed in our setting of low-rank approximation, and we briefly address this issue in Section 4.7.

There are three basic routines necessary for an efficient realization of an AWGM: (1) approximate residual evaluation (Estimate), (2) approximate Galerkin solver (solve), and (3) bulk chasing (mark and refine). We do not discuss these routines in detail here, but refer to the literature. In order to control the number of active variables (number of selected wavelets), one often uses a coarsening step in order to

[2] Chronologically, however, the second.

remove "unnecessary" coefficients. This is done by a routine called **COARSE**, which we detail for later use: For a given finitely supported $\boldsymbol{v}$, such routine is assumed to produce an approximation $\boldsymbol{v}_\varepsilon$ such that

$$\#\operatorname{supp}(\boldsymbol{v}_\varepsilon) \lesssim \min\left\{N : \|\boldsymbol{v}-\boldsymbol{w}\|_{\ell_2} \le \varepsilon,\ \boldsymbol{w} \in \ell_2(\mathcal{J}),\ \#\operatorname{supp}(\boldsymbol{w}) \le N\right\}.$$

A straightforward realization would involve sorting—with log linear complexity. To achieve linear complexity, exact sorting can be replaced by an approximate bin sorting which satisfies the above estimate. We refer to [17].

AWGM requires that A, or, equivalently, $\boldsymbol{A}$ is symmetric positive definite. Otherwise, a similar analysis applies to the normal equations with $\boldsymbol{A}^T\boldsymbol{A}$. However, the additional application of $\boldsymbol{A}^T$ hampers numerical performance and convergence estimates depend on $\kappa(\boldsymbol{A})^2$ rather than on $\kappa(\boldsymbol{A})$. The penalty for applying $\boldsymbol{A}^T$ is even more severe in the high-dimensional case due to the increase in ranks. The Richardson iterations can be applied to non-symmetric $\boldsymbol{A}$; however, in this case, the convergence will depend on $\kappa(\boldsymbol{A})^2$ as well. See [35, Section 3].

2.2 Tensor formats

We briefly review some of the basics of tensor formats, see, e.g., [19]. A *tensor of order d* is an element of a tensor space $\mathcal{V} := \otimes_{j=1}^d V_j$, where V_j are some vector spaces. We consider *topological* tensor spaces, i.e., $\mathcal{V}$ is Banach space with some norm $\|\cdot\|_{\mathcal{V}}$. Typically, V_j are themselves Banach spaces and the norm on $\mathcal{V}$ is induced by the norms on V_j. The *tensor product* $\otimes : V_1 \times \cdots \times V_d \to V_1 \otimes \cdots \otimes V_d$ is the unique multilinear mapping factoring any other multilinear mapping $\varphi : V_1 \times \cdots \times V_d \to W$ into a linear mapping $f : V_1 \otimes \cdots \otimes V_d \to W$ such that $\varphi = f \circ \otimes$, where V_j and W are some vector spaces.

The *hierarchical Tucker (HT) format* combines both the advantages of stable approximation of the Tucker format with the sparse representation of the r-term format by further decomposing the core tensor. For a general multi-index $\alpha \subset \{1,\dots,d\}$, we can define the tensor product vector space

$$V_\alpha := \bigotimes_{j\in\alpha} V_j.$$

The idea behind HT can be illustrated by the following simple observation: An element $u \in \mathcal{V}$ can be also seen as an element of $u \in V_\alpha \otimes V_{\bar\alpha}$ with $\alpha, \bar\alpha \subset \{1,\dots,d\}$ with $\bar\alpha$ being the complement of α. This defines the notion of *multilinear ranks* $r(u)_\alpha$ that depends on the choice of α. Applying this idea recursively, we start with a Tucker decomposition of $u \in V_\alpha \otimes V_{\bar\alpha}$. We then further decompose the bases U_α and $U_{\bar\alpha}$ of V_α and $V_{\bar\alpha}$, respectively, until we reach the singletons $\alpha = \{j\}$. We denote the ranks of this hierarchical representation by $r(u) = (r(u)_\alpha)_{\alpha\in T}$ with the max norm $|r(u)|_\infty$ defined in an obvious way, where T is the HT tree structure. In contrast to the Tucker format, which requires the storage of an order d tensor, the HT format stores several order 3 tensors.[3] However, note that in the worst case $r(u)$ can still behave

[3]Due to the binary decomposition $\alpha = \alpha_L \cup \alpha_R$, each transfer tensor has 2 indices related to the child nodes α_L, α_R and one index related to the parent node α.

exponentially w.r.t. d. Nonetheless, it is known that the asymptotic behavior of the storage requirements of HT are not worse than that of the r-term format and the performance of HT in practice has proven its merit. A rigorous answer to the question as to when and why functions exhibit good approximation properties in tensor tree formats remains a challenging and interesting problem.

As in the case for best N-term approximations in Eq. 2.3, we require a benchmark to assess the quality of the ranks of approximation. For this purpose, we use the benchmark introduced in [1], similar to Eq. 2.3. We use the notation $u \in \mathcal{H}_N$ to denote that u is representable in an HT format with $|r(u)|_\infty \leq N$. Given a positive, strictly increasing growth sequence, $\gamma := (\gamma(n))_{n\in\mathbb{N}_0}$ with $\gamma(0) = 1$, define an approximation class as

$$\mathcal{A}(\gamma) := \left\{ v \in \mathcal{V} : |v|_{\mathcal{A}(\gamma)} := \sup_{N\in\mathbb{N}_0} \gamma(N) \inf_{w\in\mathcal{H}_N} \|v - w\|_{\mathcal{V}} < \infty \right\},$$

with norm $\|v\|_{\mathcal{A}(\gamma)} = \|v\|_{\mathcal{V}} + |v|_{\mathcal{A}(\gamma)}$. It is known from, e.g., [33] that the best approximation error for a function with Sobolev smoothness s behaves in the worst case like

$$\max_{\alpha\in T\setminus\{1,\dots,d\}} r_\alpha^{-s\max\{1/|\alpha|,1/(d-|\alpha|)\}}.$$

One of the most important operations on tensors is truncation. It lies in the heart of all iterative tensor algorithms that rely on truncation to keep ranks low. For a given algebraic tensor $u \in \mathcal{V}$, we seek an approximation $v \in \mathcal{V}$ with $r(v)_\alpha \leq r_\alpha \leq r(u)_\alpha$ for some fixed r_α and all $\alpha \subset \{1,\dots,d\}$. In practice, this can be done by applying singular value decompositions (SVD) to matricizations $\mathcal{M}_\alpha(u) \in V_\alpha \otimes V_{\tilde{\alpha}}$, a method referred to as *higher order singular value decomposition* (HOSVD). Unlike the standard SVD, the HOSVD provides one only with a quasi-best approximation in the sense

$$\|u - v_{\mathrm{HOSVD}}\|_{\mathcal{V}} \leq \sqrt{\sum_\alpha \sum_{i\geq r_\alpha+1} (\sigma_i^\alpha)^2} \leq \sqrt{2d-3} \inf_{\substack{v\in\mathcal{V},\\ r(v)\leq r}} \|u - v\|_{\mathcal{V}}, \tag{2.4}$$

where $r = (r_\alpha)_\alpha$ is some integer vector and σ_i^α are the corresponding singular values of the α matricization. We will denote the (nonlinear) operator that produces an HOSVD of u by $\mathcal{T}(u,\varepsilon)$, i.e.,

$$\|u - \mathcal{T}(u,\varepsilon)\|_{\mathcal{V}} \leq \varepsilon.$$

The total computational work for truncating a tensor u can be bounded by a constant multiple of $dr^4 + r^2\sum_{j=1}^d n_j$, where $r = |r(u)|_\infty$ and $n_j := \dim(V_j)$.

We need to combine the wavelet coarsening with the tensor rank truncation. Recall that to apply **COARSE** to a tensor $u \in \mathcal{V}$ of finite support in the wavelet dictionary, we would have to search through all entries of u, a process that scales exponentially in d. Thus, we require low-dimensional quantities that allow us to perform this task.

For this purpose, we use *contractions*[4] introduced in [1]. For a tensor $\boldsymbol{u} \in \ell_2(\mathcal{J}^d)$ where $\mathcal{J}$ is a 1D wavelet index set, we set

$$\pi_j(\boldsymbol{u}) = (\pi_j(\boldsymbol{u})[\lambda_j])_{\lambda_j \in \mathcal{J}} := \left(\sqrt{\sum_{\lambda_1,\ldots,\lambda_{j-1},\lambda_{j+1},\ldots,\lambda_d \in \mathcal{J}^{d-1}} |\boldsymbol{u}_{\lambda_1,\ldots,\lambda_j,\ldots,\lambda_d}|^2} \right)_{\lambda_j \in \mathcal{J}}. \quad (2.5)$$

Recalling the restriction operator

$$R_{\mathcal{J}_1 \times \ldots \times \mathcal{J}_d} \boldsymbol{u}[\lambda] := \begin{cases} \boldsymbol{u}[\lambda], & \text{if } \lambda \in \mathcal{J}_1 \times \ldots \times \mathcal{J}_d, \\ 0, & \text{otherwise,} \end{cases}$$

the two important properties of these contractions are

$$\begin{aligned} \pi_j(\boldsymbol{u})[\lambda_j] &= \sqrt{\sum_k |\sigma_k^j|^2 |\boldsymbol{U}_j^k(\lambda_j)|^2}, \\ \|(I - R_{\mathcal{J}_1 \times \ldots \times \mathcal{J}_d})\boldsymbol{u}\| &\leq \sqrt{\sum_{j=1}^d \sum_{\lambda \in \mathcal{J} \setminus \mathcal{J}_j} |\pi_j(\boldsymbol{u})[\lambda]|^2}, \\ &\leq \sqrt{d} \|(I - R_{\mathcal{J}_1 \times \ldots \times \mathcal{J}_d})\boldsymbol{u}\|, \end{aligned} \quad (2.6)$$

where $\boldsymbol{U}_j^k$ is the k-th column of the j-th HOSVD basis frame and σ_k^j are the corresponding singular values. We use the notation

$$\mathrm{supp}_j(\boldsymbol{u}) := \mathrm{supp}(\pi_j(\boldsymbol{u})),$$

to refer to the 1D support of $\boldsymbol{u}$ along the j-th dimension, i.e., $\boldsymbol{u}$ can be viewed as $\boldsymbol{u} \in \ell_2(\mathrm{supp}_1(\boldsymbol{u}) \times \cdots \times \mathrm{supp}_d(\boldsymbol{u}))$.

2.3 Separable preconditioning

Suppose we want to solve an equation on the Sobolev space $\mathcal{X} \subset H^s(\Omega)$ on a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$ with appropriate boundary conditions. Typically, the point of departure is a Riesz wavelet basis Ψ_{L_2} for $L_2(\Omega)$ from which we obtain a whole range of Riesz bases for H^s by a simple diagonal scaling (see, e.g., [36, Section 5.6.3]) $\Psi_{H^1} := \boldsymbol{D}^{-s}\Psi_{L_2}$, where $\boldsymbol{D} := (\delta_{\lambda,\mu}\|\psi_\lambda\|_{H^1})_{\lambda,\mu}$. This is equivalent to reformulating (2.2) as the preconditioned infinite system

$$\boldsymbol{D}^{-s}\boldsymbol{A}\boldsymbol{D}^{-s}\boldsymbol{D}\boldsymbol{u} = \boldsymbol{D}^{-s}\boldsymbol{f}. \quad (2.7)$$

In the context of high-dimensional problems, $d \gg 1$ is large and the complexity of approximating the solution to Eq. 2.2 will in general scale exponentially with the dimension d (see, e.g., [35, Section 7]). However, given a product structure of the domain $\Omega = \times_{j=1}^d \Omega_j$ (or smooth images thereof), the problem (2.2) can be solved

[4] We remark that this is a slight abuse of terminology for general tensor contractions.

with tractable methods (see, e.g., [10]). For this, we will need Ψ to be a tensorised basis of lower-dimensional components, i.e., $\Psi := \times_{j=1}^{d} \Psi_j$ and we reconsider $\mathcal{X}$ as a tensor space $\mathcal{X} = \otimes_{j=1}^{d} \mathcal{X}_j$. This way, if A permits a separable structure or can be well approximated in such a form, than we can discretize A such that it preserves the product structure with low-dimensional components.

Unfortunately, the space $H^s(\Omega)$ is not equipped with a cross norm, i.e., for an elementary tensor product $v = v_1 \otimes \cdots \otimes v_d$

$$\|v\|_s \neq \|v_1\|_s \cdots \|v_d\|_s.$$

Considering again (2.7), this means that $\boldsymbol{D}^{-s}$ cannot be represented in a separable form. However, this issue was addressed in [3], where the exact preconditioning $\boldsymbol{D}^{-s}$ was replaced by an approximate separable scaling via exponential sum approximations. We will utilize this separate scaling both for preconditioning the Galerkin solver and the approximate residual evaluation. We briefly recall some basic properties of the said preconditioning.[5]

For certain parameters $\delta > 0$, $\eta > 0$, $T > 1$, we choose $h \in \left(0, \frac{\pi^2}{5(|\ln(\delta/2)|+4)}\right)$, $n^+ \geq h^{-1} \max\left\{\frac{4}{\sqrt{\pi}}, \sqrt{|\ln(\delta/2)|}\right\}$, and $n \geq h^{-1}\left(\ln \frac{2}{\sqrt{\pi}} + |\ln(\min\{\delta/2, \eta\})| + \frac{1}{2}\ln T\right)$. The approximation involved is

$$\frac{1}{\sqrt{t}} = \frac{2}{\sqrt{\pi}} \int_{\mathbb{R}} \frac{e^{-t\ln^2(1+e^x)}}{1+e^{-x}} dx \approx \sum_{k=-n}^{n^+} h w(kh) e^{-\alpha(kh)t} =: \varphi_{n^+,n}(t),$$

where $w(x) := \frac{2}{\sqrt{\pi}}(1+e^{-x})^{-1}$, $\alpha(x) := \ln^2(1+e^x)$, and $t > 0$ is some scaling weight. We get

$$\left|\frac{1}{\sqrt{t}} - \varphi_{n^+,n}(t)\right| \leq \frac{\delta}{\sqrt{t}}, \quad |\varphi_{n^+,\infty}(t) - \varphi_{n^+,n}(t)| \leq \frac{\eta}{\sqrt{t}}, \tag{2.8}$$

for all $t \in [1, T]$. For the exact diagonal preconditioning, the scaling weights for tensor product wavelets can be obtained by observing that H^1 (and similarly H^s) is isomorphic to the intersection of Hilbert spaces

$$H^1(\Omega) \cong \bigcap_{j=1}^{d} L_2(\Omega_1) \otimes \cdots \otimes H^1(\Omega_j) \otimes \cdots \otimes L_2(\Omega_d), \quad \text{with } \Omega = \Omega_1 \times \cdots \times \Omega_d.$$

The norm on the intersection space leads to the scaling weight $t := \sum_{j=1}^{d} \|\psi_{\lambda_j}\|_{H^1}^2$, for $\psi_\lambda = \otimes_{j=1}^{d} \psi_{\lambda_j}$. We will denote by

$$\boldsymbol{S}(\delta, \eta) \qquad \text{and} \qquad \boldsymbol{S}(\delta) := \lim_{\eta \to 0} \boldsymbol{S}(\delta, \eta)$$

[5] For ease of presentation, we restrict ourselves to $s = 1$.

the corresponding separable approximation to $\boldsymbol{D}$ and the limit, respectively. We mention important properties from [3] for later use

$$\begin{aligned}
\|\boldsymbol{D}\boldsymbol{S}^{-1}(\delta,\eta)\| &\leq 1+\delta, \quad \forall \eta > 0, && \text{(2.9a)}\\
\|\boldsymbol{D}\boldsymbol{S}^{-1}(\delta)\| &\leq 1+\delta, && \text{(2.9b)}\\
\|\boldsymbol{S}(\delta)\boldsymbol{D}^{-1}\| &\leq \frac{1}{1-\delta}, && \text{(2.9c)}\\
\|\boldsymbol{D}(\boldsymbol{D}^{-1}-\boldsymbol{S}^{-1}(\delta,\eta))R_{\mathcal{J}_T}\| &\leq \delta, \quad \forall \eta > 0, && \text{(2.9d)}\\
\|\boldsymbol{D}(\boldsymbol{S}^{-1}(\delta)-\boldsymbol{S}^{-1}(\delta,\eta))R_{\mathcal{J}_T}\| &\leq \eta, \quad \forall \delta > 0, && \text{(2.9e)}\\
\|\boldsymbol{S}(\delta)(\boldsymbol{S}^{-1}(\delta)-\boldsymbol{S}^{-1}(\delta,\eta))R_{\mathcal{J}_T}\| &\leq \frac{\eta}{1-\delta}, && \text{(2.9f)}\\
\boldsymbol{S}^{-1}(\delta,\eta) &\leq \boldsymbol{S}^{-1}(\delta), \quad \forall \eta > 0, && \text{(2.9g)}\\
1-\delta &\leq \boldsymbol{S}^{-1}(\delta)\boldsymbol{D} \leq 1+\delta, && \text{(2.9h)}\\
1-\delta &\leq \left(\boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{D}\right)_{\lambda\in\mathcal{J}_T} \leq 1+\delta, \quad \forall \eta > 0, && \text{(2.9i)}
\end{aligned}$$

where the last three inequalities are to be understood componentwise. The index set $\mathcal{J}_T$ indicates the set of wavelet indices for which (2.8) holds, depending on the parameter T. We thus seek to approximate the solution of the preconditioned equation

$$\boldsymbol{S}^{-1}(\delta)\boldsymbol{A}\boldsymbol{S}^{-1}(\delta)\boldsymbol{S}(\delta)\boldsymbol{u} = \boldsymbol{A}^{\delta}\boldsymbol{u}^{\delta} = \boldsymbol{f}^{\delta} = \boldsymbol{S}^{-1}(\delta)\boldsymbol{f},$$

with the shorthand notation

$$\boldsymbol{S}^{-1}(\delta)\boldsymbol{A}\boldsymbol{S}^{-1}(\delta) =: \boldsymbol{A}^{\delta}, \quad \boldsymbol{S}^{-1}(\delta)\boldsymbol{f} =: \boldsymbol{f}^{\delta}, \quad \boldsymbol{S}(\delta)\boldsymbol{u} =: \boldsymbol{u}^{\delta}.$$

We assume a product structure of the operator $\boldsymbol{A}$. I.e., we can write $\boldsymbol{A}$ as

$$\boldsymbol{A} = \sum_{1\leq n_1,\dots,n_d\leq R} c_{n_1,\dots,n_d} \bigotimes_{j=1}^{d} \boldsymbol{A}_j^{n_j}. \tag{2.10}$$

for some one-dimensional components $\boldsymbol{A}_j^{n_j}$ and coefficients $c_{n_1,\dots,n_d}$. Moreover, $\boldsymbol{D}$ represents the exact diagonal scaling $\boldsymbol{D} := (\delta_{\lambda,\mu}d_\lambda)_{\lambda,\mu}$ for some weights $d_\lambda^2 = d_{\lambda_1,\dots,\lambda_d}^2 = \sum_{j=1}^d d_{\lambda_j}^2$, $\lambda \in \mathcal{J}^d$. We use $\boldsymbol{D}_j := (\delta_{\lambda_j,\mu_j}d_{\lambda_j})_{\lambda_j\in\mathcal{J},\mu_j\in\mathcal{J}}$ to denote the corresponding one-dimensional components.

3 Perturbed finite-dimensional descent method

For further presentation, we formulate a general descent method with perturbations for solving the linear system $Ax = b$, $A : \mathcal{V} \to \mathcal{V}$, $x, b \in \mathcal{V}$, where A is an s.p.d. matrix and $\mathcal{V}$ is a finite-dimensional vector space, possibly an algebraic tensor space with $N := \dim(\mathcal{V}) < \infty$, i.e., $\mathcal{V} \cong \mathbb{R}^N$.

For simplicity of presentation, we omit preconditioning at this point. The analysis for the case of exact preconditioning remains the same. Approximate preconditioning adds a perturbation to the descent direction.

We will frequently use the associated quadratic functional

$$f(x) := \frac{1}{2}\langle x, Ax\rangle - \langle b, x\rangle \equiv f_{A,b}(x),$$

where $\langle\cdot,\cdot\rangle$ denotes the standard inner product on $\mathcal{V}$ with induced Euclidean norm $\|\cdot\|$ and $\|\cdot\|_A := \langle\cdot, A\cdot\rangle$ the energy norm. The very well-known descent method then reads as follows.

Algorithm 1 Descent method for minimizing $f(x)$.

Input: $x^{(0)} \in \mathcal{V}$

1: $k \leftarrow 0$
2: **while** stopping criterion for $f(x^{(k)})$ not satisfied **do**
3: choose/update descent direction $d^{(k)}$
4: $d^{(k)} \leftarrow d^{(k)} + \varepsilon_1^{(k)}$ (e.g., truncation)
5: compute step size α_k
6: $x^{(k+1)} \leftarrow x^{(k)} + \alpha_k d^{(k)}$
7: $x^{(k+1)} \leftarrow x^{(k+1)} + \varepsilon_2^{(k+1)}$
8: $k \leftarrow k + 1$
9: **end while**

The abstract stopping criterion from line 2 depends on the particular choice of problem and descent method. E.g., in this work, we use a CG descent method for a linear problem, i.e., the algorithm terminates when the residual satisfies a certain error tolerance: see also line 4 of Algorithm 3.

In a tensor-based solver, lines 4 and 7 are typical candidates for truncating a tensor due to the increase in ranks after the summation. The quantities $\varepsilon_j^{(k)}$, $j = 1, 2$, represent the error incurred due to truncation, where $x^{(k)}$ is replaced by a truncated version $\tilde{x}^{(k)} := \mathcal{T}\left(x^{(k)}, \varepsilon\right)$, such that $\|\varepsilon_2^{(k)}\| \leq \varepsilon$ for the truncation error $\varepsilon_2^{(k)} := \tilde{x}^{(k)} - x^{(k)}$. We emphasize that the analysis has to rely solely on the control of the *magnitude* of $\varepsilon_2^{(k)}$ without restricting the *direction* of $\varepsilon_2^{(k)}$, which destroys optimality features of conjugate directions.

3.1 Gradient descent

Choosing $d^{(k)} = r^{(k)} + \varepsilon_2^{(k)}$, with $r^{(k)} := b - Ax^{(k)}$ being the residual, and using the optimal step size α_k leads to the well-known gradient-type descent method. The following lemma shows that appropriately choosing $\varepsilon_1^{(k)}$ and $\varepsilon_2^{(k)}$ ensures the same asymptotic convergence as the exact gradient descent method.

Proposition 3.1 *For the choice $d^{(k)} = r^{(k)}$ in line 3 and*

$$\alpha_k = \arg\min_{\alpha\in\mathbb{R}} f(x^{(k)} + \alpha d^{(k)})$$

Adv Comput Math (2020) 46:59

in line 5 (exact line search) of Algorithm 1*, we have the estimate for the error* $e^{(k)} := x^* - x^{(k)}$

$$\|e^{(k)}\|_A \leq \theta^k \|e^{(0)}\|_A + \sum_{j=0}^{k-1} \theta^{k-j-1} \left(\frac{\|\varepsilon_1^{(j)}\|_A}{\lambda_{\min}} + \|\varepsilon_2^{(j+1)}\|_A \right), \tag{3.1}$$

with reduction factor $\theta := \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}$*, and* $\lambda_{\max}$ *and* $\lambda_{\min}$ *being the largest and smallest eigenvalues of A, respectively.*

Proof It holds that the iterate $x^{(k+1)}$ can be written as $x^{(k+1)} = x^{(k)} + \alpha_k r^{(k)} + \alpha_k \varepsilon_1^{(k)} + \varepsilon_2^{(k+1)}$ and the error reads $e^{(k+1)} = (I - \alpha_k A)e^{(k)} + \alpha_k \varepsilon_1^{(k)} + \varepsilon_2^{(k+1)}$. The optimal step size is known to be $\alpha_k = \frac{\langle d^{(k)}, d^{(k)} \rangle}{\langle d^{(k)}, A d^{(k)} \rangle}$. Let $\{\lambda_j\}_{j=1,\ldots,N}$ denote the eigenvalues of A and $\{\psi_j\}_{j=1,\ldots,N}$ the corresponding orthonormal basis of eigenvectors. Since A is s.p.d., we get the standard estimate ($c_j := \langle d^{(k)}, \psi_j \rangle$)

$$\langle d^{(k)}, A d^{(k)} \rangle = \langle \sum_{j=1}^{N} c_j \psi_j, \sum_{j=0}^{N} c_j \lambda_j \psi_j \rangle = \sum_{j=1}^{N} \lambda_j c_j^2 \geq \lambda_{\min} \|d^{(k)}\|^2. \tag{3.2}$$

Using standard arguments for the analysis of the gradient descent method (cf. [18, Thm. 9.2.3]), we get $\|e^{(k+1)}\|_A \leq \theta \|e^{(k)}\|_A + \frac{\|\varepsilon_1^{(k)}\|_A}{\lambda_{\min}} + \|\varepsilon_2^{(k+1)}\|_A$, which proves (3.1). □

Remark 3.2 There are many related results on inexact gradient descent methods in optimization, see, e.g., [11, 12, 32], or in low-rank approximation, see, e.g., [7, 15]. The form of Proposition 3.1 highlights how perturbations in the descent direction *and* solution propagate through the iterations, and serves as a comparison for the error estimate of the perturbed CG method from Theorem 3.4.

3.2 Conjugate gradient descent

The (rank-)truncated (P)CG method was first proposed in [27, Algorithm 2] with promising numerical results.

Obviously, the perturbed CG method does not preserve orthogonality of the search directions w.r.t. $\langle \cdot, A \cdot \rangle$ and the resulting algorithm is not a Krylov method (see also below). Nevertheless, we can guarantee the perturbed CG to be a descent method which in turn will provide us with a convergence estimate.

Lemma 3.3 *Let* $\kappa := \frac{\lambda_{\max}}{\lambda_{\min}}$ *and fix some* $\tau \in (0, \frac{1}{\sqrt{1+\kappa^2}})$*. Let* $\delta_1, \delta_2 > 0$*, and* $\gamma > 0$ *be chosen such that* $\frac{3}{2}\delta_1 + \delta_2 \leq \frac{1}{\tau^2} - (1 + \kappa^2)$ *and* $(1 - \frac{\delta_1}{2})\tau \geq \gamma$*. If the error sequence* $\varepsilon_1^{(k)}$ *satisfies*

$$\|\varepsilon_1^{(k)}\| \leq \min \left\{ \frac{\delta_1}{2}, \frac{\delta_2 \|r^{(k)}\|}{2|\beta_{k-1}| \, \|d^{(k-1)}\|} \right\} \|r^{(k)}\|, \tag{3.3}$$

Algorithm 2 Truncated (P)CG method.

Input: $x^{(0)} \in \mathcal{V}$

1: $r^{(0)} \leftarrow b - Ax^{(0)}$, $d^{(0)} \leftarrow r^{(0)} + \varepsilon_1^{(0)}$
2: $k \leftarrow 0$
3: **while** stopping criterion for $f(x^{(k)})$ not satisfied **do**
4: $\quad \alpha_k \leftarrow \frac{\langle r^{(k)}, d^{(k)} \rangle}{\langle d^{(k)}, Ad^{(k)} \rangle}$,
5: $\quad x^{(k+1)} \leftarrow x^{(k)} + \alpha_k d^{(k)} + \varepsilon_2^{(k+1)}$
6: $\quad \beta_k \leftarrow -\frac{\langle r^{(k+1)}, Ad^{(k)} \rangle}{\langle d^{(k)}, Ad^{(k)} \rangle}$
7: $\quad d^{(k+1)} \leftarrow r^{(k+1)} + \beta_k d^{(k)} + \varepsilon_1^{(k+1)}$
8: $\quad k \leftarrow k + 1$
9: **end while**

then $d^{(k)}$ is a descent direction with $\langle r^{(k)}, d^{(k)} \rangle \geq \gamma \|r^{(k)}\| \|d^{(k)}\|$, where γ does not depend on k.

Proof First we show that $\|r^{(k)}\| \geq \tau \|d^{(k)}\|$. To this end, note that

$$\begin{aligned} \|d^{(k)}\|^2 = \langle d^{(k)}, d^{(k)} \rangle &= \|r^{(k)}\|^2 + \beta_{k-1}^2 \|d^{(k-1)}\|^2 + \|\varepsilon_1^{(k)}\|^2 + 2\beta_{k-1} \langle r^{(k)}, d^{(k-1)} \rangle \\ &\quad + 2\langle r^{(k)}, \varepsilon_1^{(k)} \rangle + 2\beta_{k-1} \langle d^{(k-1)}, \varepsilon_1^{(k)} \rangle. \end{aligned} \tag{3.4}$$

Next, we get

$$\begin{aligned} \langle r^{(k)}, d^{(k-1)} \rangle &= \langle b - A(x^{(k-1)} + \alpha_{k-1} d^{(k-1)}), d^{(k-1)} \rangle \\ &= \langle r^{(k-1)}, d^{(k-1)} \rangle - \frac{\langle r^{(k-1)}, d^{(k-1)} \rangle}{\langle d^{(k-1)}, Ad^{(k-1)} \rangle} \langle Ad^{(k-1)}, d^{(k-1)} \rangle = 0. \end{aligned}$$

For the term $\beta_{k-1}^2 \|d^{(k-1)}\|^2$, we get

$$\begin{aligned} \beta_{k-1}^2 \|d^{(k-1)}\|^2 &= \frac{|\langle r^{(k)}, Ad^{(k-1)} \rangle|^2}{|\langle d^{(k-1)}, Ad^{(k-1)} \rangle|^2} \|d^{(k-1)}\|^2 \overset{(3.2)}{\leq} \frac{|\langle r^{(k)}, Ad^{(k-1)} \rangle|^2}{\lambda_{\min}^2 |\langle d^{(k-1)}, d^{(k-1)} \rangle|^2} \|d^{(k-1)}\|^2 \\ &\leq \frac{\lambda_{\max}^2 \|r^{(k)}\|^2 \|d^{(k-1)}\|^2}{\lambda_{\min}^2 \|d^{(k-1)}\|^4} \|d^{(k-1)}\|^2 \leq \kappa^2 \|r^{(k)}\|^2. \end{aligned}$$

Using Eq. 3.3, we estimate the term (3.4) as $\|d^{(k)}\|^2 \leq (1 + \kappa^2 + \frac{\delta_1}{2} + \delta_1 + \delta_2) \|r^{(k)}\|^2 \leq \frac{1}{\tau^2} \|r^{(k)}\|^2$. This finally gives us the desired claim

$$\begin{aligned} \langle r^{(k)}, d^{(k)} \rangle &= \langle r^{(k)}, r^{(k)} \rangle + \langle r^{(k)}, \varepsilon_1^{(k)} \rangle \geq \langle r^{(k)}, r^{(k)} \rangle - \|r^{(k)}\| \|\varepsilon_1^{(k)}\| \\ &= \|r^{(k)}\| (\|r^{(k)}\| - \|\varepsilon_1^{(k)}\|) \geq \|r^{(k)}\|^2 (1 - \frac{\delta_1}{2}) \geq \gamma \|r^{(k)}\| \|d^{(k)}\|. \end{aligned}$$

□

With this preparation at hand, we get the following convergence estimate.

Theorem 3.4 *Let the assumptions of* Lemma 3.3 *hold. For the truncation tolerance* ε_2 *set*

$$\|\varepsilon_2^{k+1}\| \leq \theta\mu(\lambda_{\max})^{-1}\|r^{(k)}\|, \tag{3.5}$$

with

$$\theta := \sqrt{1-\frac{\gamma^2}{2\kappa}}, \quad \gamma < 1, \quad \kappa > 1, \quad \mu < \theta^{-1} - 1. \tag{3.6}$$

Then, we have

$$\|e^{(k)}\|_A \leq [\theta(1+\mu)]^k \|e^{(0)}\|_A, \tag{3.7}$$

with the error reduction factor

$$\varrho := \theta(1+\mu) < 1.$$

Proof Without loss of generality, we can assume the solution is at the origin $x^* = 0$ and thus $b = 0$. Since $d^{(k)}$ is a descent direction by Lemma 3.3, [22, Lemma 6.2.2] yields $f(x^{(k+1)}) \leq f(x^{(k)}) - \frac{\gamma^2}{4\lambda_{\max}}\|r^{(k)}\|^2$. Using an eigenbasis of A as in Eq. 3.2, we get

$$f(x^{(k)}) = \frac{1}{2}\langle x^{(k)}, Ax^{(k)}\rangle = \frac{1}{2}\sum_{j=1}^{N}\lambda_j c_j^2, \quad \|r^{(k)}\|^2 = \langle Ax^{(k)}, Ax^{(k)}\rangle = \sum_{j=1}^{N}\lambda_j^2 c_j^2.$$

This gives

$$f(x^{(k+1)}) \leq \frac{1}{2}\sum_{j=1}^{N}\lambda_j c_j^2(1-\frac{\gamma^2}{2\lambda_{\max}}\lambda_j) \leq (1-\frac{\gamma^2\lambda_{\min}}{2\lambda_{\max}})\frac{1}{2}\sum_{j=1}^{N}\lambda_j c_j^2 = (1-\frac{\gamma^2}{2\kappa})f(x^{(k)}).$$

The identity $2f(x) = \|x\|_A^2$ gives the desired claim for θ as in Eq. 3.6. Finally, we get with Eq. 3.5

$$\begin{aligned}\|e^{(k+1)}\|_A &\leq \theta\|e^{(k)}\|_A + \|\varepsilon_2^{(k+1)}\|_A,\\ &\leq \theta\|e^{(k)}\|_A + \theta\mu\|e^{(k)}\|_A,\\ &= \varrho\|e^{(k)}\|_A.\end{aligned}$$

This completes the proof. □

Remark 3.5 Note that the rate in Eq. 3.6 is *asymptotically* the same as in Proposition 3.1 for large κ. This is not surprising, since we used the same approach for analyzing the convergence as in the gradient descent method. Of course, Eq. 3.6 is more pessimistic, since it applies to a generalized descent method.

The preceding analysis is a worst-case scenario that guarantees convergence of the method with a monotonic decrease of the error in the energy norm. However, numerically, the perturbed CG performs far better than the gradient descent method. This is due to the fact that the perturbed CG inherits some nice properties of its exact counterpart, as can be seen in the following lemma. Moreover, the analysis in Theorem 3.4 is quite general, since we only require *local optimality* (i.e., a descent direction) and the resulting bound in Eq. 3.7 is thus by no means optimal.

Note, that according to Eq. 3.7, the truncation tolerance $\|\varepsilon_2^{(k+1)}\|$ should be set proportional to $\theta\|e^{(k)}\|_A$. However, since the error reduction factor θ corresponds to a worst-case scenario, this tolerance might be unnecessarily prohibitive and significantly hamper quantitative performance.

A more detailed look on the estimates from [22, Lemma 6.2.2] reveals $f(x^{(k+1)}) \leq \frac{1}{2}\sum_{j=1}^{N}\lambda_j c_j^2 - \alpha_k\langle r^{(k)}, d^{(k)}\rangle$, which suggests to choose an adaptive tolerance proportional to $\alpha_k\|d^{(k)}\|$. This is precisely the case for the adaptive tolerance strategy in [27, Algorithm 2]. Hence, we use this in our subsequent numerical experiments.

Lemma 3.6 *For the perturbed CG method we have the following representations*

$$r^{(k)} = (I - A\,p^{(k)}(A))r^{(0)} - A\left(\sum_{j=0}^{k-1} q_{k-j-1}^{(k)}(A)\varepsilon_1^{(j)} + A\sum_{j=1}^{k} g_{k-j}^{(k)}(A)\varepsilon_2^{(j)}\right),$$

$$e^{(k)} = (I - A\,p^{(k)}(A))e^{(0)} - \left(\sum_{j=0}^{k-1} q_{k-j-1}^{(k)}(A)\varepsilon_1^{(j)} + \sum_{j=1}^{k} g_{k-j}^{(k)}(A)\varepsilon_2^{(j)}\right),$$

where $p^{(k)} \in \mathcal{P}_{k-1}$, *i.e., a polynomial of degree* $k-1$, $g_j^{(k)} \in \mathcal{P}_j$ *with* $g_j^{(k)}(0) = 1$, $j = 0, \ldots, k-1$, *and* $q_j^{(k)} \in \mathcal{P}_j$ *such that* $p^{(k)}(t) = \sum_{j=0}^{k-1} q_j^{(k)}(t)$.

Proof We prove the assertion by induction over k. For $k = 1$, we have $x^{(1)} = x^{(0)} + \alpha_0(r^{(0)} + \varepsilon_1^{(0)}) + \varepsilon_2^{(1)} = x^{(0)} + \alpha_0 A r^{(0)} + \alpha_0\varepsilon_1^{(0)} + \varepsilon_2^{(1)}$. As a consequence, $r^{(1)} = b - Ax^{(1)} = (I - \alpha_0 A)r^{(0)} - \alpha_0 A\varepsilon_1^{(0)} - A\varepsilon_2^{(1)}$ and

$$\begin{aligned} d^{(1)} &= r^{(1)} + \beta_0 d^{(0)} + \varepsilon_1^{(1)} = (I - \alpha_0 A)r^{(0)} - \alpha_0 A\varepsilon_1^{(0)} - A\varepsilon_2^{(1)} + \beta_0(r^{(0)} + \varepsilon_1^{(0)}) + \varepsilon_1^{(1)} \\ &= (I + \beta_0 I - \alpha_0 A)r^{(0)} + (\beta_0 I - \alpha_0 A)\varepsilon_1^{(0)} + \varepsilon_1^{(1)} - A\varepsilon_2^{(1)}, \end{aligned}$$

from which the assertion follows for $k = 1$. Now, let the claim hold for some $k \geq 1$, then, we get by induction that

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \alpha_k d^{(k)} + \varepsilon_2^{(k+1)}, \\ &= x^{(0)} + p^{(k)}(A)r^{(0)} + \sum_{j=0}^{k-1} q_{k-j-1}^{(k)}(A)\varepsilon_1^{(j)} + \sum_{j=1}^{k} g_{k-j}^{(k)}(A)\varepsilon_2^{(j)} \\ &\quad + \alpha_k\left(\tilde{p}^{(k)}(A)r^{(0)} + \sum_{j=0}^{k} \tilde{q}_{k-j}^{(k)}(A)\varepsilon_1^{(j)} + A\sum_{j=1}^{k} \tilde{g}_{k-j}^{(k)}(A)\varepsilon_2^{(j)}\right) + \varepsilon_2^{(k+1)} \\ &= x^{(0)} + p^{(k+1)}(A)r^{(0)} + \sum_{j=0}^{k} q_{k-j}^{(k+1)}(A)\varepsilon_1^{(j)} + \sum_{j=1}^{k+1} g_{k-j+1}^{(k+1)}(A)\varepsilon_2^{(j)}, \end{aligned}$$

with $p^{(k+1)} := p^{(k)} + \alpha_k\tilde{p}^{(k)}$, $q_j^{(k+1)} := q_j^{(k)} + \alpha_k\tilde{q}_j^{(k)}$ for $j < k$ and $q_k^{(k+1)} := \alpha_k\tilde{q}_k^{(k)}$ as well as $g_j^{(k+1)} := g_j^{(k)} + \alpha_k t\tilde{g}_{j-1}^{(k)}$ for $j > 0$ and $g_0^{(k+1)} := 1$. Note that the

properties stated in this Lemma hold for the polynomials $p^{(k+1)}$, $q_j^{(k+1)}$, and $g_j^{(k+1)}$. Finally,

$$\begin{aligned} d^{(k+1)} &= r^{(k+1)} + \beta_k d^{(k)} + \varepsilon_1^{(k+1)} \\ &= (I - A\,p^{(k)}(A))r^{(0)} - A\sum_{j=0}^{k} q_{k-j}^{(k)}(A)\varepsilon_1^{(j)} - A\sum_{j=1}^{k+1} g_{k-j+1}^{(k)}(A)\varepsilon_2^{(j)} \\ &\quad +\beta_k\left(\tilde{p}^{(k)}(A)r^{(0)} + \sum_{j=0}^{k}\tilde{q}_{k-j}^{(k)}(A)\varepsilon_1^{(j)} + A\sum_{j=1}^{k}\tilde{g}_{k-j}^{(k)}(A)\varepsilon_2^{(j)}\right) + \varepsilon_1^{(k+1)} \\ &= (I - Ap^{(k)}(A) + \beta_k\tilde{p}^{(k)}(A))r^{(0)} + \varepsilon_1^{(k)} - A\sum_{j=0}^{k} q_{k-j}^{(k)}(A)\varepsilon_1^{(j)} \\ &\quad +\beta_k\sum_{j=0}^{k}\tilde{q}_{k-j}^{(k)}(A)\varepsilon_1^{(j)} - A\varepsilon_2^{(k+1)} - A\sum_{j=1}^{k}(g_{k-j+1}^{(k)}(A) - \beta_k\tilde{g}_{k-j}^{(k)}(A))\varepsilon_2^{(j)}, \end{aligned}$$

which completes the proof. □

Similar to its exact counterpart, the perturbed (P)CG is thus a polynomial method both in the initial residual and in the perturbations. It is easy to see that the polynomials $\{p^{(j)}\}_j$ are *not* orthogonal w.r.t. the discrete inner product $\langle p, q\rangle_{\mathrm{CG}} := \langle p(A)r^{(0)}, q(A)r^{(0)}\rangle$, see [16, Example 2.4.8]. Consequently the resulting iterates do not minimize $\langle p^{(k)}, t^{-1}p^{(k)}\rangle_{\mathrm{CG}} = \|e^{(k)}\|_A^2$.

Though the perturbed CG is a straightforward extension of its exact counterpart, the iterates are not characterized by minimization problems anymore. Thus, the typical notions of optimality or orthogonality of Krylov methods are lost. Moreover, we lose information about "how far" these inexact iterates are from the optimal iterates. This is due to the fact that nothing specific about the direction is required in the truncation step, other than it has to satisfy an error bound. Of course, numerically perturbed (P)CG still converges faster than perturbed gradient descent. However, we believe a rigorous proof of this would require more knowledge about the truncation step and thus most likely a modification of the algorithm itself.

4 HTucker-adaptive wavelet Galerkin method

As already said earlier, the new HTucker-adaptive wavelet Galerkin method (HTAWGM) relies on the strategy

$$\cdots \rightarrow \textbf{SOLVE} \rightarrow \textbf{ESTIMATE} \rightarrow \textbf{MARK and REFINE} \rightarrow \cdots$$

which is analogous to an adaptive FEM solver. We detail the ingredients as follows.

4.1 SOLVE

We use a Galerkin solver based on the CG iterations described in Section 3.2 with the approximate separable preconditioning from [3], see Section 2.3. The arising procedure is referred to as

$$\mathbf{PCG}(\boldsymbol{S}^{-1}(\delta), \boldsymbol{A}^{\delta}, \boldsymbol{f}^{\delta}, \boldsymbol{u}^{(0)}, \Lambda, \varepsilon),$$

where $\boldsymbol{S}^{-1}(\delta)$ is the preconditioning operator, $\boldsymbol{A}^{\delta}$ is the discrete (infinite dimensional) operator, $\boldsymbol{f}^{\delta}$ is the right hand side, $\boldsymbol{u}^{(0)}$ is the initial guess, Λ is a finite index set on which the iterations are performed, and ε is the residual tolerance.

Remark 4.1 We shall assume a separable structure for the operator A and thus will not discuss the approximation of more general operators (for this see, e.g., [1]). Thus, evaluating $\boldsymbol{A}_{\Lambda}\boldsymbol{u}$ on a finite set Λ boils down to applying the low-dimensional components of $\boldsymbol{A}_{\Lambda}$ to the leaves of $\boldsymbol{u}$. For the low-dimensional evaluation, we use the evaluation procedures from [23, Chapter 6].

4.2 ESTIMATE

For this step, we need a procedure for approximate residual evaluation. This requires determining an extended index set $\tilde{\Lambda} \supset \Lambda$ based on a desired tolerance $\varepsilon > 0$ and evaluating

$$\| R_{\tilde{\Lambda}} \left(\boldsymbol{f}^{\delta} - \boldsymbol{A}^{\delta} E_{\Lambda} \boldsymbol{u}_{\Lambda} \right) \|.$$

Again, due to the separable structure of $\boldsymbol{A}$, we only need to build $\tilde{\Lambda} = \tilde{\Lambda}_1 \times \cdots \times \tilde{\Lambda}_d$ from the low-dimensional components $\tilde{\Lambda}_j$, $j = 1, \ldots, d$. For this purpose, we use the method from [23, Chapter 7]. Additionally, we need to approximate scaling $\boldsymbol{S}^{-1}(\delta)$, which we discuss in detail later. We refer to this procedure as

$$\mathbf{RES}(\boldsymbol{S}^{-1}(\delta), \boldsymbol{A}^{\delta}, \boldsymbol{f}^{\delta}, \boldsymbol{u}^{\delta}, \varepsilon),$$

where ε refers to the relative accuracy in the sense that

$$\| \left(\boldsymbol{f}^{\delta} - \boldsymbol{A}^{\delta} E_{\Lambda} \boldsymbol{u}_{\Lambda} \right) - \tilde{\boldsymbol{r}} \| \leq \varepsilon \| \tilde{\boldsymbol{r}} \|,$$

and $\tilde{\boldsymbol{r}}$ is the approximate residual.

4.3 MARK and REFINE

In AFEM, one first marks certain elements, which are then refined by a chosen strategy: **REFINE**. In AWGM, these steps are performed together. The current index set Λ is extended, which drives the adaptivity of the algorithm. We use a standard bulk chasing strategy with a parameter $\alpha \in (0, 1)$, described as follows. Suppose the current approximation $\boldsymbol{u}$ is supported on Λ, then we determine a (minimal) set $\tilde{\Lambda} \supset \Lambda$ on which the approximate residual evaluation is performed. Then, we compute an intermediate set $\bar{\Lambda}$ with $\Lambda \subset \bar{\Lambda} \subset \tilde{\Lambda}$ such that

$$\| R_{\bar{\Lambda}} \boldsymbol{r} \| \geq \alpha \| \boldsymbol{r} \|, \tag{4.1}$$

where $\boldsymbol{r}$ is the approximate residual supported on $\tilde{\Lambda}$.

In a low-dimensional setting, Eq. 4.1 is realized by an approximate sorting of the entries in $\boldsymbol{r}$ and forming $\bar{\Lambda}$ by the minimal number of largest entries that satisfy (4.1). Such an approach is clearly not feasible for large dimensions $d \gg 1$.

In the tensor setting, we can only use low-dimensional quantities and thus determine $\bar{\Lambda}$ by sorting the contractions $\pi_j(\boldsymbol{r})$ using the **COARSE** routine from the low-dimensional setting, where $\mathbf{COARSE}(\boldsymbol{u}, \varepsilon)$ returns a tensor $\boldsymbol{v}$ with $\|\boldsymbol{u} - \boldsymbol{v}\| \leq \varepsilon$. We refer to the resulting procedure as

$$\mathbf{EXPAND}(\Lambda, \boldsymbol{r}, \alpha).$$

4.4 HT-AWGM algorithm

We now have all algorithmic ingredients at hand to describe a general AWGM procedure based on a tensor format in Algorithm 3. We use the notation $\mathcal{C}(\boldsymbol{u}, \varepsilon)$ to denote $\mathbf{COARSE}(\boldsymbol{u}, \varepsilon)$; $\mathcal{T}(\boldsymbol{u}, \varepsilon)$ to denote truncation and

$$\|(\boldsymbol{A}^\delta)^{-1}\| \leq (\lambda_{\min})^{-1}, \quad \|\boldsymbol{A}^\delta\| \leq \lambda_{\max}$$

Algorithm 3 HT-AWGM.

Input: Tolerance $\varepsilon > 0$, initial finite index set $\Lambda^{(0,0)} \neq \emptyset$, $\delta > 0$, $\alpha \in (0, 1)$, $\omega_0, \omega_1, \omega_2, \omega_3, \omega_4, \omega_5 > 0$, $M \in \mathbb{N}$.

$\boldsymbol{u}^{(0,0)} \leftarrow 0$, $\boldsymbol{r}^{(0,0)} \leftarrow \omega_0$, $\omega_0^{(0)} \leftarrow \omega_0$
for $k = 0, \ldots$ **do**
  for $m = 0, \ldots, M$ **do**
    $\boldsymbol{u}^{(k,m+1)} \leftarrow \mathbf{PCG}(\boldsymbol{S}^{-1}(\delta), \boldsymbol{A}^\delta, \boldsymbol{f}^\delta, \boldsymbol{u}^{(k,m)}, \Lambda^{(k,m)}, \omega_2\|\boldsymbol{r}^{(k,m)}\|)$
    $\boldsymbol{r}^{(k,m+1)} \leftarrow \mathbf{RES}(\boldsymbol{S}^{-1}(\delta), \boldsymbol{A}^\delta, \boldsymbol{f}^\delta, \boldsymbol{u}^{(k,m+1)}, \omega_1)$
    if $(1 + \omega_1)\|\boldsymbol{r}^{(k,m+1)}\| \leq \varepsilon$ **then**
      return $\boldsymbol{u}_\varepsilon \leftarrow \boldsymbol{u}^{(k,m+1)}$
    end if
    if $(1 + \omega_1)\|\boldsymbol{r}^{(k,m+1)}\| \leq \omega_3\omega_0^{(k)}$, or $m = M$ **then**
      $\boldsymbol{u}^{(k+1,0)} \leftarrow \mathcal{T}(\boldsymbol{u}^{(k,m+1)}, \omega_4\lambda_{\min}^{-1}\omega_0^{(k)})$
      $\boldsymbol{u}^{(k+1,0)} \leftarrow \mathcal{C}(\boldsymbol{u}^{(k+1,0)}, \omega_5\lambda_{\min}^{-1}\omega_0^{(k)})$
      $\Lambda^{(k+1,0)} \leftarrow \mathrm{supp}(\boldsymbol{u}^{(k+1,0)})$
      $\boldsymbol{r}^{(k+1,0)} \leftarrow \mathbf{RES}(\boldsymbol{S}^{-1}(\delta), \boldsymbol{A}^\delta, \boldsymbol{f}^\delta, \boldsymbol{u}^{(k+1,0)}, \omega_1)$
      $\omega_0^{(k+1)} \leftarrow (\omega_3 + \omega_4 + \omega_5)\omega_0^{(k)}$
      break
    end if
    $\Lambda^{(k,m+1)} \leftarrow \mathbf{EXPAND}(\Lambda^{(k,m)}, \boldsymbol{r}^{(k,m+1)}, \alpha)$
  end for
end for

The involved parameters have the following meaning:

- ω_0 is the initial estimate for the right hand side, i.e., $\omega_0 \geq \|\boldsymbol{f}^\delta\|$,

- ω_1 is the relative precision of the residual evaluation,
- ω_2 drives the tolerance for the approximate Galerkin solutions,
- ω_3 is the required error reduction rate before truncation and coarsening,
- ω_4 drives the truncation tolerance that controls rank growth,
- ω_5 drives the coarsening tolerance that controls index set growth and influences rank growth by controlling the maximum wavelet level,
- α is the bulk criterion parameter that drives adaptivity.

4.5 Convergence of HT-AWGM

We start proving the convergence of the algorithm by investigating the approximate residual evaluation. Two types of approximation are involved for the operator: (a) the finite index set approximation of $\boldsymbol{A}$ and (b) the approximation of the exact diagonal scaling $\boldsymbol{D}^{-1}$, resp. $\boldsymbol{S}^{-1}(\delta)$.

Lemma 4.2 *Let $\boldsymbol{v}$ be finitely supported and let $\boldsymbol{A}_\varepsilon$ denote an approximation to $\boldsymbol{A}$ in the sense that $\left\|\boldsymbol{D}^{-1}(\boldsymbol{A}-\boldsymbol{A}_\varepsilon)\boldsymbol{D}^{-1}\boldsymbol{v}\right\| \le \varepsilon$. Moreover, let $\boldsymbol{f}_\varepsilon$ be an approximation to $\boldsymbol{f}$ such that $\left\|\boldsymbol{D}^{-1}(\boldsymbol{f}-\boldsymbol{f}_\varepsilon)\right\| \le \varepsilon$. Finally, assume that $\|\boldsymbol{S}^{-1}(\delta)\boldsymbol{f}_\varepsilon\| \le C_f\|\boldsymbol{f}^\delta\|$ for all $\varepsilon > 0$ with $C_f \ge 1$. Then,*

$$\begin{aligned}&\left\|(\boldsymbol{f}^\delta - \boldsymbol{A}^\delta\boldsymbol{v}) - \boldsymbol{S}^{-1}(\delta,\eta)(\boldsymbol{f}_\varepsilon - \boldsymbol{A}_\varepsilon\boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{v})\right\| \\ &\le \varepsilon(1+\delta)(2+\delta) + \frac{\eta}{1-\delta}\left(C_f\|\boldsymbol{f}^\delta\| + 2\|\boldsymbol{A}^\delta\|\|\boldsymbol{v}\|\right) + 2\frac{(1+\delta)^2}{1-\delta}\eta\varepsilon, \quad (4.2)\end{aligned}$$

with $\eta, \delta > 0$.

Proof We begin by splitting the left-hand side of Eq. 4.2 into two parts

$$\begin{aligned}&\left\|(\boldsymbol{f}^\delta - \boldsymbol{A}^\delta\boldsymbol{v}) - \boldsymbol{S}^{-1}(\delta,\eta)(\boldsymbol{f}_\varepsilon - \boldsymbol{A}_\varepsilon\boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{v})\right\| \\ &\le \underbrace{\left\|\boldsymbol{f}^\delta - \boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{f}_\varepsilon\right\|}_{=:\,\text{(I)}} + \underbrace{\left\|\boldsymbol{A}^\delta\boldsymbol{v} - \boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{A}_\varepsilon\boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{v}\right\|}_{=:\,\text{(II)}}.\end{aligned}$$

We further split (I) as

$$\left\|\boldsymbol{f}^\delta - \boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{f}_\varepsilon\right\| \le \|\boldsymbol{S}^{-1}(\delta)(\boldsymbol{f}-\boldsymbol{f}_\varepsilon)\| + \|(\boldsymbol{S}^{-1}(\delta) - \boldsymbol{S}^{-1}(\delta,\eta))\boldsymbol{f}_\varepsilon\|$$

and get the first part $\|\boldsymbol{S}^{-1}(\delta)(\boldsymbol{f}-\boldsymbol{f}_\varepsilon)\| = \|\boldsymbol{S}^{-1}(\delta)\boldsymbol{D}\boldsymbol{D}^{-1}(\boldsymbol{f}-\boldsymbol{f}_\varepsilon)\| \le (1+\delta)\varepsilon$, where the last inequality follows from the property $\|\boldsymbol{S}^{-1}(\delta)\boldsymbol{D}\| \le 1+\delta$. For the second part in (I), we get

$$\begin{aligned}\|(\boldsymbol{S}^{-1}(\delta) - \boldsymbol{S}^{-1}(\delta,\eta))\boldsymbol{f}_\varepsilon\| &= \|(\boldsymbol{S}^{-1}(\delta) - \boldsymbol{S}^{-1}(\delta,\eta))\boldsymbol{S}(\delta)\boldsymbol{S}^{-1}(\delta)\boldsymbol{f}_\varepsilon\| \\ &\le \tfrac{\eta}{1-\delta}\|\boldsymbol{S}^{-1}(\delta)\boldsymbol{f}_\varepsilon\| \le C_f\tfrac{\eta}{1-\delta}\|\boldsymbol{f}^\delta\|,\end{aligned}$$

where we used the fact $\|(\boldsymbol{S}^{-1}(\delta) - \boldsymbol{S}^{-1}(\delta,\eta))\boldsymbol{S}(\delta)\| \leq \frac{\eta}{1-\delta}$. In a similar fashion, we split (II) into 2 parts

$$\begin{aligned}(\text{II}) \;&\leq \underbrace{\|\boldsymbol{S}^{-1}(\delta)(\boldsymbol{A}-\boldsymbol{A}_\varepsilon)\boldsymbol{S}^{-1}(\delta)\boldsymbol{v}\|}_{=:(\text{II.1})} \\ &\quad + \underbrace{\|\boldsymbol{S}^{-1}(\delta)\boldsymbol{A}_\varepsilon\boldsymbol{S}^{-1}(\delta)\boldsymbol{v} - \boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{A}_\varepsilon\boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{v}\|}_{=:(\text{II.2})},\end{aligned}$$

and follow the proof of [3, Proposition 15]. For the first term, we get

$$(\text{II.1}) = \|[\boldsymbol{S}^{-1}(\delta)\boldsymbol{D}]\boldsymbol{D}^{-1}(\boldsymbol{A}-\boldsymbol{A}_\varepsilon)\boldsymbol{D}^{-1}[\boldsymbol{D}\boldsymbol{S}^{-1}(\delta)]\boldsymbol{v}\| \leq (1+\delta)^2\varepsilon,$$

where we used Eq. 2.9b. The second term (II.2) involves the approximation errors $\|\boldsymbol{S}(\delta)(\boldsymbol{S}^{-1}(\delta) - \boldsymbol{S}^{-1}(\delta,\eta))\boldsymbol{v}\|$ and $\|\boldsymbol{S}^{-1}(\delta)(\boldsymbol{A}-\boldsymbol{A}_\varepsilon)\boldsymbol{S}^{-1}(\delta)\boldsymbol{v}\|$. For the former, we use Eq. 2.9f and the latter can be bounded by $(1+\delta)^2\varepsilon$ as in (II.1). Altogether we get

$$(\text{II.2}) \leq \tfrac{2\eta}{1-\delta}(\|\boldsymbol{A}^\delta\|\|\boldsymbol{v}\| + (1+\delta)^2\varepsilon),$$

which completes the proof. □

For a given tolerance $tol > 0$ and a finite tensor $\boldsymbol{v}$, we can specify ε and η as

$$\varepsilon \leq \frac{tol}{3(1+\delta)(2+\delta)}, \quad \eta \leq \min\left\{\frac{1-\delta}{2}, \frac{tol(1-\delta)}{3(C_f\|\boldsymbol{f}^\delta\| + 2\|\boldsymbol{A}^\delta\|\|\boldsymbol{v}\|)}\right\}.$$

By Eq. 4.2, this would ensure

$$\|(\boldsymbol{f}^\delta - \boldsymbol{A}^\delta\boldsymbol{v}) - \boldsymbol{S}^{-1}(\delta,\eta)(\boldsymbol{f}_\varepsilon - \boldsymbol{A}_\varepsilon S^{-1}(\delta,\eta)\boldsymbol{v})\| \leq tol.$$

As a consequence, given the parameter $\omega_1 \in (0,1)$ from Algorithm 3 and some fixed $\delta > 0$, we can now use Eq. 4.2 to ensure

$$\|(\boldsymbol{f}^\delta - \boldsymbol{A}^\delta\boldsymbol{v}) - \boldsymbol{S}^{-1}(\delta,\eta)(\boldsymbol{f}_\varepsilon - \boldsymbol{A}_\varepsilon S^{-1}(\delta,\eta)\boldsymbol{v})\| \leq \omega_1\|\boldsymbol{S}^{-1}(\delta,\eta)(\boldsymbol{f}_\varepsilon - \boldsymbol{A}_\varepsilon S^{-1}(\delta,\eta)\boldsymbol{v})\|. \quad (4.3)$$

With all the above ingredients at hand, it is now easy to prove that Algorithm 3 converges for an appropriate choice of parameters. There are two main components. First, we choose ω_1, ω_2, and α appropriately such that we ensure in each inner iteration $m \to m+1$ of Algorithm 3 a guaranteed error reduction. Second, we choose ω_3, ω_4, and ω_5 such that after truncation and coarsening we still ensure an error reduction for the outer iteration $k \to k+1$.

We use the notation

$$\|\cdot\|_A := \langle\cdot, \boldsymbol{A}^\delta\cdot\rangle,$$

to denote the energy norm.

Proposition 4.3 *Let* $\Lambda^{(0)} = \Lambda_1^{(0)} \times \cdots \times \Lambda_d^{(0)}$ *and all* $\Lambda_j^{(0)}$ *are assumed to have a tree structure*[6] *as required in* [23, Chapter 6]. *Let the parameters satisfy* $0 < \omega_1 < \alpha$ *and*

$$\omega_2 < \frac{(1-\omega_1)(\alpha+\omega_1)}{1+\omega_1}\kappa(\boldsymbol{A}^\delta)^{-1}.$$

This guarantees an error reduction in the inner iterations

$$\|\boldsymbol{u} - \boldsymbol{u}^{(k,m+1)}\|_A \le \vartheta \|\boldsymbol{u} - \boldsymbol{u}^{(k,m)}\|_A, \tag{4.4}$$

with

$$\vartheta := \left(1 - \left(\frac{\alpha-\omega_1}{1+\omega_1}\right)^2 \kappa^{-1}(\boldsymbol{A}^\delta) + \left(\frac{\omega_2}{1-\omega_1}\right)^2 \kappa(\boldsymbol{A}^\delta)\right)^{1/2} < 1 \tag{4.5}$$

Moreover, if $M \in \mathbb{N}$ *is chosen such that*

$$M \ge M^* = M^*(\delta) := \left\lceil \left| \frac{\ln(\omega_3[\kappa(\boldsymbol{A}^\delta)]^{-1/2})}{\ln(\vartheta)} \right| \right\rceil, \tag{4.6}$$

and

$$\omega_3 + \omega_4 + \omega_5 < 1, \tag{4.7}$$

then the error decreases in each outer iteration such that

$$\|\boldsymbol{u} - \boldsymbol{u}^{(k,0)}\| \le \lambda_{\min}^{-1}\omega_0(\omega_3+\omega_4+\omega_5)^k. \tag{4.8}$$

This ensures Algorithm 3 *terminates after at most* K^*M^* *steps, where*

$$K^* = K^*(\varepsilon,\delta) := \left\lceil \left| \frac{\ln([\varepsilon\kappa(\boldsymbol{A}^\delta)\omega_3\omega_0(1+\omega_1)]^{-1}(1-\omega_1))}{\ln(\omega_3+\omega_4+\omega_5)} \right| \right\rceil, \tag{4.9}$$

with the output satisfying

$$\|\boldsymbol{f}^\delta - \boldsymbol{A}^\delta \boldsymbol{u}_\varepsilon\| \le \varepsilon.$$

Proof The statement in Eq. 4.4 with ϑ as in Eq. 4.5 is an immediate consequence of the assumptions and an application of [35, Prop. 4.2]. The conditions on α, ω_1 and ω_2 ensure $0 < \vartheta < 1$.

In the inner iterations, we thus get for any k

$$\begin{aligned}\|\boldsymbol{u} - \boldsymbol{u}^{(k,m)}\| &\le \lambda_{\min}^{-1/2}\|\boldsymbol{u} - \boldsymbol{u}^{(k,m)}\|_A \le \lambda_{\min}^{-1/2}\vartheta^m\|\boldsymbol{u} - \boldsymbol{u}^{(k,0)}\|_A,\\ &\le \sqrt{\kappa(\boldsymbol{A}^\delta)}\vartheta^m\|\boldsymbol{u} - \boldsymbol{u}^{(k,0)}\| \le \sqrt{\kappa(\boldsymbol{A}^\delta)}\vartheta^m\lambda_{\min}^{-1}\omega_0^{(k)}.\end{aligned}$$

The requirement (4.6) ensures

$$\|\boldsymbol{u} - \boldsymbol{u}^{(k,M)}\| \le \sqrt{\kappa(\boldsymbol{A}^\delta)}\vartheta^M\lambda_{\min}^{-1}\omega_0^{(k)} \le \omega_3\lambda_{\min}^{-1}\omega_0^{(k)}. \tag{4.10}$$

[6]This allows the application of an *exact* **APPLY** routine from [23] on finite index sets with linear complexity. I.e., no additional error terms are introduced in the convergence estimate.

Alternatively, the first if-condition in line 9 ensures

$$\|\boldsymbol{u}-\boldsymbol{u}^{(k,m+1)}\| \le \lambda_{\min}^{-1}\|\boldsymbol{A}^{\delta}(\boldsymbol{u}-\boldsymbol{u}^{(k,m+1)})\| \le \lambda_{\min}^{-1}(1+\omega_1)\|\boldsymbol{r}^{(k,m+1)}\| \le \omega_3\lambda_{\min}^{-1}\omega_0^{(k)}.$$

Hence, after truncation and coarsening, we obtain

$$\begin{aligned}\|\boldsymbol{u}-\boldsymbol{u}^{(k+1,0)}\| &\le \|\boldsymbol{u}-\boldsymbol{u}^{(k,m+1)}\| + \|\boldsymbol{u}^{(k,m+1)} - \mathcal{T}(\boldsymbol{u}^{(k,m+1)}, \lambda_{\min}^{-1}\omega_4\omega_0^{(k)})\| \\ &\quad + \|\boldsymbol{u}^{k+1,0} - \mathcal{T}(\boldsymbol{u}^{(k,m+1)}, \lambda_{\min}^{-1}\omega_4\omega_0^{(k)})\| \\ &\le \lambda_{\min}^{-1}\omega_0^{(k)}(\omega_3+\omega_4+\omega_5) = \lambda_{\min}^{-1}\omega_0(\omega_3+\omega_4+\omega_5)^{k+1},\end{aligned}$$

which shows (4.8). Combining (4.10) and (4.8), we obtain (4.9). Together with (4.7), this completes the proof. □

4.6 Complexity

The complexity in rank and discretization is controlled by the intermediate truncation and coarsening steps in line 10 and 11 of Algorithm 3. This is done in analogy to the re-coarsening step in the non-tensor case as in, e.g., [8]; and to the tensor recompression and coarsening as in [1]. In [17], it was shown that an AWGM without re-coarsening is optimal for a moderate choice of α. Unfortunately, the same ideas do not carry over to the tensor case. For a detailed discussion, see Section 4.7.

In order to capture the optimal ranks and index set size w.r.t. $\boldsymbol{u}$, we must choose a truncation tolerance in line 10 and a coarsening tolerance in line 11 slightly above the error $\|\boldsymbol{u}-\boldsymbol{u}^{(k,m+1)}\|$. In addition, since in the tensor case we can only numerically realize quasi-optimal approximations w.r.t. rank and discretization, quasi-optimality constants from Eqs. 2.4 to 2.6 are involved.

Proposition 4.4 *Let $\boldsymbol{u}^{\delta} \in \mathcal{A}(\gamma)$ and $\pi_j(\boldsymbol{u}^{\delta}) \in \mathcal{A}_s$ for all $1 \le j \le d$. Assume the sequence γ is admissible*

$$\rho(\gamma) := \sup_{n\in\mathbb{N}} \frac{\gamma(n)}{\gamma(n-1)} < \infty.$$

Finally, let the parameters ω_4, ω_5 satisfy

$$\omega_4 > (\sqrt{2d-3})\omega_3, \quad \omega_5 > \sqrt{d}(1+\sqrt{2d-3})\omega_3. \tag{4.11}$$

Then, the following estimates hold

$$|r(\boldsymbol{u}^{(k,0)})|_{\infty} \le \gamma^{-1}\left(C_0(\omega_3+\omega_4+\omega_5)^{-k}\|\boldsymbol{u}^{\delta}\|_{\mathcal{A}(\gamma)}\right), \quad \|\boldsymbol{u}^{(k,0)}\| \le C_1\|\boldsymbol{u}^{\delta}\|_{\mathcal{A}(\gamma)},$$

$$\sum_{j=1}^{d} \#\operatorname{supp}_j(\boldsymbol{u}^{(k,0)}) \le C_2(\omega_3+\omega_4+\omega_5)^{-k/s}\left(\sum_{j=1}^{d}\|\pi_j(\boldsymbol{u}^{\delta})\|_{\mathcal{A}_s}\right)^{1/s},$$

$$\sum_{j=1}^{d}\|\pi_j(\boldsymbol{u}^{(k,0)})\|_{\mathcal{A}_s} \le C_3\sum_{j=1}^{d}\|\pi_j(\boldsymbol{u}^{\delta})\|_{\mathcal{A}_s},$$

with the constants

$$C_0 := \frac{\lambda_{\min}\sqrt{2d-3}}{\omega_0(\omega_4-\omega_3\sqrt{2d-3})}\rho(\gamma), \quad C_1 := 1+\frac{(\omega_3+\omega_4)\sqrt{2d-3}}{\omega_4-\omega_3\sqrt{2d-3}},$$
$$C_2 := 2d\left(\frac{\lambda_{\min}\omega_3\sqrt{2d-3}}{\omega_0(\omega_4-\omega_3\sqrt{2d-3})}\right)^{1/s},$$
$$C_3 := 2^s(1+3^s)+2^{4s}d^{\max(1,s)}\frac{1+\omega_4(\sqrt{2d-3}+\sqrt{d}(1+\sqrt{2d-3}))}{\omega_4-\omega_3\sqrt{2d-3}}.$$

Proof The proof is an application of [1, Thm. 7]. □

The complexity requirement (4.11) together with the convergence requirement (4.7) implies $\omega_3 < [1+\sqrt{2d-3}+\sqrt{d}(1+\sqrt{2d-3})]^{-1}$.

Proposition 4.4 ensures the outer iterates $\boldsymbol{u}^{(k,0)}$ to have quasi-optimal support size and ranks. We first demonstrate that the quasi-optimal support size is preserved by the inner iterates $\boldsymbol{u}^{(k,m)}$.

In the estimates following in this subsection, we require the basic assumption of efficient approximability of the right-hand side, i.e.,

$$\sum_{j=1}^{d}\#\pi_j(\boldsymbol{f}_\varepsilon^\delta) \le C\varepsilon^{-1/s}\left(\sum_{j=1}^{d}\|\pi_j(\boldsymbol{f}^\delta)\|_{\mathcal{A}_s}\right)^{1/s}, \quad \sum_{j=1}^{d}\|\pi_j(\boldsymbol{f}_\varepsilon^\delta)\|_{\mathcal{A}_s} \le C\sum_{j=1}^{d}\|\pi_j(\boldsymbol{f}^\delta)\|_{\mathcal{A}_s}, \quad (4.12)$$

for any $\varepsilon > 0$ and a constant $C > 0$ independent of ε.

Proposition 4.5 *Assume that the one-dimensional components of $\boldsymbol{A}$ are s^*-compressible. Let the assumptions of Proposition 4.4 hold for $0 < s < s^*$. Moreover, let the assumptions of Theorem 3.4 be satisfied. Then, the intermediate index sets satisfy*

$$\sum_{j=1}^{d}\#\Lambda_j^{(k,m)} \le C\|\boldsymbol{u}^\delta-\boldsymbol{u}^{(k,m)}\|^{-1/s}\left(\sum_{j=1}^{d}\|\pi_j(\boldsymbol{u}^\delta)\|_{\mathcal{A}_s}\right)^{1/s},$$

for a constant C independent of k and m.

Proof On iteration (k,m) we get the following.

1. Due to Theorem 3.4, we can ensure an upper bound on the number of **PCG** iterations. Let $\boldsymbol{r}_{CG}^i$ denote the inner **PCG** residual at **PCG** iteration i and $\boldsymbol{e}_{CG}^i$ the corresponding error. Then,

$$\|\boldsymbol{r}_{CG}^i\| \le \lambda_{\max}^{1/2}\|\boldsymbol{e}_{CG}^i\|_A \le \lambda_{\max}^{1/2}\varrho^i\|\boldsymbol{e}_{CG}^0\|_A \le \kappa^{1/2}\varrho^i\|\boldsymbol{r}^{(k,m)}\|_A \le \omega_2\|\boldsymbol{r}^{(k,m)}\|_A.$$

Thus, the number of **PCG** iterations is bounded by

$$i \le I^* := \left\lceil\left|\frac{\ln(\omega_2\kappa^{-1/2})}{\ln(\varrho)}\right|\right\rceil. \quad (4.13)$$

2. Applying the same proof as in [8, Prop. 6.7], together with the results from [1, Thm. 8] and Eq. 4.12, we obtain

$$\|\pi_j(\boldsymbol{u}^{(k,m+1)})\|_{\mathcal{A}_s} \leq C(I^*)\|\pi_j(\boldsymbol{u}^{(k,m)})\|_{\mathcal{A}_s},$$

for $1 \leq j \leq d$.

3. Applying once more [1, Thm. 8], Eq. 4.12, and the above

$$\begin{aligned}\|\pi_j(\boldsymbol{r}^{(k,m+1)})\|_{\mathcal{A}_s} &\leq C(\|\pi_j(\boldsymbol{f}^{\delta})\|_{\mathcal{A}_s} + \|\pi_j(\boldsymbol{u}^{(k,m+1)})\|_{\mathcal{A}_s}),\\ &\leq \tilde{C}(\|\pi_j(\boldsymbol{f}^{\delta})\|_{\mathcal{A}_s} + \|\pi_j(\boldsymbol{u}^{(k,m)})\|_{\mathcal{A}_s}),\\ &\leq \bar{C}(\|\pi_j(\boldsymbol{f}^{\delta})\|_{\mathcal{A}_s} + \|\pi_j(\boldsymbol{u}^{(k,0)})\|_{\mathcal{A}_s}),\end{aligned}$$

for $1 \leq j \leq d$ and a constant $\bar{C} > 0$ independent of k or m.

4. Let $\mathcal{C}(\boldsymbol{v}, N)$ denote the routine **COARSE** retaining N terms, i.e., $\sum_{j=1}^{d} \#\operatorname{supp}_j(\pi_j(\boldsymbol{v})) \leq N$. Let $\mathcal{C}^o(\boldsymbol{v}, N)$ denote the best N-term approximation over product sets, such that $\sum_{j=1}^{d} \#\operatorname{supp}_j(\pi_j(\boldsymbol{v})) \leq N$. For a given $\varepsilon > 0$, take N to be minimal such that

$$\|\boldsymbol{v} - \mathcal{C}^o(\boldsymbol{v}, N)\| \leq \varepsilon.$$

Then, by property (2.6)

$$\|\boldsymbol{v} - \mathcal{C}(\boldsymbol{v}, N)\| \leq \sqrt{d}\|\boldsymbol{v} - \mathcal{C}^o(\boldsymbol{v}, N)\| \leq \varepsilon.$$

Consequently

$$\min\{N : \|\boldsymbol{v} - \mathcal{C}(\boldsymbol{v}, N)\| \leq \varepsilon\} \leq \min\left\{N : \|\boldsymbol{v} - \mathcal{C}^o(\boldsymbol{v}, N)\| \leq \frac{\varepsilon}{\sqrt{d}}\right\}.$$

5. As shown in the proof of [1, Thm. 7], the best N-term approximation over product sets satisfies the property

$$\min\left\{N : \|\boldsymbol{v} - \mathcal{C}^o(\boldsymbol{v}, N)\| \leq \varepsilon\right\} \leq 2d\varepsilon^{-1/s}\left(\sum_{j=1}^{d}\|\pi_j(\boldsymbol{v})\|_{\mathcal{A}_s}\right)^{1/s}.$$

Combining 3.5. with Proposition 4.4, we get the desired claim

$$\begin{aligned}\sum_{j=1}^{d}\#\Lambda_j^{(k,m+1)} &\leq C(\sqrt{1-\alpha^2}\|\boldsymbol{r}^{(k,m+1)}\|)^{-1/s}\left(\sum_{j=1}^{d}\|\pi_j(\boldsymbol{f}^{\delta})\|_{\mathcal{A}_s} + \|\pi_j(\boldsymbol{u}^{(k,0)})\|_{\mathcal{A}_s}\right)^{1/s}\\ &\leq \tilde{C}\|\boldsymbol{u}^{\delta} - \boldsymbol{u}^{(k,m+1)}\|^{-1/s}\left(\sum_{j=1}^{d}\|\pi_j(\boldsymbol{u}^{\delta})\|_{\mathcal{A}_s}\right)^{1/s},\end{aligned}$$

with a constant $\tilde{C} > 0$ independent of k or m. This completes the proof. □

The maximum wavelet level appearing in $\Lambda^{(k,m)}$ influences the rank of the preconditioning $\boldsymbol{S}^{-1}(\delta, \eta)$. To show that intermediate ranks can only deteriorate in a controlled manner, we require the following lemma.

Lemma 4.6 *Let the assumptions of* Proposition 4.5 *be satisfied for* $0 < s < s^*$. *Additionally, assume the data* $\boldsymbol{f}$ *and operator* $\boldsymbol{A}$ *have excess regularity for some* $t > 0$

$$\|\boldsymbol{D}_j^{-1+t}\pi_j(\boldsymbol{f}_\varepsilon)\| \lesssim \|\boldsymbol{D}_j^{-1+t}\pi_j(\boldsymbol{f})\| < \infty, \quad \|\boldsymbol{D}_j^{-1+t}\boldsymbol{A}_j^{n_j}\boldsymbol{D}_j^{-1-t}\| < \infty, \tag{4.14}$$

for any $1 \le j \le d$, *any* $1 \le n_j \le R$, *and any* $\varepsilon > 0$ *(see* Eq. 2.10*). Essentially* (4.14) *requires the one-dimensional components* f *to have regularity* H^{-1+t} *and the one-dimensional wavelet basis to have regularity* H^{1+t}, *which in turn ensures a slightly faster decay of the wavelet coefficients.*

Then, on iteration (k, m) *the maximum level arising in* $\Lambda^{(k,m)}$ *can be bounded by*

$$t^{-1}\log_2\left(C^{kM^*I^*+m}\|\boldsymbol{u}^\delta - \boldsymbol{u}^{(k,m)}\|^{-1-1/2s}\max_j\|\boldsymbol{D}_j^t\boldsymbol{f}^\delta\|\left(\sum_{j=1}^d\|\pi_j(\boldsymbol{u}^\delta)\|_{\mathcal{A}_s}\right)^{1/2s}\right),$$

where $C > 0$ *is a constant independent of* k *and* m, *and* M^* *and* I^* *are defined in* Eqs. 4.6 *and* 4.13 *respectively.*

Proof We want to apply [3, Lemma 37], i.e., the maximum level depends on the decay of the wavelet coefficients and the size of the tensor. To this end, note that $\Lambda^{(k,0)}$ is obtained by coarsening $\boldsymbol{u}^{(k-1,m)}$ for an m that satisfies line 9 of Algorithm 3. Thus, we need to estimate $\|\boldsymbol{D}_j^t\boldsymbol{u}^{(k-1,m)}\|$ and the support size of $\boldsymbol{u}^{(k-1,m)}$. For the latter we apply Proposition 4.5.

For the former, we can apply [3, Prop. 39] together with assumption (4.14), since $\boldsymbol{u}^{(k-1,m)}$ is a polynomial in $\boldsymbol{f}^\delta$ (cf. Lemma 3.6) and excess regularity is stable under truncation or coarsening. This gives the desired claim for $\Lambda^{(k,0)}$.

The set $\Lambda^{(k,m)}$, $m > 1$, is obtained by coarsening the approximate residual $\boldsymbol{r}^{(k,m)}$. Thus, as above, we need to estimate $\|\boldsymbol{D}_j^t\boldsymbol{r}^{(k,m)}\|$ and the support size of $\boldsymbol{r}^{(k,m)}$. To this end, note that the approximate residual is of the form

$$\boldsymbol{r}^{(k,m)} = \boldsymbol{S}^{-1}(\delta,\eta_k)(\boldsymbol{f}_{\varepsilon_k} - \boldsymbol{A}_{\varepsilon_k}\boldsymbol{S}^{-1}(\delta,\eta_k)\boldsymbol{u}^{(k,m)}),$$

for ε_k and η_k chosen according to Lemma 4.2. Applying assumption (4.14) and [3, Prop. 39] to $\boldsymbol{u}^{(k,m)}$, we get

$$\|\boldsymbol{D}_j^t\boldsymbol{r}^{(k,m)}\| \le C^{kM^*I^*+m}\|\boldsymbol{D}_j^t\boldsymbol{f}^\delta\|,$$

for $C > 0$ independent of k or m.

For the support size of $\boldsymbol{r}^{(k,m)}$, we apply (4.12), the compressibility of $\boldsymbol{A}$ together with [1, Thm. 8], and Proposition 4.5. This gives

$$\sum_{j=1}^d \#\pi_j(\boldsymbol{r}^{(k,m)}) \le C\|\boldsymbol{u}^\delta - \boldsymbol{u}^{(k,m)}\|^{-1/s}\left(\sum_{j=1}^d\|\pi_j(\boldsymbol{u}^\delta)\|_{\mathcal{A}_s}\right)^{1/s},$$

and the desired claim follows by an application of [3, Lemma 37]. □

Finally, we demonstrate that intermediate ranks of the numerical solution depend on the approximation accuracy and ranks of the exact solution. In the following, $r(\boldsymbol{A})$

and $r(\boldsymbol{f})$ denote the (finite) ranks of the non-preconditioned operator and right-hand side.

Proposition 4.7 *Let the assumptions of* Proposition 4.5 *and* Lemma 4.6 *hold. Let* I^* *from* Eq. 4.13 *denote the bound on the number of* ***PCG*** *iterations. Then, we can bound the ranks of the arising intermediate iterates as*

$$|r(\boldsymbol{u}^{(k,m)})|_\infty \leq C|r(\boldsymbol{A})|_\infty^{mI^*}\left[1+|\ln(\|\boldsymbol{u}^\delta-\boldsymbol{u}^{(k,m)}\|)|+\ln\left(\sum_{j=1}^d\|\pi_j(\boldsymbol{u}^\delta)\|_{\mathcal{A}_s}\right)\right]^{2mI^*}\times$$
$$\times\left[\gamma^{-1}\left(C\frac{\|\boldsymbol{u}^\delta\|_{\mathcal{A}(\gamma)}}{\|\boldsymbol{u}^\delta-\boldsymbol{u}^{(k,m)}\|}\right)+|r(\boldsymbol{f})|_\infty\right]=:\hat{r},$$

for a constant $C>0$ *independent of* k *or* m.

Proof Applying Lemma 4.6 and [3, Theorem 34], we get for the rank of the preconditioner at step (k,m)

$$|r(\boldsymbol{S}^{-1}(\delta,\eta_{k,m}))|_\infty\leq C\left(1+|\ln(\|\boldsymbol{u}^\delta-\boldsymbol{u}^{(k,m)}\|)|+k+\ln\left(\sum_{j=1}^d\|\pi_j(\boldsymbol{u}^\delta)\|_{\mathcal{A}_s}\right)\right).$$

Using Proposition 4.3, k can be bounded by $1+|\ln(\|\boldsymbol{u}^\delta-\boldsymbol{u}^{(k,m)}\|)|$. Finally, since $\boldsymbol{u}^{(k,m)}$ is a polynomial in $\boldsymbol{f}^\delta$ and $\boldsymbol{u}^{(k,0)}$ (cf. Lemma 3.6) and together with Proposition 4.4, we get the desired claim. □

Corollary 4.8 *Under the assumptions of* Proposition 4.7*, the number of operations to produce the iterate* $\boldsymbol{u}^{(k,m)}$ *can be bounded as*

$$\mathcal{O}\left(\left[1+\left|\ln(\boldsymbol{\varepsilon}^{(k,m)})\right|\right]^{8(M^*+1)I^*}\left[1+\gamma^{-1}\left(C(\boldsymbol{\varepsilon}^{(k,m)})^{-1}\right)\right]^{4(M^*+1)I^*}(\boldsymbol{\varepsilon}^{(k,m)})^{-1/s}\right),$$

where $\boldsymbol{\varepsilon}^{(k,m)}:=\|\boldsymbol{u}^\delta-\boldsymbol{u}^{(k,m)}\|$ *and* $C>0$ *is independent of* $\boldsymbol{\varepsilon}^{(k,m)}$.

Proof The dominant part for the complexity estimate is truncation. For a finite tensor $\boldsymbol{v}$, the work for truncating is bounded by

$$d|r(\boldsymbol{v})|_\infty^4+|r(\boldsymbol{v})|_\infty^4\sum_{j=1}^d\#\pi_j(\boldsymbol{v})$$

Application of Proposition 4.7 yields the desired claim. □

Remark 4.9 A few remarks on Corollary 4.8 are in order.

1. The factor $\varepsilon^{-1/s}$ is the work related to the approximation of the frames of $\boldsymbol{u}^\delta$. It does not dominate the complexity estimate.
2. The factor $\gamma^{-1}(C\frac{\|\boldsymbol{u}^\delta\|_{\mathcal{A}(\gamma)}}{\varepsilon})$ reflects the low-rank approximability of $\boldsymbol{u}^\delta$. Unlike in standard AWGM methods, due to the heavy reliance on truncation techniques

to keep ranks small, we cannot expect the dependence on this factor to be linear but rather algebraic at best. To achieve linear complexity, if at all possible, would require a fundamentally different approach to approximate u.

3. The dimension dependence on $d \gg 1$ is hidden in the constants and the rank growth factor $\gamma^{-1}(C\frac{\|\boldsymbol{u}^\delta\|_{\mathcal{A}(\gamma)}}{\varepsilon})$. In particular, approximability of $\boldsymbol{f}$, $\boldsymbol{A}$, $\boldsymbol{u}^\delta$ and the behavior of $\kappa(\boldsymbol{A}^\delta)$ determine the overall amount of work w.r.t. d. E.g., in [3, Thm. 26], the authors assume γ to be exponential in the rank r and independent of d; the sparsity of frames of $\boldsymbol{f}$ to be independent of d and the overall support size of $\boldsymbol{f}$ to grow at most linearly in d; the excess regularity to be t, $\kappa(\boldsymbol{A}^\delta)$ and the ranks of $\boldsymbol{A}$ to be independent of d; the number of operations to compute $\boldsymbol{f}_\varepsilon$ to grow at most polynomially in d. With these assumptions, the authors show the number of required operations to compute $\boldsymbol{u}_\varepsilon$ to grow at most as $d^{C\ln(d)}|\ln(\varepsilon)|^{C\ln(d)}$ w.r.t. d. Here, $\ln(d)$ stems from the fact that the quasi-optimality of truncation and coarsening depends on d.

4.7 Discussion

For a long time, the question of optimality for classical adaptive methods remained open. In particular, it was unclear if adaptive algorithms recovered the minimal index set (of wavelets or finite elements) required for the current error, up to a constant. In [8], the authors showed for an elliptic problem solved via an adaptive wavelet Galerkin routine that indeed optimality can be achieved. Crucial for optimality was a re-coarsening step, as in line 11 of Algorithm 3. In [17], it was shown that optimality can be attained without a re-coarsening step by a careful choice of the bulk chasing parameter α. In [34], the results were extended to finite elements.

It was thus of interest for us to investigate if we can ensure index set optimality without the re-coarsening step in line 11 of Algorithm 3. By "optimality" we refer to the optimal *product* index set.

In short, this fails for the current form of the algorithm. On one hand, the choice of the bulk chasing parameter $0 < \alpha < 1$ is a delicate balance between optimality and convergence. In [17], it was shown that $\alpha < \kappa(\boldsymbol{A})^{-1/2}$ ensures optimality, while any choice $\alpha > 0$ ensures convergence.

On the other hand, by the nature of high-dimensional problems, if we want to avoid exponential scaling in d, we have to consider each $\boldsymbol{\Lambda}_j$ in the product $\boldsymbol{\Lambda}_1 \times \cdots \times \boldsymbol{\Lambda}_d$ separately. This leads to the necessity of aggregating information, as is done via the contractions in Eq. 2.5. Such aggregation means we can estimate magnitudes at best only up to a dimension dependent constant. Specifically, $\sqrt{d}$ in Eq. 2.6.

Thus, for a given $\alpha > 0$, computing the minimal index set would be of exponential complexity. Computing the minimal index set via contractions for a given α, we can show that the resulting set is optimal for an adjusted value of

$$\tilde{\alpha} := \sqrt{\frac{\alpha^2 + d - 1}{d}}.$$

For $d > 1$, this value is too close to 1 and cannot additionally satisfy $\tilde{\alpha} < \kappa(\boldsymbol{A})^{-1/2}$ for realistic values of $\kappa(\boldsymbol{A})$.

From a different perspective, suppose we use contractions to determine the index set in the first dimension only and then iterate this procedure over all dimensions. Choosing $\alpha < \kappa(\boldsymbol{A})^{-1/2}$ ensures the optimality of the resulting index sets. However, the final relative error is bounded by $\sqrt{d(1-\alpha^2)}$. Hence, for realistic $\kappa(\boldsymbol{A})$, we lose convergence. The range of values for optimality and convergence do not intersect since the additional constant $\sqrt{d}$ is larger than 1. This mismatch lies at the heart of the issue.

5 Numerical experiments

In this section, we test our implementation of **HT-AWGM** analyzed in the previous section. In particular, we are interested in the behavior of ranks and the discretization. We choose a simple model problem and vary the dimension d. We consider $-\Delta u = 1$ in $\Omega := (0,1)^d$, $u = 0$ on $\partial\Omega$ in its variational formulation of finding $u \in H_0^1(\Omega)$ such that $a(u,v) := \int_\Omega \langle \nabla u(x), \nabla v(x)\rangle dx = 1(v)$ for all $v \in H_0^1(\Omega)$. The corresponding operator is given by $A : H_0^1(\Omega) \to H^{-1}(\Omega)$, where $A(u) := a(u,\cdot)$, which is boundedly invertible and self-adjoint.

For the discretization, we use tensor products of L_2-orthonormal piecewise polynomial cubic B-spline multiwavelets. We use our own implementation of an HTucker library. All of the software is implemented in C++. For more details, see, e.g., [31]. We set the HT tree to be a perfectly balanced binary tree. We vary the dimension as $d = 2, 4, 8, 16, 32$.

5.0.1 Results

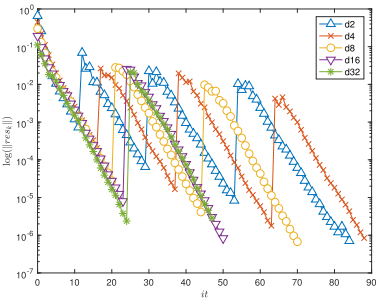
In Fig. 1a, we display the convergence history with respect to the number of overall iterations. Due to the structure of the linear operator A, the condition number $\kappa(\boldsymbol{A}^\delta)$ is independent of d. Moreover, the parameters $\alpha, \omega_1, \omega_2 \in (0,1)$ are chosen the same for all dimensions. Thus, the theoretical convergence rate of **HT-AWGM** is independent of d, which is observed in Fig. 1a.

However, the parameters $\omega_3, \omega_4, \omega_5$ depend on d which result in different tolerances for the re-truncation and re-coarsening step.[7]

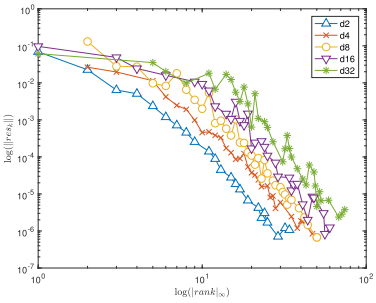
In Fig. 1b, we show the behavior of ranks of the numerical solution $\boldsymbol{u}_k$. The data points are sorted by rank, where for repeating ranks we took the minimum of the corresponding residual. For all dimensions d, we observe an exponential decay w.r.t. ranks, which is according to expectation for the Laplacian. As stated in Remark 4.9 and consistent with the observations in [3], we expect the ranks to scale logarithmically in the dimension.

In Fig. 1c, we plot the sum of the supports of frames and the corresponding residual. Since we are using cubic multiwavelets, we expect the convergence w.r.t. the

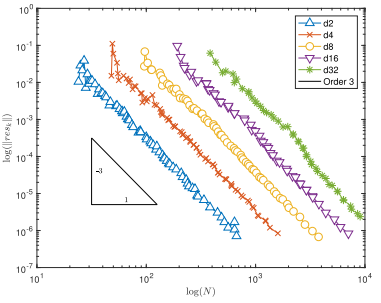
[7] In the graphics, re-truncation and re-coarsening are counted as one iteration step, though technically it is not a **HT-AWGM** iteration step.



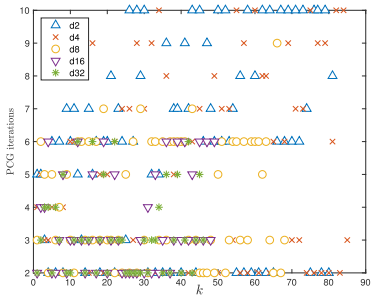
(a) Residual per iteration.



(b) Residual vs. ranks.



(c) Residual vs. size of support of frames.



(d) # **PCG** iterations per **HT-AWGM** iteration.

Fig. 1 **HT-AWGM** for different dimensions d

support size to be of order 3 and the dimension dependence to be slightly more than linear.

Finally, Fig. 1d shows the number of **PCG** iterations in each **HT-AWGM** iteration. We see that **PCG** requires between 2 and 10 iterations to achieve a fixed error reduction (ω_2) for all dimensions d, since $\kappa(\boldsymbol{A}^\delta)$ does not depend on d.

5.0.2 Discussion

We would like to emphasize that, unlike in classical non-tensor adaptive methods, for high-dimensional tensor methods, ranks are crucial for performance. The size of the wavelet discretization affects the performance indirectly, since the maximum wavelet level affects the ranks in the preconditioning. However, this is not necessarily a feature solely of the preconditioning. Larger frames imply we are searching for low-dimensional manifolds in higher dimensional spaces. In the worst-case scenario, this implies the ranks of such manifolds will grow.

A few numerical considerations significantly improve the overall performance. For **PCG**, choosing the adaptive tolerance is a trade off between the number iterations and how expensive each iteration is. We found that choosing the adaptive tolerance

0.1 (parameter δ in [27, Algorithm 2]) yields the best results. For experiments varying the adaptive tolerance, we refer to [27].

Moreover, note that in each **PCG** iteration, the preconditioned matrix-vector product only has to be computed once, since this can be avoided for computing the energy norm of the search direction. We are only interested in computing the residual and thus we can also avoid computing an intermediate matrix-vector product and apply the preconditioning, summation, and truncation to the residual directly. This gives the same result, but involves much lower intermediate ranks, since the truncation tolerances are relative to the residual and not $\|\boldsymbol{A}^{\delta}\boldsymbol{u}_k\|$.

5.0.3 Comparison

We conclude this section by discussing some implementation variants for **HT-AWGM** and a comparison with the method presented in [3] which we refer to as **HT-RICH**.

1. As discussed in Section 4.7, re-coarsening is required by the theory to ensure the complexity estimates, as without this step one cannot guarantee that the complexity of the frame supports and consequently ranks will remain bounded. Numerically, however, it might be advantageous to omit the re-coarsening step, if it does not significantly improve the observed ranks. The same considerations apply to the re-truncation step.
2. The most time-consuming part of **HT-AWGM** (and our implementation of **HT-RICH**) is the application and subsequent truncation of the preconditioned operator

$$\boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{A}_\Lambda\boldsymbol{S}^{-1}(\delta,\eta),$$

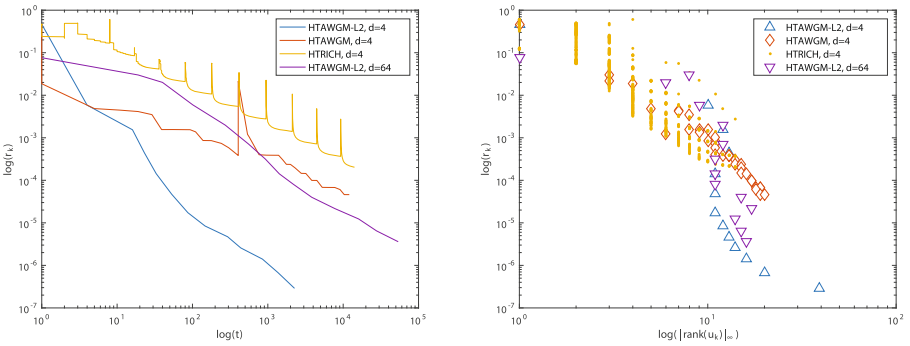
which essentially involves applying the preconditioner $\boldsymbol{S}^{-1}(\delta,\eta)$ twice. The simultaneous application and truncation were detailed in [3].

It is standard for preconditioned CG methods on finite-dimensional linear systems that, instead of applying $\boldsymbol{S}^{-1}(\delta,\eta)\boldsymbol{A}_\Lambda\boldsymbol{S}^{-1}(\delta,\eta)$ to the residual, we rewrite the preconditioned linear system such that we only have to apply

$$\boldsymbol{S}^{-2}(\delta,\eta)\boldsymbol{A}_\Lambda,$$

to the residual. This can be applied in the **PCG** step of **HT-AWGM** as well, which significantly reduces the computational effort, since now we only have to apply the preconditioner $\boldsymbol{S}^{-2}(\delta,\eta)$ once.

For finite-dimensional systems, both alternatives would be equivalent, with the latter requiring fewer matrix-vector applications. However, in the adaptive setting of a growing discretization, this essentially means we are working with coefficients $\boldsymbol{u}$ re-scaled in L_2. I.e., upon truncating we can only guarantee L_2-error control – not H^1. The same modification for **HT-RICH** was applied in [2], where the authors observed a reduction in numerical cost at the expense of a lack of H^1-error control.



(a) Residual vs. time (in seconds). (b) Residual vs. solution ranks.

Fig. 2 Run time performance of **HT-AWGM** and **HT-RICH**

In Fig. 2, we compare the run time performance of two versions of **HT-AWGM** and our implementation of the method from [3], which we refer to as **HT-RICH**.[8] We set $f = 1$.

For **HT-AWGM**, we compare the following two versions: one as before with **PCG** with H^1-error control, with a re-truncation but without a re-coarsening step; and one without a re-coarsening and a re-truncation step, and a **PCG** routine with L_2-error control as explained above. The bulk chasing parameter is set to $\alpha = 0.1$ in the former version, and to $\alpha = 0.95$ in the latter.

We refer to the latter version as "HTAWGM-L2." For **HT-RICH**, we set all parameters to the theoretical values as described in [3].

In Fig. 2a, we observe that all routines exhibit similar asymptotic behavior. For higher dimensions, the convergence rate for **HT-AWGM** remains the same as is predicted by the theoretical expectations for this model problem. From Fig. 2b, we see that the rank behavior for a given residual accuracy is similar for all routines. This is also the reason for the same asymptotic behavior in Fig. 2a, as tensor ranks effectively determine the overall complexity. Moreover, the support of frames of the solution behaves as in Fig. 2c for all algorithm variants.

The (non-asymptotic) run time behavior, however, differs. The **HT-AWGM** version without re-truncation is much faster than both **HT-AWGM** with re-truncation and **HT-RICH**. The H^1-error control in the former ensures that the number of **PCG** iterations remains bounded even for higher accuracies and wavelet levels. However, each iteration is more expensive, such that overall the convergence is slower. For **HT-AWGM**, we observe one additional truncation step and the same convergence as for **HT-RICH**, with a constant speed up factor.

[8] Both **HT-AWGM** and **HT-RICH** were implemented using the same libraries and computational routines; the tests were performed on the same hardware.

5.0.4 Summary

We highlight the following key messages that numerical tests in this work (and in part in [3]) suggest:

- Low-rank methods allow to deal with very high-dimensional problems, inaccessible to classical methods, *provided* the differential operator has a simple tensor product structure.
- The most important factor driving complexity by far is the ranks of the numerical iterates.
- The asymptotic behavior of both **HT-AWGM** and **HT-RICH** is similar (as expected), with the potential of speed-ups within **HT-AWGM** due to the flexibility of use of faster iterative methods in the **SOLVE** step (cf. Section 4).
- Behavior of ranks and support of frames for the Poisson problem within **HT-AWGM** seems to be unaffected by additional re-coarsening or re-truncation steps. However, as discussed in Section 4.7, this cannot be shown rigorously for the current version of the algorithm.

6 Conclusion

We introduced and analyzed an adaptive wavelet Galerkin scheme for high-dimensional elliptic equations. To deal with the *curse of dimensionality*, we utilized low-rank tensor methods, specifically the hierarchical Tucker format. The method is adaptive both in the wavelet representation and the tensor ranks.

We have shown that the method converges and that the numerical solution has quasi-optimal ranks and wavelet representation. The computational complexity depends solely on the Besov regularity and low-rank approximability of the solution. We provided numerical experiments for the Poisson equation for $d = 2, 4, 8, 16$ and 32 dimensions.

The method is well suited for problems where d is large, provided certain favorable separability assumptions are satisfied: the operator is either low rank or can be accurately approximated by low-rank operators; the condition of the operator only mildly depends on the dimension; the right-hand side is either low rank or can be accurately approximated by low-rank functions. The dominating part of the complexity is the ranks.

HT-AWGM involves a re-truncation and a re-coarsening step to ensure optimality w.r.t. ranks and wavelet representation. Although there is evidence to support that both can be avoided, we were not able to devise a rigorous framework to show this. In future work, we want to construct a more clever rank extension strategy, avoid both the re-truncation and re-coarsening steps, and consider different Galerkin solvers.

Acknowledgments We would like to thank Markus Bachmayr, Rob Stevenson, and Wolfgang Dahmen for their very helpful comments on this work. This paper was partly written when Mazen Ali was a visiting researcher at Centrale Nantes in collaboration with Anthony Nouy. We acknowledge Anthony Nouy for the helpful discussions and financial support.

Funding information This study was financially supported by the European Model Reduction Network (TD COST Action TD1307).

References

1. Bachmayr, M., Dahmen, W.: Adaptive near-optimal rank tensor approximation for high-dimensional operator equations. Found Comput. Math. **15**(4), 839–898 (2015)
2. Bachmayr, M., Dahmen, W.: Adaptive low-rank methods for problems on Sobolev spaces with error control in L_2. ESAIM Math. Model. Numer Anal. **50**(4), 1107–1136 (2016)
3. Bachmayr, M., Dahmen, W.: Adaptive low-rank methods: problems on Sobolev spaces. SIAM J. Numer. Anal. **54**(2), 744–796 (2016)
4. Bachmayr, M., Schneider, R.: Iterative methods based on soft thresholding of hierarchical tensors. Found Comput. Math. **17**(4), 1037–1083 (2017)
5. Bachmayr, M., Schneider, R., Uschmajew, A.: Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. Found. Comput. Math., pp 1–50 (2016)
6. Ballani, J., Grasedyck, L.: A projection method to solve linear systems in tensor format. Numer Linear Algebra Appl. **20**(1), 27–43 (2013)
7. Billaud-Friess, M., Nouy, A., Zahm, O.: A tensor approximation method based on ideal minimal residual formulations for the solution of high-dimensional problems. ESAIM: Mathematical Modelling and Numerical Analysis **48**(6), 1777–1806 (2014)
8. Cohen, A., Dahmen, W., DeVore, R.: Adaptive wavelet methods for elliptic operator equations: convergence rates. Math. Comp. **70**(233), 27–75 (2001)
9. Cohen, A., Dahmen, W., DeVore, R.: Adaptive wavelet methods. II. Beyond the elliptic case. Found. Comput. Math. **2**(3), 203–245 (2002)
10. Dahmen, W., DeVore, R., Grasedyck, L., Süli, E.: Tensor-sparsity of solutions to high-dimensional elliptic partial differential equations. Found Comput. Math. **16**(4), 813–874 (2016)
11. D'Aspremont, A.: Smooth optimization with approximate gradient. SIAM Journal on Optimization **19**, 1171–1183 (2008)
12. Devolder, O., Glineur, F., Nesterov, Y.: First-order methods of smooth convex optimization with inexact oracle. Mathematical Programming **146**(1-2), 37–75 (2013)
13. DeVore, R.A.: Nonlinear approximation. In: Acta Numerica, vol. 7, pp. 51–150 (1998). Cambridge Univ. Press, Cambridge, 1998
14. Dolgov, S., Khoromskij, B.: Simultaneous state-time approximation of the chemical master equation using tensor product formats. Numer. Linear Algebra Appl. **22**(2), 197–219 (2015)
15. Dolgov, S.V., Savostyanov, D.V.: Alternating minimal energy methods for linear systems in higher dimensions. SIAM Journal on Scientific Computing **36**(5), A2248–A2271 (2014)
16. Fischer, B.: Polynomial Based Iteration Methods for Symmetric Linear Systems. Wiley-Teubner Series Advances in Numerical Mathematics. Wiley, Chichester (1996). B.G. Teubner, Stuttgart
17. Gantumur, T., Harbrecht, H., Stevenson, R.: An optimal adaptive wavelet method without coarsening of the iterands. Math Comp. **76**(258), 615–629 (2007)
18. Hackbusch, W.: Iterative Lösung Groß Er Schwachbesetzter Gleichungssysteme, vol. 69. B. G. Teubner, Stuttgart (1991)
19. Hackbusch, W., calculus, n.umerical.t.ensor.: Tensor Spaces Vol. 42 of Springer Series in Computational Mathematics. Springer, Heidelberg (2012)
20. Hackbusch, W., Kühn, S.: A new scheme for the tensor representation. J Fourier Anal. Appl. **15**(5), 706–722 (2009)
21. Holtz, S., Rohwedder, T., Schneider, R.: The alternating linear scheme for tensor optimization in the tensor train format. SIAM J. Sci. Comput. **34**(2), A683–A713 (2012)
22. Jarre, F., Stoer, J.: Optimierung. Springer, Berlin (2004)
23. Kestler, S.: On the Adaptive Tensor Product Wavelet Galerkin Method with Applications in Finance. PhD thesis, Ulm University (2013)
24. Khoromskij, B.N.: Tensor-structured preconditioners and approximate inverse of elliptic operators in $\mathbb{R}^d$. Constr. Approx. **30**(3), 599–620 (2009)
25. Khoromskij, B.N., Oseledets, I.: Quantics-TT collocation approximation of parameter-dependent and stochastic elliptic PDEs. Comput. Methods Appl Math. **10**(4), 376–394 (2010)

26. Khoromskij, B.N., Schwab, C.: Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs. SIAM J. Sci. Comput. **33**(1), 364–385 (2011)
27. Kressner, D., Tobler, C.: Low-rank tensor Krylov subspace methods for parametrized linear systems. SIAM J. Matrix Anal. Appl. **32**(4), 1288–1316 (2011)
28. Nochetto, R.H., Siebert, K.G., Veeser, A.: Theory of adaptive finite element methods: an introduction. In: Multiscale, Nonlinear and Adaptive Approximation, pp. 409–542. Springer, Berlin (2009)
29. Oseledets, I.V.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)
30. Oseledets, I.V., Dolgov, S.V.: Solution of linear systems and matrix inversion in the TT-format. SIAM J. Sci. Comput. **34**(5), A2718–A2739 (2012)
31. Rupp, A.: High Dimensional Wavelet Methods for Structured Financial Products. PhD thesis, Ulm University (2013)
32. Schmidt, M., Roux, N.L., Bach, F.R.: Convergence rates of inexact proximal-gradient methods for convex optimization. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 24, pp. 1458–1466. Curran Associates, Inc. (2011)
33. Schneider, R., Uschmajew, A.: Approximation rates for the hierarchical tensor format in periodic Sobolev spaces. J. Complexity **30**(2), 56–71 (2014)
34. Stevenson, R.: Optimality of a standard adaptive finite element method. Found Comput. Math. **7**(2), 245–269 (2007)
35. Stevenson, R.: Adaptive wavelet methods for solving operator equations: an overview. In: Multiscale, Nonlinear and Adaptive Approximation, pp. 543–597. Springer, Berlin (2009)
36. Urban, K.: Wavelet Methods for Elliptic Partial Differential Equations. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford (2009)