Research Article

Maximilian Bauer* and Mario Bebendorf

Block-Adaptive Cross Approximation of Discrete Integral Operators

https://doi.org/10.1515/cmam-2019-0085
Received May 13, 2019; revised December 30, 2019; accepted January 13, 2020

Abstract: In this article, we extend the adaptive cross approximation (ACA) method known for the efficient approximation of discretisations of integral operators to a block-adaptive version. While ACA is usually employed to assemble hierarchical matrix approximations having the same prescribed accuracy on all blocks of the partition, for the solution of linear systems, it may be more efficient to adapt the accuracy of each block to the actual error of the solution as some blocks may be more important for the solution error than others. To this end, error estimation techniques known from adaptive mesh refinement are applied to automatically improve the blockwise matrix approximation. This allows to interlace the assembling of the coefficient matrix with the iterative solution.

Keywords: ACA, Error Estimators, BEM, Non-local Operators, Hierarchical Matrices, Fast Solvers

MSC 2010: 65N38

1 Introduction

Methods for the solution of boundary value problems of elliptic partial differential equations typically involve non-local operators. Examples are fractional diffusion problems (see [1, 14, 15]) and boundary integral methods (see [29, 30]), in which amenable boundary value problems are reformulated as integral equations over the boundary of the computational domain. The discretisation of such non-local operators usually leads to fully populated matrices. The corresponding systems of linear equations are quite expensive to solve numerically in the typical case of large numbers of unknowns. Additionally, assembling and storing the discrete operators may be quite costly. Even if the formulation of the boundary value problem is local and its discretisation (e.g. by the finite element method) leads to a sparse matrix, it may be useful to construct non-local approximations (e.g. for preconditioning) to its inverse [6] or to triangular factorisations. In both cases, suitable techniques have to be employed if the boundary value problem is to be solved with linear or almost linear complexity.

There are several ways how fully populated discretisations of elliptic operators can be treated efficiently. In addition to fast multipole methods [20] or mosaic decomposition methods [31], hierarchical matrices ($\mathcal{H}$-matrices) [22, 24] have been introduced for the asymptotic optimal numerical treatment. This kind of methods can be regarded to be based on a hierarchical partition of the discretised operator $A \in \mathbb{R}^{N \times N}$ into suitable blocks combined with a local low-rank approximation. Hence, fully populated matrices can be approximated by data-sparse representations which require only a logarithmic-linear amount of storage.

*Corresponding author: Maximilian Bauer, Faculty of Mathematics, Physics and Computer Science, University of Bayreuth, 95447 Bayreuth, Germany, e-mail: maximilian1.bauer@uni-bayreuth.de
Mario Bebendorf, Faculty of Mathematics, Physics and Computer Science, University of Bayreuth, 95447 Bayreuth, Germany, e-mail: mario.bebendorf@uni-bayreuth.de
Furthermore, these representations can be applied to vectors with logarithmic-linear complexity. In addition, $\mathcal{H}$-matrices provide approximate substitutes for the usual matrix operations including the LU factorisation with logarithmic-linear complexity; see [7, 23]. These properties strongly accelerate iterative solvers like the conjugate gradient (CG) method or GMRES (see [28]) for the solution of the arising linear systems of equations.

A crucial step for the efficient treatment of fully populated matrices $A$ is their assembling. While explicit kernel approximations were used in the early stages of fast methods for the treatment of integral operators, in recent years, the adaptive cross approximation (ACA) (see [4, 12]) has become quite popular. The latter method employs only few of the original matrix entries of $A$ for its data-sparse approximation by hierarchical matrices. Since one is usually interested in approximations $\tilde{A}$ of $A$ leading to a sufficiently accurate result when $\tilde{A}$ or its inverse is applied to arbitrary vectors, it is common to approximate all blocks of the matrix partition uniformly with a prescribed accuracy. The universality of the approximation to the discrete operator $A$ comes at a high price due to the generation of information that may be redundant for the solution of the linear system. When the quantity of interest is the error of the solution $x$ of the linear system $Ax = b$, $\tilde{A}$ obtained from usual ACA may not be the ideal approximation to $A$. In such a situation, certain parts of the discrete operator $A$ may be more important than others. Conversely, depending on the right-hand side $b$, some blocks may not be that important for the error of $x$.

In this article, we propose to construct the hierarchical matrix approximation in a block-adaptive way, i.e., in addition to the adaptivity of ACA on each block, we add another level of adaptivity to the construction of the hierarchical matrix. This variant of ACA will be referred to as BACA. In order to find an $\mathcal{H}$-matrix approximation that is better suited for the solution error, we employ techniques known from adaptivity together with suitable error estimators. The strategy of error estimation is well known in the context of numerical methods for partial differential or integral equations. Adaptive methods usually focus on mesh refinement (see e.g. [18, 25]). For the new method BACA, similar strategies will be used to successively improve blockwise low-rank approximations, whereas the underlying grid (and the hierarchical block structure) does not change. Therefore, ACA and BACA approximate the operator on the same grid. Although the ideas of this approach are presented in connection with ACA, they can also be applied to other low-rank approximation techniques.

In contrast to constructing the approximation $\tilde{A}$ of $A$ the usual way and solving a single linear system $\tilde{A}x = b$, we generate a sequence of $\mathcal{H}$-matrix approximations $A_k$ to $A$ and solve each linear system $A_k x_k = b$ for $x_k$. At first glance, this seems to be more costly than solving $\tilde{A}x = b$ only once. However, the construction of the next approximation $A_{k+1}$ can be steered by the approximate solution $x_k$, and $x_{k+1}$ can be efficiently computed as an update of $x_k$. Furthermore, the accuracy of $x_k$ in the iterative solver can be adapted to the error estimator. This allows to interlace the assembling process of the matrix with the iterative solution of the linear system. In addition to reducing the numerical effort for assembling and storing the matrix, this approach has the practical advantage that the blockwise accuracy of ACA is not a required parameter any more. Choosing this parameter in usual ACA approximations is not obvious as the relation between the blockwise accuracy and the error of the solution usually depends on the respective problem. With the new approach presented in this article, the accuracy of the approximation is automatically adapted.

The article is organised as follows. In Section 2, the model problem is formulated for which the new method will be explained. We consider Galerkin discretisations of a boundary integral formulation of Laplace’s equation with Dirichlet boundary conditions. Section 3 gives a brief overview of hierarchical matrices and ACA as these methods are the foundations of our new method. In order to successively improve blockwise low-rank approximations, it is mandatory to choose those blocks which are responsible for large contributions to the error. Therefore, in Section 4, suitable error estimators will be proposed and investigated with respect to efficiency and reliability. Using these, a marking and refinement strategy will be presented and the convergence of the residuals $\lim_{k \to \infty} \|b - Ax_k\| = 0$ will be analysed. Numerical examples will be presented in Section 5 which show that the new method is more efficient than ACA with respect to both computational time and required storage in situations where the solution in some sense contains structural differences or situations that result from locally over-refined meshes.
2 Model Problem

As a model problem, we consider the Laplace equation on a domain $\Omega \subset \mathbb{R}^d$ (or its complement $\Omega^c := \mathbb{R}^d \setminus \overline{\Omega}$) with sufficiently smooth boundary $\partial \Omega$, i.e.

$$-\Delta u = 0 \quad \text{in} \ \Omega \ (\text{or} \ \Omega^c),$$
$$u = g \quad \text{on} \ \partial \Omega,$$

where the Dirichlet data $g$ is a given function in the trace space $H^{1/2}(\partial \Omega)$ of the Sobolev space $H^1(\Omega)$. This kind of problem (in particular, the exterior boundary value problem with suitable conditions at infinity) can be treated via boundary integral formulation using the fundamental solution

$$S(x) = \begin{cases}
\frac{-1}{2\pi} \ln |x|, & d = 2, \\
\frac{1}{(d-2)|x|^{d-2}}, & d > 2,
\end{cases}$$

for $x \in \mathbb{R}^d \setminus \{0\}$, where $\omega_d$ denotes the area of the $d$-dimensional unit sphere. Defining the operators

$$\nabla \psi(x) := \int_{\partial \Omega} \psi(y) S(x - y) \, ds_y \quad \text{(single layer potential)},$$
$$\mathcal{K} \phi(x) := \int_{\partial \Omega} \phi(y) \partial_n(y) S(x - y) \, ds_y \quad \text{(double layer potential)},$$

the solution of (2.1) is given via the representation formula

$$u(x) = \nabla \psi(x) - \mathcal{K} g(x), \quad x \in \mathbb{R}^d \setminus \partial \Omega,$$

where the Neumann data $\psi \in H^{-1/2}(\partial \Omega)$ is the solution of the boundary integral equation

$$\nabla \psi = \left( \frac{1}{2} \nabla + \mathcal{K} \right) g \quad \text{on} \ \partial \Omega.$$  \hfill (2.2)

If we extend the $L^2$-scalar product to a duality pairing between $H^{-1/2}(\partial \Omega)$ and $H^{1/2}(\partial \Omega)$, the boundary integral equation (2.2) can be stated in variational form as

$$\left( \nabla \psi, \psi' \right)_{L^2} = \left( \left( \frac{1}{2} \nabla + \mathcal{K} \right) g, \psi' \right)_{L^2}, \quad \psi' \in H^{-1/2}(\partial \Omega).$$

Due to the coercivity of the bilinear form $(\cdot, \cdot)_{L^2}$, the Riesz–Fischer theorem yields the existence of the unique solution $\psi \in H^{-1/2}(\partial \Omega)$.

In order to solve (2.2) or its variational form numerically, Galerkin discretisations are often used. In the case $d = 3$, the boundary $\partial \Omega$ is decomposed into a conforming triangulation $T$ consisting of $N$ plane surface triangles; see [29]. Let the set $\{\psi_1, \ldots, \psi_N\}$ denote the canonical basis of the space of piecewise constant functions $\mathcal{P}_0(T)$. Then, for $\psi = \sum_{j=1}^{N} x_j \psi_j$, $x_j \in \mathbb{R}$, the discretisation of the variational problem reads

$$\left( \left( \frac{1}{2} \nabla + \mathcal{K} \right) g, \psi \right)_{L^2} = \left( \nabla \psi, \psi_1 \right)_{L^2} = \int_{\partial \Omega} \int_{\partial \Omega} S(x - y) \psi(y) \, ds_y \, ds_x$$
$$= \sum_{j=1}^{N} x_j \left[ \int_{\partial \Omega} \int_{\partial \Omega} S(x - y) \psi_j(y) \psi_i(x) \, ds_y \, ds_x \right], \quad i = 1, \ldots, N,$$

which due to the Riesz–Fischer theorem is uniquely solvable, too. With

$$a_{ij} := \int_{\partial \Omega} \int_{\partial \Omega} S(x - y) \psi_j(y) \psi_i(x) \, ds_y \, ds_x \quad \text{and} \quad b_i := \left( \left( \frac{1}{2} \nabla + \mathcal{K} \right) g, \psi_i \right)_{L^2}, \quad i, j = 1, \ldots, N,$$

the above stated discretisation results in the system of linear equations

$$Ax = b, \quad A \in \mathbb{R}^{N \times N}, \quad b \in \mathbb{R}^N.$$  \hfill (2.3)
Since the matrix $A$ is fully populated, our goal is to reduce the required storage for $A$ and the numerical effort for its construction to logarithmic-linear complexity. To this end, hierarchical matrices will be employed; see Section 3. The efficiency of hierarchical matrices is based on a blockwise low-rank approximation. The construction of these low-rank approximations is often done via adaptive cross approximation; see Section 3.1. While in hierarchical matrix approximations all blocks are commonly approximated with the same prescribed accuracy, in this article, we propose to construct the hierarchical matrix approximation in a block-adaptive way, i.e., in addition to the adaptivity of ACA on each block, we add another level of adaptivity to the construction of the hierarchical matrix.

Although we consider the model problem (2.1), the method presented in this article can be applied in more general situations as long as the matrix $A$ results from finite element discretisation of an integral representation of the problem. Examples are the Helmholtz equation (see [30]), Lamé equations (see [30]), the Lippmann–Schwinger equation (see e.g. [13, 26]), fractional diffusion problems (see [1, 14, 15]), the Gauss transform (see e.g. [21]) and many others.

### 3 Hierarchical Matrices and the Adaptive Cross Approximation

This section gives a short overview of the hierarchical matrix (H-matrix) structure and the ACA method, which we are going to extend. H-matrices were introduced by Hackbusch [22] and Hackbusch and Khoromskij [24]. Their efficiency is based on a suitable hierarchical partitioning of the matrix into sub-blocks and the blockwise approximation by low-rank matrices.

We consider matrices $A ∈ ℝ^{N×N}$ whose entries result from the discretisation of a non-local operator such as the boundary integral formulation of the boundary value problem described in Section 2. Although $A$ is fully populated, it can often be approximated by low-rank matrices on suitable blocks $t \times s$ consisting of rows and columns $t, s ∈ I := \{1, \ldots, N\}$ of $A$, i.e.,

$$A_{ts} ≈ UV^T, \quad U ∈ ℝ^{r×r}, \quad V ∈ ℝ^{s×s},$$

where the rank $r$ is small compared with $|t|$ and $|s|$. If $A$ discretises an integral representation or the inverse of second-order elliptic partial differential operators, then the condition

$$\min\{\text{diam } X_t, \text{diam } X_s\} < β \text{dist}(X_t, X_s), \quad β > 0,$$

(3.1)

on the block $t \times s$ turns out to be appropriate; see [5]. Here,

$$X_t := \bigcup_{i \in t} X_i$$

is the union of the supports $X_i := \text{supp } ψ_i, i ∈ t$, and

$$\text{diam } X = \sup_{x,y ∈ X} |x - y| \quad \text{and} \quad \text{dist}(X, Y) = \inf_{x ∈ X, y ∈ Y} |x - y|$$

denote the diameter and the distance of two bounded sets $X, Y ⊂ ℝ^d$. All blocks $t \times s$ satisfying this condition are called admissible; see Figure 1.

A partition $P$ of the matrix indices $I \times I$ is usually constructed using cluster trees, i.e. trees $T_I$ satisfying the conditions

1. $I$ is the root of the tree,
2. If $ν ∈ T_I$ is not a leaf, then $ν$ has sons $ν_1, \ldots, ν_p ∈ T_I$ so that $ν = ν_1 ∪ \cdots ∪ ν_p$.

The $ℓ$-th level of $T_I$ will be referred to as $T_{I(ℓ)}, \ell = 0, \ldots, L$, where $L$ is the depth of $T_I$. The set of leaves of the tree $T_I$ is denoted by $ℒ(T_I)$. Given a cluster tree $T_I$ for $I$, a block cluster tree $T_{I×I}$ for $I × I$ is constructed by recursive subdivision descending the cluster tree $T_I$ for the rows and for the columns concurrently until (3.1) is satisfied or one the clusters cannot be subdivided any further. Hence, a matrix partition consisting of blocks either satisfying (3.1) or being small can be found as the leaves $P := ℒ(T_{I×I})$ of $T_{I×I}$. $P$ is the union of
the sets of admissible and non-admissible blocks $P_{\text{adm}}$ and $P_{\text{non-adm}}$, respectively. The sparsity constant $c_{sp}$ is defined as

$$c_{sp} := \max\left\{ \max_{t \in T_f} c_{sp}^t(s), \max_{s \in I_s} c_{sp}(s) \right\},$$

where $c_{sp}^t(s) := |\{ s \in I : t \times s \in P \}|$ denotes the maximum number of blocks $t \times s$ contained in $P$ for a given cluster $t \in T_f$ and $c_{sp}(s) := |\{ t \in I : t \times s \in P \}|$ the maximum number of blocks $t \times s \in P$ for a cluster $s \in I_s$. For more details on cluster trees (in particular, for their construction), the reader is referred to [7, 23].

Given a matrix partition $P$, we define the set of $\mathcal{H}$-matrices with blockwise rank at most $r$ by

$$\mathcal{H}(P, r) := \{ M \in \mathbb{R}^{I \times I} : \text{rank } M_b \leq r \text{ for all } b \in P_{\text{adm}} \}.$$

Due to the structure of hierarchical matrices and due to the fact that each block $t \times s$ has complexity $r(|t| + |s|)$, the complexity of the whole $\mathcal{H}$-matrix can be shown to be of the order $rN \log N$; see [7, 23].

### 3.1 Matrix Construction

There are several ways how low-rank approximations of a matrix block can be constructed. A common analytic approach is to replace the kernel functions of the integral operators shown in Section 2 either with truncated kernel expansions such as the multipole expansion [20] or with interpolating polynomials. Algebraic approaches such as [19] or randomised methods [27] tackle the matrix block directly. For matrices $A$ resulting from the discretisation of non-local operators, the adaptive cross approximation (ACA) method introduced in [4, 9, 12] constructs the low-rank approximation using only few of the original entries of $A$. Non-admissible blocks cannot be approximated but are small, so they are computed entry by entry. We focus on a single admissible block $A_{15} \in \mathbb{R}^{r \times s}$ of $A$. With $R_0 := A_{15}$, we define two sequences of vectors $u_r \in \mathbb{R}^{I}$ and $v_r \in \mathbb{R}^{I}$ for $r = 1, 2, 3, \ldots$ by Algorithm 1.

The matrix

$$S_r := \sum_{i=1}^{r} u_i v_i^T$$

has rank at most $r$ and can be shown (under reasonable assumptions) to be an approximation of $A_{15}$ with remainder $R_r := A_{15} - S_r$ having relative accuracy

$$\|R_r\|_F \leq \epsilon_{\text{ACA}} \|A_{15}\|_F,$$

(3.2)

where $\| \cdot \|_F$ denotes the Frobenius norm. The set $Z$ collects all vanishing rows of the matrix $R_r$. The vectors $u_r$ and $v_r$ are columns and rows of $R_{r-1}$, i.e.

$$u_r = (R_{r-1})_{i_r}^T, \quad v_r = (R_{r-1})_{i_r,s}.$$

The row indices $i_r$ have to be chosen such that the Vandermonde matrix corresponding to the system in which the approximation error is to be estimated is non-singular; for details, see [7]. If the $i_r$-th row of $R_{r-1}$ is nonzero and hence is used as $v_r$ after scaling $v_r$ with $1/(v_r)_{i_r}$, it is also added to $Z$ since the $i_r$-th row of the following
Let \( r = 1; Z = \emptyset; \varepsilon_{ACA} > 0 \)

repeat

\[ \text{find } i_r \text{ by some rule} \]
\[ \tilde{v}_r := A_{i_r,s} \]

for \( l = 1, \ldots, r-1 \) do

\[ \tilde{v}_r := \tilde{v}_r - (u_l)_{i_l} v_l \]

end for

\[ Z := Z \cup \{ i_r \} \]

if \( \tilde{v}_r \) does not vanish then

\[ j_r := \arg\max_{j \in s} |(\tilde{v}_r)_j|; v_r := (\tilde{v}_r)_{j_r}^{-1} \tilde{v}_r \]
\[ u_r := A_{t,j_r} \]

for \( l = 1, \ldots, r-1 \) do

\[ u_r := u_r - (v_l)_{j_r} u_l \]

end for

\[ r := r + 1 \]

end if

until \( \|u_{r+1}\|_2 \|v_{r+1}\|_2 \leq \frac{\varepsilon_{ACA}(1-\beta)}{1+\varepsilon_{ACA}} \|S_r\|_F \) or \( Z = t \)

Algorithm 1: Adaptive Cross Approximation (ACA).

remainder \( R_r \) will vanish. It can be shown that the numerical effort of ACA is of the order \( |Z|^2 (|t| + |s|) \). While for usual ACA the number of steps \( |Z| \) conducted by the previous algorithm typically depends logarithmically on the desired blockwise accuracy \( \varepsilon_{ACA} \) (see [7] for a convergence analysis), in the next section, the number of steps \( |Z| \) for each block \( A_{ts} \) will be steered by an adaptive algorithm that aims at the quality of the whole approximation of \( A \).

4 Error Estimators and the Block-Adaptive Cross Approximation (BACA)

The usual way of treating discrete problems (2.3) is to construct an hierarchical matrix approximation \( \tilde{A} \) (e.g. via ACA) such that each block \( \tilde{A}_b, b \in P \), satisfies (cf. (3.2))

\[ \|A_b - \tilde{A}_b\|_F \leq \varepsilon \|A_b\|_F \quad \text{for all } b \in P. \] (4.1)

Due to the independence of the blocks, this can be done in parallel; see [11]. While (4.1) guarantees local properties of the \( \mathcal{H} \)-matrix approximation, the global impact of this condition is difficult to estimate. Notice that (4.1) still implies the obvious global property \( \|A - \tilde{A}\|_F \leq \varepsilon \|A\|_F \); however, for instance, the eigenvectors corresponding to small eigenvalues of \( A \) and \( \tilde{A} \) are significantly less accurate than \( \varepsilon \) in general, and one has to apply suitable techniques (see [8]) in order to be able to guarantee spectral equivalence of \( A \) and \( \tilde{A} \). Another example is that condition (4.1) does not preserve positive definiteness of the matrix \( A \) for \( \tilde{A} \) unless additional stabilisation techniques are applied; see [10]. In other words, the low-rank approximation is usually constructed regardless of the importance of the respective block for global properties of the matrix.

\( \tilde{A} \) obtained from usual ACA may not be the ideal approximation to \( A \) when the norm of the solution error of the associated solution is the measure. In order to find an \( \mathcal{H} \)-matrix approximation that is better suited for this problem, we employ techniques known from adaptivity together with suitable error estimators. The strategy of error estimation is well known in the context of numerical methods for partial differential or integral equations. There are several types of a posteriori error estimators. The method of this article is inspired by the \((h - h/2)\)-version investigated in [17]. Adaptive methods usually focus on mesh refinement. For BACA, similar strategies will be used to successively improve blockwise low-rank approximations, whereas the underlying grid (and the hierarchical block structure \( P \)) does not change. Therefore, ACA and the new method BACA approximate the operator on the same grid.
In contrast to constructing the matrix approximation \( Ax = b \), we build a sequence of \( \mathcal{H} \)-matrix approximations \( A_k \) of \( A \) and solve each linear system \( A_k x_k = b \) for \( x_k \). At first glance, this seems to be more costly than solving \( Ax = b \) only once. However, the approximation of \( A_{k+1} \) can be steered by the approximate solution \( x_k \), and \( x_{k+1} \) can be computed as an update of \( x_k \). Furthermore, the accuracy of \( x_k \) can be low for small \( k \), and we can adapt it to the estimated error.

The following method will be based on residual error estimators (see e.g. [2, 3]). Notice that if \( A : V \rightarrow V' \) denotes the operator that is discretised by \( A \) and \( u_h \) refers to the Galerkin approximation of the exact solution to \( Au = f \), then, for the residual error \( \| f - A u_h \|_{V'} \), we have

\[
c_B \| u - u_h \|_V \leq \| f - A u_h \|_{V'} = \sup_{\varphi \in V} \frac{|a(u - u_h, \varphi)|}{\| \varphi \|_V} \leq c_S \| u - u_h \|_V
\]

provided that the continuous bilinear form \( a : V \times V \rightarrow \mathbb{R} \) corresponding to \( A \), i.e. \( |a(u, v)| \leq c_S \| u \|_V \| v \|_V \) for all \( u, v \in V \), satisfies an \( \inf \)-\( \sup \) condition

\[
\inf \sup_{u \in V, v \in V} \frac{|a(u, v)|}{\| u \|_V \| v \|_V} \geq c_B.
\]

Hence, the solution error \( \| u - u_h \|_V \) is equivalent with the residual error \( \| f - A u_h \|_{V'} \), and in the rest of this section, we may focus on the residual error.

In Algorithm 2, \( \hat{A}_k \) denotes a better approximation of \( A \) than \( A_k \), that is, we assume that the saturation condition

\[
\| \hat{A}_k x_k - Ax_k \|_2 \leq q_{\text{sat}} \| A_k x_k - Ax_k \|_2
\]

(4.2)

holds with \( 0 < q_{\text{sat}} < 1 \). A natural choice for \( \hat{A}_k \) is the improved approximation that results from \( A_k \) by adding a fixed number of additional steps of ACA for each admissible block and by setting \( (\hat{A}_k)_b = A_b \) for all other blocks \( b \in P_{\text{non-adm}} \). It seems that two \( \mathcal{H} \)-matrices \( A_k \) and \( \hat{A}_k \) are required in Algorithm 2. Since they are strongly related, it is actually sufficient to store only \( \hat{A}_k \). Notice that the ACA steps required for the construction of \( A_{k+1} \) can be adopted from \( \hat{A}_k \), and \( \hat{A}_{k+1} \) can be computed from \( \hat{A}_k \) by improving the blocks contained in \( M_k \).

---

1. Start with a coarse \( \mathcal{H} \)-matrix approximation \( A_0 \) of \( A \), and set \( k = 0 \).
2. Given \( \vartheta \geq 0 \), apply CG to the linear system \( A_k x_k = b \) until the residual error of the iterate \( x_k = x_k^\alpha \) satisfies

\[
\| b - A_k x_k \|_2 \leq \alpha \| (A_k - \hat{A}_k) x_k \|_2
\]

(use \( x_{k-1} \) as a starting vector of an iterative solver; \( x_{-1} := 0 \)).
3. Given \( 0 < \vartheta < 1 \), find a set of marked blocks \( M_k \subset P_{\text{adm}} \) with minimal cardinality such that

\[
\eta_k(M_k) \geq \vartheta \eta_k,
\]

where \( \eta_k^2(M) := \sum_{s \in M} \| (A_k - \hat{A}_k)_s \|_2^2 \) and \( \eta_k := \eta_k(P_{\text{adm}}) \).

4. Let

\[
A_{k+1} = \begin{cases} 
(\hat{A}_k)_b, & b \in M_k, \\
(A_k)_b, & b \in P \setminus M_k.
\end{cases}
\]

5. If \( \eta_k > \varepsilon_{\text{BACA}} \), increment \( k \), and go to (2).

**Algorithm 2:** Block-Adaptive ACA.

Notice that the CG iteration in step (2) of Algorithm 2 is guaranteed to terminate with the provided stopping criterion as CG terminates after a finite number of iterations with the exact solution.
Algorithm 1 contains a stopping parameter $\epsilon_{\text{ACA}}$, which describes the desired accuracy of the low-rank approximation for the respective block. While it is difficult to choose $\epsilon_{\text{ACA}}$ so that the resulting solution of the linear system satisfies a prescribed accuracy, the parameter $\epsilon_{\text{EAMA}}$ in Algorithm 2 gives an upper bound on the residual error $\|b - Ax_k\|_2$ of $x_k$ as we shall see in Lemma 4.1.

In addition to the ability of the error estimator

\[ \eta_k^2 = \sum_{i \in \mathcal{P}_{\text{adm}}} \| (A_k - \hat{A}_k) t_i (x_k) \|_2^2 \]  

introduced in Algorithm 2 to localise the error, its reliability is a main issue for a successful adaptive procedure. In particular, the dependence on the residual error $\delta_k := \|b - A_k x_k\|_2$ of the approximate solution $x_k$ of $A_k x_k = b$ in Algorithm 2 has to be investigated.

**Lemma 4.1.** Let assumption (4.2) be valid. Then $\eta_k$ is reliable, i.e., it holds

\[ \|b - Ax_k\|_2 \leq \frac{1 + \alpha (1 + q_{\text{sat}})}{1 - q_{\text{sat}}} \| (A_k - \hat{A}_k)x_k \|_2 \leq \sqrt{c_{\text{sp}} L (1 + \alpha (1 + q_{\text{sat}}))} \eta_k. \]

**Proof.** We make use of the decomposition of $A \in \mathcal{A}(P, k)$ into a sum of level matrices $A^{(\ell)}$, which contain the blocks $b \in P$ of $A$ belonging to the $\ell$-th level of the block cluster tree $T_{k,t}$,

\[ A = \sum_{\ell=1}^{L} A^{(\ell)}. \]

Notice that $A^{(\ell)}$ is a Cartesian product block matrix. Due to $(\sum_{i=1}^{n} a_i)^2 \leq n \sum_{i=1}^{n} a_i^2$ for all $a_i \in \mathbb{R}$, $i = 1, \ldots, n$, we have

\[ \| (A_k - \hat{A}_k)x_k \|_2^2 \leq \left( \sum_{\ell=1}^{L} \| (A_k - \hat{A}_k)^{(\ell)} x_k \|_2 \right)^2 \leq L \sum_{\ell=1}^{L} \| W^{(\ell)}_k x_k \|_2^2 \]

\[ = L \sum_{\ell=1}^{L} \left( \sum_{t \in T^{(\ell)}_{k,t}} \sum_{s \in \mathcal{P}} (A_k - \hat{A}_k) t_{is}(x_k) \right)^2 \]

\[ \leq L \sum_{\ell=1}^{L} \sum_{t \in T^{(\ell)}_{k,t}} \sum_{s \in \mathcal{P}} \| (A_k - \hat{A}_k) t_{is}(x_k) \|_2^2 \]

\[ = c_{\text{sp}} L \eta_k^2. \]

The lemma follows from

\[ \|b - Ax_k\|_2 \leq \|b - A_k x_k\|_2 + \| (A_k - \hat{A}_k)x_k \|_2 + \| \hat{A}_k x_k - Ax_k \|_2 \quad \leq (1 + \alpha) \| (A_k - \hat{A}_k)x_k \|_2 + q_{\text{sat}} \| A_k x_k - Ax_k \|_2 \]

\[ \leq (1 + \alpha) \| (A_k - \hat{A}_k)x_k \|_2 + q_{\text{sat}} (\| (A_k - \hat{A}_k)x_k \|_2 + \| b - Ax_k \|_2). \]

Although the efficiency of the error estimator $\eta_k$, i.e., $\eta_k \leq \|b - Ax_k\|_2$, can be observed in numerical experiments, its provability is still an open question. Nevertheless, the computable expression $\| (A_k - \hat{A}_k)x_k \|_2$ can be used as a lower bound for $\|b - Ax_k\|_2$ if an upper bound on $\alpha$ is assumed.

**Lemma 4.2.** Let (4.2) be valid and let $\alpha \leq 1/2$. Then

\[ \|b - Ax_k\|_2 \geq \frac{1 - \alpha (1 + q_{\text{sat}})}{1 + q_{\text{sat}}} \| (A_k - \hat{A}_k)x_k \|_2. \]

**Proof.** The saturation assumption (4.2) yields

\[ \| (A_k - \hat{A}_k)x_k \|_2 \leq \| A_k x_k - Ax_k \|_2 + \| A_k x_k - \hat{A}_k x_k \|_2 \leq (1 + q_{\text{sat}}) \| A_k x_k - A_k x_k \|_2 \]

\[ \leq (1 + q_{\text{sat}}) (\| b - Ax_k \|_2 + \| b - \hat{A}_k x_k \|_2) \]

\[ \leq (1 + q_{\text{sat}}) (\| b - Ax_k \|_2 + (1 + q_{\text{sat}}) \| A_k - \hat{A}_k \|_2). \]

\[ \hfill \Box \]
In the following lemma, the convergence of the error estimator \( \eta_k \) is investigated. Its convergence is a consequence of the Dörfler marking strategy (4.3) (see also [16]). Notice that, due to the convergence of ACA, either

\[
\lim_{k \to \infty} (A_k)_b = A_b
\]

or \( b \notin M_k \) from some index \( k_0 \in \mathbb{N} \) on, which implies that \( (A_k)_b = (A_{k+1})_b \) for \( k \geq k_0 \). Hence,

\[
\lim_{k \to \infty} A_k = A_* \in \mathbb{R}^{N_x,N}
\]

and

\[
\lim_{k \to \infty} \|A_{k+1} - A_k\| = 0 = \lim_{k \to \infty} \|\hat{A}_{k+1} - \hat{A}_k\|. \tag{4.5}
\]

**Lemma 4.3.** Assume that \( A_* \) is invertible and \( a \) is sufficiently small. Then it holds that

\[
\eta_{k+1}^2 \leq q \eta_k^2 + z_k,
\]

where \( z_k \) converges to zero and \( q < 1 \). Furthermore, \( \lim_{k \to \infty} \eta_k = 0 \).

**Proof.** Using the definition of \( A_{k+1} \), \( \eta_{k+1}^2 \) reads

\[
\sum_{t \times s \in P} \| (A_{k+1} - \hat{A}_{k+1}) ts (x_{k+1}) s \|_2^2 = \sum_{t \times s \in M_k} \| (A_{k} - \hat{A}_{k}) ts (x_{k+1}) s \|_2^2 + \sum_{t \times s \notin M_k} \| (A_{k} - \hat{A}_{k}) ts (x_{k+1}) s \|_2^2.
\]

Due to Young’s inequality and (4.3), we have, for the second term,

\[
\sum_{t \times s \notin M_k} \| (A_{k} - \hat{A}_{k}) ts (x_{k+1}) s \|_2^2 \leq (1 + \epsilon) \eta_k^2 (P \setminus M_k) + (1 + 1/\epsilon) \sum_{t \times s \in M_k} \| (A_{k} - \hat{A}_{k}) ts (x_{k+1} - x_k) s \|_2^2 = (1 + \epsilon)(1 - \theta^2) \eta_k^2 + (1 + 1/\epsilon) \sum_{t \times s \in M_k} \| (A_{k} - \hat{A}_{k}) ts (x_{k+1} - x_k) s \|_2^2
\]

for all \( \epsilon > 0 \). The last sum can be estimated as

\[
\sum_{t \times s \in M_k} \| (A_{k} - \hat{A}_{k}) ts (x_{k+1} - x_k) s \|_2^2 = \sum_{s \in S} \sum_{t \times s \in M_k} \| (A_{k} - \hat{A}_{k}) ts (x_{k+1} - x_k) s \|_2^2 \leq \max_{s \in S} \| (A_{k} - \hat{A}_{k}) ts \|_2^2 \sum_{s \in S} \sum_{t \times s \in M_k} \| (x_{k+1} - x_k) s \|_2^2 \leq c_{sp} l \| x_{k+1} - x_k \|_2^2 \max_{s \in S} \| (A_{k} - \hat{A}_{k}) ts \|_2^2 \leq c_{sp} l \| A_{k+1} \|_2^2 \| (A_{k+1} x_{k+1} - A_k x_k) \|_2^2 \| x_{k+1} - x_k \|_2^2 + \| A_{k+1} - A_k \|_2 \| A_{k+1} - A_k \|_2 \| x_{k+1} - x_k \|_2^2.
\]

Due to \( \| b - A_k x_k \|_2 \leq a^2 \| (A_{k} - \hat{A}_k) x_k \|_2^2 \leq a^2 c_{sp} L \eta_k^2 \), we obtain, for the choice \( \epsilon := \frac{1}{2} \theta^2/(1 - \theta^2) \),

\[
\eta_{k+1}^2 \leq (1 - \frac{1}{2} \theta^2) \eta_k^2 + y [\| \eta_{k+1}^2 + \| z_k \|_2], \tag{4.7}
\]

where \( y := 3 a^2 \frac{2 - \theta^2}{\theta^2} (c_{sp} L \| A_{k} - A_{k+1} \|_2 \| A_{k+1}^{-1} \|_2^2) \) and

\[
z_k := \sum_{t \times x \in M_k} \| (A_{k} - \hat{A}_{k+1}) ts (x_{k+1}) s \|_2^2 + 3 (1 + 1/\epsilon) c_{sp} l L \| A_{k} - \hat{A}_{k} \|_2^2 \| A_{k+1}^{-1} \|_2^2 \| A_{k+1} - A_k \|_2 \| x_{k} \|_2^2.
\]

This follows from \( A_{k+1}^{-1} \to A_*^{-1} \) for \( k \to \infty \) and (4.5). Note that \( \{x_k\}_{k \in \mathbb{N}} \) is bounded as

\[
\| x_k \|_2 \leq \| A_k^{-1} \|_2 \| b \|_2 + \| A_k x_k - b \|_2 \leq \| A_k^{-1} \|_2 (\| b \|_2 + a \| (A_k - \hat{A}_k x_k) \|_2) \leq \| A_k^{-1} \|_2 \| b \|_2 + a \| A_k^{-1} \|_2 (\| A_k \|_2 + \| \hat{A}_k \|_2) |x_k|_2.
\]
and $a$ is sufficiently small. Choosing $a$ such that $y < \theta^2/4$, (4.7) leads to the first part of the assertion with

$$q := \frac{1 - \frac{1}{2} \theta^2 + y}{1 - y} < 1.$$  

For the proof of the second part, we pursue the ideas of the estimator reduction principle, which was introduced in [3]. Let $z > 0$ be a number with $z_k \leq z$ for all $k$. With the estimator reduction (4.6), it follows that

$$\eta_{k+1}^2 \leq q \eta_k^2 + z_k \leq q^2 \eta_{k-1}^2 + q z_{k-1} + z \leq \cdots \leq q^k \eta_0^2 + \sum_{i=0}^{k-1} q^{k-i} z_i$$

$$\leq q^{k+1} \eta_0^2 + z \sum_{i=0}^{k} q^i \leq \eta_0^2 + \frac{z}{1-q}.$$  

Hence, the sequence $\{\eta_k\}_{k \in \mathbb{N}_0}$ is bounded, and we define $M := \limsup_{k \to \infty} \eta_k^2$. Using the estimator reduction (4.6) once more leads to

$$M = \limsup_{k \to \infty} \eta_{k+1}^2 \leq q \limsup_{k \to \infty} \eta_k^2 + \limsup_{k \to \infty} z_k = qM.$$  

Hence, $M = 0$, and we obtain

$$0 \leq \liminf_{k \to \infty} \eta_k \leq \limsup_{k \to \infty} \eta_k = 0$$

and finally $\lim_{k \to \infty} \eta_k = 0$. \hfill \Box

**Theorem 4.4.** The residuals $r_k := b - Ax_k$ of the sequence $\{x_k\}_{k \in \mathbb{N}}$ constructed by Algorithm 2 converge to zero.

**Proof.** With Lemma 4.1 and Lemma 4.3, we obtain that

$$\|b - Ax_k\|_2 \leq \sqrt{c_{bp}^L \frac{1 + a(1 + q_{sat})}{1 - q_{sat}}} \eta_k \to 0.$$ \hfill \Box

## 5 Numerical Experiments

The numerical experiments are subdivided into three parts. At first, we investigate the storage requirements of BACA. After that, the quality of the proposed error estimator is discussed. Finally, the third part treats the acceleration of the matrix approximation and solution of the boundary value problem in comparison with ACA. In order to be able to compare these results with the results obtained from ACA, we prescribe the same error of $u_h$. Therefore,

$$e_h := \frac{\|u - u_h\|_{L^2}}{\|u\|_{L^2}}$$

describes the relative error of the solution $u_h$.

The approximation steps are executed without any parallelisation in both cases. We use the conjugate gradient method as a solver for the arising linear systems of equations without preconditioning. In the case of BACA, the accuracy of the iterative solver is adapted to the size of $\|A_k - \tilde{A}_k\|_2$; see Lemma 4.1. For all the experiments that are based on ACA, the accuracy of CG is set to $10^{-8}$.

As numerical examples, we consider a family of boundary value problems, in which the singularity of the right-hand side is moved towards the boundary of the computational domain $\Omega$, i.e.

$$-\Delta u(x) = 0 \quad \text{in } \Omega := B_1(0),$$

$$u(x) = S(x - p_i) \quad \text{on } \partial \Omega, \ i = 1, 2, 3, 4,$$

for $p_i = (x_i, 0, 0)^T$ with $x_1 = 10.0, x_2 = 1.5, x_3 = 1.1, x_4 = 1.05$. In these tests, we use the following parameters: the minimal block size $b_{\text{min}} = 15$, the blockwise accuracy $\epsilon_{\text{ACA}} = 10^{-6}$ of ACA, and the admissibility parameter $\beta = 0.8$.  

\[ \text{DE GRIYER} \]
5.1 Storage Reduction

We start with a uniform decomposition of the unit ball into 642 points and 1 280 triangles for the boundary value problems (5.1). Note that ACA does not take into account the right-hand side. Hence, the approximation does not depend on the choice of $p_i$. So we obtain the left picture of Figure 2 after the application of ACA to the discrete integral operator in each of the cases $i = 1, \ldots, 4$.

The proposed algorithm BACA provides the results shown in Table 1, in which the parameter $\theta = 0.9$ was used. Here, the ranks of $\hat{A}_k$ are above $A_k$ by two ACA steps. The parameter $\alpha$ from Lemma 4.1 is set to 100. Since less accurate results can be expected when the singularity is moved towards the boundary (see the column labelled $e_h$), we have to use different tolerances $\varepsilon_{\text{BACA}}$. Compared with the approximation obtained from ACA, we obtain lower storage requirements for the matrix approximation $\hat{A}_k$.

The storage reduction and the lower relative storage (storage for the approximation divided by the storage for $A$) observed in Table 1 are also visible from Figure 2, where the respective approximation $\hat{A}_k$ of $A$ with its blockwise ranks is shown (green blocks). Therein, red blocks are constructed entry-wise without approximation no matter if they are admissible or not. Red admissible blocks have to be fully computed in order to reach the desired accuracy.

Example (5.1) has been chosen in order to investigate different right-hand sides which lead to structural differences in the respective solution. Notice that the influence of the singularity on the solution is stronger the smaller the distance of $p_i$ to the boundary of $\Omega$ becomes. Hence, for $p_i$ close to the boundary, certain parts of the discrete integral operator are more important for the accuracy of the solution than others. Even for a relatively large distance, i.e. $p_1 = (10, 0, 0)^T$, smaller ranks and hence a better compression rate than ACA can be observed; see Figure 2. This effect is even stronger if $p_i$ approaches the boundary. The average
Table 2: Average ranks for the matrices $\hat{A}_k$ and $A_k$ of BACA for $p_1, \ldots, p_4$ compared with ACA.

| matrix | $p_1$ | $p_2$ | $p_3$ | $p_4$ |
|--------|-------|-------|-------|-------|
| $A_k$  | 12.54 | 7.44  | 5.00  | 3.71  |
| $\hat{A}_k$ | —    | 8.52  | 6.62  | 5.38  | 5.01  |

Table 3: Number of computed entries, storage and relative storage for the approximation constructed by BACA for $p_3$.

| $N$    | $\varepsilon_{\text{BACA}}$ | $\|b - A_h x_k\|_2$ | $e_h$ | number of computed entries | storage (MB) | rel. stor. (%) |
|--------|-----------------------------|---------------------|-------|----------------------------|--------------|----------------|
| 1280   | 1e-04                       | 4.14e-04            | 0.264 | 3.22e05                    | 2.48         | 39.7           |
| 7168   | 1e-05                       | 2.31e-05            | 0.077 | 3.32e06                    | 25.47        | 13.0           |
| 28672  | 1e-06                       | 2.56e-05            | 0.023 | 2.10e07                    | 160.98       | 5.1            |
| 114688 | 1e-06                       | 2.13e-05            | 0.007 | 8.76e07                    | 837.92       | 1.7            |

Table 4: Number of computed entries, storage and relative storage for the approximation constructed by ACA for $p_3$.

| $N$    | $e_h$ | number of computed entries | storage (MB) | rel. stor. (%) |
|--------|-------|----------------------------|--------------|----------------|
| 1280   | 0.264 | 5.01e05                    | 3.85         | 61.5           |
| 7168   | 0.077 | 4.91e06                    | 37.53        | 19.8           |
| 28672  | 0.023 | 2.66e07                    | 203.70       | 6.5            |
| 114688 | 0.007 | 1.35e08                    | 1034.56      | 2.1            |

ranks (the sum of the ranks of all blocks divided by number of blocks) are shown in Table 2. So Table 1 and Figure 2 indicate that the proposed error estimator detects those matrix blocks which are important for the respective problem; see Figure 3. Therefore, BACA is able to construct an $H$-matrix approximation that is particularly suited to solve a linear system for a specific right-hand side.

The improved storage requirements can be observed also for finer triangulations of the boundary. Table 3 shows the results for the boundary value problem (5.1) in the case $i = 3$ for several numbers of degrees of freedom $N$. In addition to the relative storage and the storage requirements, the number of computed entries for the construction of $A_h$ are presented. Since the solution $u_h$ is expected to be more accurate the larger $N$ is, the accuracy $\varepsilon_{\text{BACA}}$ of the error estimator has to be adapted to $N$. All other parameters remain unchanged. In comparison, the results of ACA applied to the boundary value problem (5.1) in the case $i = 3$ can be seen in Table 4.
Hence, BACA requires significantly less original matrix entries than the usual construction via ACA of the \( \mathcal{T} \)-matrix approximation to obtain approximately the same relative error of \( u_h \).

### 5.2 Quality of the Error Estimator

In the following tests, we validate the error estimator \( \eta_k \) introduced in (4.4). At first, the reliability statement of Lemma 4.1 together with the lower bound \( \| (A_k - \hat{A}_k)x_k \|_2 \) of Lemma 4.2 is investigated. Again, the boundary element method combined with BACA is tested with the problem described in (5.1) and a triangulation with 14338 points and 28672 triangles. The accuracy \( \varepsilon_{\text{BACA}} \) of the error estimator is set to \( 10^{-7} \). We start BACA with a coarse approximation \( A_0 \) which has been generated by applying four ACA steps to each block. Table 5 and Figure 4 contain the results for \( \alpha = 1/2 \) and three refinement parameters \( \theta \). The ranks of \( \hat{A}_k \) are ahead of \( A_k \) by three ACA steps.

The proposed error estimator \( \eta_k \) estimates the residual error \( \| b - Ax_k \|_2 \) in an appropriate way. The expression \( \frac{1}{2} \| (A_k - \hat{A}_k)x_k \|_2 \) can serve as a lower bound for the residual error. Table 5 shows that the convergence with respect to \( k \) is the faster the refinement parameter \( \theta \) is closer to 1. A larger parameter \( \theta \) in (4.3) leads to a larger set \( M_k \) of marked blocks, which is likely to produce redundant information in \( A_{k+1} \). Conversely, small refinement parameters \( \theta \) usually lead to matrix approximations with lower storage requirements.

The numerical results confirm the theoretical findings of Lemma 4.1 and Lemma 4.2. The proposed error estimator \( \eta_k \) is reliable, and \( \frac{1}{2} \| (A_k - \hat{A}_k)x_k \|_2 \) can be used as a lower bound on the residual error.

**Figure 4:** The error estimator, the residual and the lower bound \( \| (A_k - \hat{A}_k)x_k \|_2 \) for several parameters \( \theta \).
5.3 Acceleration of the Matrix Approximation

The results represent only a part of our aims of the extension of ACA. As already mentioned in the introduction, we are also interested in an acceleration of the matrix approximation and an acceleration of the solution process. Since the computation of matrix entries is by far the most time-consuming part of the boundary element method and we have seen that BACA requires less entries, we can expect improved computational times.

We consider again the family of boundary value problems (5.1). The ranks of $\hat{A}_k$ are ahead of $A_k$ by two ACA steps, and we choose $\alpha = 100$. The ratio of the time required for BACA and the time of the ACA can be observed in Figures 5 and 6 for two different triangulations. The two figures show that we obtain a lower time consumption even in the case of $p_1$, where the structural differences in the right-hand side are low. If the singularity is moved closer to the boundary of $\Omega$, BACA speeds up continuously.

One of the biggest advantages of ACA is its linear logarithmic complexity. In order to observe this behaviour, the sphere discretised with 642 points and 1280 triangles is considered. We keep the resulting geometry and increase the number of triangles $N$ for the boundary value problem with $p_1$. The time columns of Table 6 show the expected complexity of ACA and also of the new BACA. The growth factor of $N$ is four and the growth factor of the computational time is between five and six.

Table 5: Numerical results for $\eta_k$ compared with the residual for different $\theta$.

| $k$ | $\theta = 0.6$ | $\theta = 0.7$ | $\theta = 0.9$ | $\theta = 0.6$ | $\theta = 0.7$ | $\theta = 0.9$ |
|-----|----------------|----------------|----------------|----------------|----------------|----------------|
| 1   | 1.36e-05       | 1.36e-05       | 1.36e-05       | 1.52e-05       | 1.41e-05       | 1.07e-05       |
| 2   | 1.08e-05       | 9.77e-06       | 6.07e-06       | 1.31e-05       | 1.19e-05       | 5.68e-06       |
| 3   | 8.49e-06       | 6.96e-06       | 2.71e-06       | 1.18e-05       | 9.45e-06       | 3.16e-06       |
| 4   | 6.78e-06       | 4.98e-06       | 1.20e-06       | 9.78e-06       | 7.05e-06       | 1.59e-06       |
| 5   | 5.41e-06       | 3.57e-06       | 5.60e-07       | 8.24e-06       | 5.55e-06       | 6.68e-07       |
| 6   | 4.33e-06       | 2.58e-06       | 2.56e-07       | 6.87e-06       | 4.22e-06       | 3.46e-07       |
| 7   | 3.47e-06       | 1.85e-06       | 1.19e-07       | 5.93e-06       | 3.39e-06       | 1.91e-07       |
| 8   | 2.80e-06       | 1.32e-06       | 5.96e-08       | 4.90e-06       | 2.42e-06       | 1.04e-07       |
| 9   | 2.24e-06       | 9.52e-07       | —              | 4.15e-06       | 1.85e-06       | —              |
| 10  | 1.79e-06       | 6.96e-07       | —              | 3.53e-06       | 1.55e-06       | —              |
| 11  | 1.44e-06       | 5.04e-07       | —              | 3.11e-06       | 1.03e-06       | —              |
| 12  | 1.15e-06       | 3.64e-07       | —              | 2.37e-06       | 6.84e-07       | —              |
| 13  | 9.31e-07       | 2.65e-07       | —              | 1.99e-06       | 5.19e-07       | —              |
| 14  | 7.53e-07       | 1.92e-07       | —              | 1.70e-06       | 4.11e-07       | —              |
| 15  | 6.07e-07       | 1.40e-07       | —              | 1.34e-06       | 3.19e-07       | —              |

Figure 5: Time ratio (BACA/ACA) for a triangulation with 7168 triangles.

Figure 6: Time ratio (BACA/ACA) for a triangulation with 28672 triangles.
Effects on Partially Overrefined Surfaces

In the following tests, we apply ACA and BACA to triangulations which are partially overrefined. The boundary of the ellipsoid \( \Omega = \{ x \in \mathbb{R}^3 : x_1^2 + x_2^2 + x_3^2/9 = 1 \} \) is more refined on the mid-region than on the remaining geometry; see Figure 7.

The numerical results of BACA for the ellipsoid \( \rho_2, \varepsilon_{BACA} = 10^{-6} \) with a growing number of triangles are contained in Table 7. The accuracy of the block approximation for ACA is \( \varepsilon_{ACA} = 10^{-6} \). The minimal block size is also given in Table 7, which we increase with the growing size of the problem. In order to compare the results, we keep the relative error at the same level.

Inspecting the time columns of Tables 7 and 8 reveals that BACA is significantly faster than ACA. For the largest problem, BACA is more than twice as fast as ACA. The reason for the acceleration is that the solution does not benefit from the local over-refinement. While the solver based on ACA is not able to detect this,
the combination of solution and error estimation used in BACA leads to lower number of CG iterations and computational time. For the largest problem the number of CG iterations required for the solution based on ACA is more than twice as large as for the solution based on BACA. Note that this situation persists if an optimal preconditioner is employed, which leads to a bounded number of CG iterations with respect to \( N \) in both cases. We obtain no significant difference in the storage requirements after the application of the two algorithms. Hence, the observed speed-up results from the coupling of the iterative solver and the error estimator rather than from the reduction of the storage requirements. In addition to these facts, a decreasing number of required iterations \( k \) can be observed with a growing number of unknowns \( N \).

The ratio of the time required for ACA and for BACA can be seen in Figure 8. The curves stabilise at a certain constant, which seems to depend on the geometry and the number and the location of refined areas.

6 Conclusion

In this article, we have introduced a new method which extends the adaptive cross approximation (ACA) to a block-adaptive version (BACA) for a setting in which only a single linear system is to be solved efficiently. In such a situation, the matrix approximation via blockwise low-rank matrices can be adapted to the right-hand side. To identify those blocks which are important for the residual error, we employ techniques known from adaptive mesh refinement and prove convergence of the adaptive procedure. The new method is validated by numerical experiments which show that a significant reduction in storage and computational time can be achieved in situations where the solution in some sense contains structural differences and situations that result from locally over-refined meshes. The combination of BACA with adaptive grid refinement will be investigated in future work.

Funding: This work was supported by the DFG project BE2626/4-1.

References

[1] M. Ainsworth and C. Glusa, Aspects of an adaptive finite element method for the fractional Laplacian: A priori and a posteriori error estimates, efficient implementation and multigrid solver, *Comput. Methods Appl. Mech. Engrg.* 327 (2017), 4–35.
[2] M. Aurada, M. Feischl, T. Führer, M. Karkulik and D. Praetorius, Efficiency and optimality of some weighted-residual error estimator for adaptive 2D boundary element methods, *Comput. Methods Appl. Math.* 13 (2013), no. 3, 305–332.
[3] M. Aurada, S. Ferraz-Leite and D. Praetorius, Estimator reduction and convergence of adaptive BEM, *Appl. Numer. Math.* 62 (2012), no. 6, 787–801.
[4] M. Bebendorf, Approximation of boundary element matrices, *Numer. Math.* 86 (2000), no. 4, 565–589.
[5] M. Bebendorf, Efficient inversion of the Galerkin matrix of general second-order elliptic operators with nonsmooth coefficients, *Math. Comp.* 74 (2005), no. 251, 1179–1199.
[6] M. Bebendorf, Approximate inverse preconditioning of finite element discretizations of elliptic operators with nonsmooth coefficients, *SIAM J. Matrix Anal. Appl.* 27 (2006), no. 4, 909–929.
[7] M. Bebendorf, Hierarchical Matrices, Lect. Notes Comput. Sci. Eng. 63, Springer, Berlin, 2008.
[8] M. Bebendorf, M. Bollhöfer and M. Bratsch, On the spectral equivalence of hierarchical matrix preconditioners for elliptic problems, *Math. Comp.* 85 (2016), no. 302, 2839–2861.
[9] M. Bebendorf and R. Grzhibovskis, Accelerating Galerkin BEM for linear elasticity using adaptive cross approximation, *Math. Methods Appl. Sci.* 29 (2006), no. 14, 1721–1747.
[10] M. Bebendorf and W. Hackbusch, Stabilized rounded addition of hierarchical matrices, *Numer. Linear Algebra Appl.* 14 (2007), no. 5, 407–423.
[11] M. Bebendorf and R. Kriemann, Fast parallel solution of boundary integral equations and related problems, *Comput. Vis. Sci.* 8 (2005), no. 3–4, 121–135.
[12] M. Bebendorf and S. Rjasanow, Adaptive low-rank approximation of collocation matrices, *Computing* 70 (2003), no. 1, 1–24.
[13] Y. Chen, A fast, direct algorithm for the Lippmann–Schwinger integral equation in two dimensions, *Adv. Comput. Math.* 16 (2002), no. 2–3, 175–190.
[14] M. D’Elia and M. Gunzburger, The fractional Laplacian operator on bounded domains as a special case of the nonlocal diffusion operator, *Comput. Math. Appl.* 66 (2013), no. 7, 1245–1260.
[15] E. Di Nezza, G. Palatucci and E. Valdinoci, Hitchhiker’s guide to the fractional Sobolev spaces, *Bull. Sci. Math.* 136 (2012), no. 5, 521–573.
[16] W. Dörfler, A convergent adaptive algorithm for Poisson’s equation, *SIAM J. Numer. Anal.* 33 (1996), no. 3, 1106–1124.
[17] S. Ferraz-Leite and D. Praetorius, Simple a posteriori error estimators for the $h$-version of the boundary element method, *Computing* 83 (2008), no. 4, 135–162.
[18] T. Gantumur, Adaptive boundary element methods with convergence rates, *Numer. Math.* 124 (2013), no. 3, 471–516.
[19] S. A. Goreinov, E. E. Tyrtyshnikov and N. L. Zamarashkin, A theory of pseudoskeleton approximations, *Linear Algebra Appl.* 261 (1997), 1–21.
[20] L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, *J. Comput. Phys.* 73 (1987), no. 2, 325–348.
[21] L. Greengard and J. Strain, The fast Gauss transform, *SIAM J. Sci. Statist. Comput.* 12 (1991), no. 1, 79–94.
[22] W. Hackbusch, A sparse matrix arithmetic based on $\mathcal{H}$-matrices. I. Introduction to $\mathcal{H}$-matrices, *Computing* 62 (1999), no. 2, 89–108.
[23] W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis*, Springer Ser. Comput. Math. 49, Springer, Heidelberg, 2015.
[24] W. Hackbusch and B. N. Khoromskij, A sparse $\mathcal{H}$-matrix arithmetic. II. Application to multi-dimensional problems, *Computing* 64 (2000), no. 1, 21–47.
[25] M. Karkulik, G. Of and D. Praetorius, Convergence of adaptive 3D BEM for weakly singular integral equations based on isotropic mesh-refinement, *Numer. Methods Partial Differential Equations* 29 (2013), no. 6, 2081–2106.
[26] F. Lanzara, V. Maz’ya and G. Schmidt, Numerical solution of the Lippmann–Schwinger equation by approximate approximations, *J. Fourier Anal. Appl.* 10 (2004), no. 6, 645–660.
[27] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin and M. Tygert, Randomized algorithms for the low-rank approximation of matrices, *Proc. Natl. Acad. Sci. USA* 104 (2007), no. 51, 20167–20172.
[28] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.
[29] S. A. Sauter and C. Schwab, *Boundary Element Methods*, Springer Ser. Comput. Math. 39, Springer, Berlin, 2011.
[30] O. Steinbach, *Numerical Approximation Methods for Elliptic Boundary Value Problems*, Springer, New York, 2008.
[31] E. Tyrtyshnikov, Mosaic-skeleton approximations. Toeplitz matrices: Structures, algorithms and applications, *Calcolo* 33 (1996), no. 1–2, 47–57.