A Parallel Fast Boundary Element Method Using Cyclic Graph Decompositions *

Dalibor Lukáš†, Petr Kovář†, Tereza Kovářová†, and Michal Merta†

Abstract

We propose a method of a parallel distribution of densely populated matrices arising in boundary element discretizations of partial differential equations. In our method the underlying boundary element mesh consisting of n elements is decomposed into N submeshes. The related $N \times N$ submatrices are assigned to N concurrent processes to be assembled. Additionally we require each process to hold exactly one diagonal submatrix, since its assembling is typically most time consuming when applying fast boundary elements. We obtain a class of such optimal parallel distributions of the submeshes and corresponding submatrices by cyclic decompositions of undirected complete graphs. It results in a method the theoretical complexity of which is $O((n/\sqrt{N})\log(n/\sqrt{N}))$ in terms of time for the setup, assembling, matrix action, as well as memory consumption per process. Nevertheless, numerical experiments up to $n = 2744832$ and $N = 273$ on a real-world geometry document that the method exhibits superior parallel scalability $O((n/N)\log n)$ of the overall time, while the memory consumption scales accordingly to the theoretical estimate.

*This work was supported by the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the project SPOMECH - Creating a multidisciplinary R&D team for reliable solution of mechanical problems (CZ.1.07/2.3.00/20.0070) funded by the European Regional Development Fund and the national budget of the Czech Republic via the Research and Development for Innovations Operational Programme, as well as Czech Ministry of Education, Youth and Sports via the project Large Research, Development and Innovations Infrastructures (LM2011033). The work was also supported by VŠB—Technical University of Ostrava under the grant SGS SP2013/191.

†VŠB–Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic. Emails: {dalibor.lukas,petr.kovar,tereza.kovarova,michal.merta}@vsb.cz

Keywords: boundary element method, parallel computing, graph decomposition

1 Introduction

Boundary element methods (BEM) [RS07] have become an efficient tool for solution of partial differential equations. When compared to more popular volume discretization techniques, BEM eliminates interior degrees of freedom and reduces the problem formulation to the boundary, which is particularly efficient in case of unbounded computational domains or in shape optimization [EH06, LPZ12]. Unfortunately, when implementing BEM, we have to deal with two difficulties: an integration of singular kernels and a dense matter of system matrices.

Concerning the singular integrals, semi-analytical integration techniques have been developed [RS07] for many types of kernels. Here one calculates inner collocation integrals analytically while quadrature rules are employed for outer Galerkin integrals. Alternatively, the whole integration domain is decomposed into simplices, singularities are transferred to corners and then removed by Duffy's substitution [Duf82]. The resulting regularized kernels are proved to be analytical [SS10] so that a Gauss quadrature converges exponentially.

As far as the density of system matrices is considered, a lot of effort has been devoted to their sparsification, which reduces the original quadratic complexity to linear or almost linear $O(n \log n)$, where n denotes the number of degrees of freedom. Such methods are then referred to as fast BEM. The idea of sparsification relies on the fact that for far field contributions to the system matrix the integral kernel is smooth and it can be replaced by low-rank approximations. This technique goes back to the fast multipole method [Rok85] or the panel clustering [HN89], where low-rank function approximations were proposed and analyzed. We can also proceed in a pure algebraic manner [Beb00] in terms of low-rank matrix approximants. Finally, several approaches have already been proposed for preconditioning [MT97, PS92] of resulting systems so that the overall computational complexity of an iterative solution method is still $O(n \log n)$.

Yet, a parallel implementation of the boundary element method remains an issue. There are many papers, cf. [GKS98], dealing with parallel implementations of various tree algorithms such as Burnes-Hut method [BH86] or

the fast multipole method, but, to our best knowledge, only one paper [BK05] is dealing with efficient parallel assembling and action of the matrix. The general problem is an optimal assignment of leaves (jobs) of the tree to processes. The simplest but load-unbalanced method is the list scheduling, which assigns next not yet executed job to the idle process. On the other hand, a well load-balanced method is the largest process time (LPT), where jobs are sorted according to their costs. However, LPT suffers from an expensive setup and a parallely unscalable complexity of the matrix action, which is due to that the maximum number of processes sharing a row or a column, the so-called sharing constant, is $O(N)$. As a remedy Bebendorf and Kriemann in [BK05] propose a global sorting of elements by utilizing space-filling curves [Sag94]. The latter in combination with sequence partitioning [OM95] leads to a reduction of the sharing constant to $O(\sqrt{N})$, which is in [BK05] numerically documented to lead to the optimal scalability $O((n/N)\log n + n/\sqrt{N})$ of the matrix assembly as well as matrix-vector product. However, the scheduling phase suffers from $O(n)$ memory consumption and $O(N(n-N))$ computational complexity for the sequence partitioning [OM95].

In this paper we propose a novel approach, which we prove to enjoy the parallel scalability $O((n/\sqrt{N})\log(n/\sqrt{N}))$ of the memory consumption per process, time for the setup, matrix action as well as matrix assembly, while the latter prevails. Numerical experiments exhibit the superior scalability $O((n/N)\log(n))$ of the total time. We consider a decomposition of the underlying mesh into N parts, subsequently, the system matrix is decomposed into $N \times N$ blocks. We employ N concurrent processes, to each of which we assign N blocks of the matrix to be assembled. The assignment is done in such a way that we minimize the amount of the related mesh parts, thus, the total memory consumption for storing the mesh and related structures is minimal. Additionally, in order to balance the load each process holds exactly one diagonal block since these are typically most time and memory consuming within a fast BEM. It turns out that the problem can be formulated in terms of graph theory as a decomposition of an undirected complete graph K_N into N complete subgraphs K_M, where $M(M-1)+1 = N$. This is a combinatorial complexity optimization problem. Fortunately, when one restricts to cyclic decompositions the optima are known for certain values of $N = 3, 7, 13, 21, 31$, etc. Provided that each submesh consists of n/N elements, the sharing constant takes the optimal value $M = \frac{1}{2}(1 + \sqrt{4N-3})$, see (10). We base our parallel fast BEM algorithm on these graph decompositions.

The rest of the paper is organized as follows. In Section 2 we recall a boundary element discretization of the Laplace equation in 3 dimensions. In Section 3 we describe two fast BEM techniques. In Section 4 we give an introduction to cyclic decompositions of graphs and list some known optimal results, on top of which we build our parallel implementation in Section 5. In Section 6 we document the expected theoretical complexity on numerical tests using geometry of a shaft up to almost 3 millions of degrees of freedom and 273 processes. In Section 7 we conclude.

2 Boundary element methods

For simplicity we consider a 3-dimensional bounded Lipschitz domain Ω. We look for a solution $u \in H^1(\Omega)$ to the following Dirichlet boundary value problem for the Laplace operator:

$$\begin{aligned} -\triangle u &= 0 \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma := \partial\Omega, \end{aligned} \tag{1}$$

where $g \in H^{1/2}(\Gamma)$ denotes a prescribed Dirichlet datum. Our exposition can be easily modified to more general settings as those involving mixed boundary conditions, exterior domains as well as to other elliptic operators, e.g., Lamé, Stokes, Helmholtz, Maxwell, for which boundary integral formulations are available.

The solution of (1) can be represented as follows:

$$u(x) = \int_\Gamma \frac{\partial u}{\partial n}(y)\, G(x,y)\, dS(y) - \int_\Gamma u(y)\, \frac{\partial G}{\partial n(y)}(x,y)\, dS(y), \quad x \in \Omega, \tag{2}$$

where $G(x,y) := 1/(4\pi\|x-y\|)$ denotes the fundamental solution to the Laplace equation. The terms on the right-hand side of (2) are referred to as the single-layer operator $\tilde{V}$ and the double-layer operator W, respectively. They can be extended continuously to linear bounded operators $\tilde{V} : H^{-1/2}(\Gamma) \to H^1(\Omega)$ and $W : H^{1/2}(\Gamma) \to H^1(\Omega)$. It remains to calculate the Neumann datum $t := \frac{\partial u}{\partial n} \in H^{-1/2}(\Gamma)$. We can apply the trace operator $\gamma_{\mathrm{D}} : H^1(\Omega) \to H^{1/2}(\Gamma)$ to both sides of (2) and we arrive at the first-kind boundary integral equation

$$u(x) = V(t)(x) - \left(-\frac{1}{2}u(x) + K(u)(x)\right),$$

where $V := \gamma_{\mathrm{D}} \circ \tilde{V} : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma)$ and $K : H^{1/2}(\Gamma) \to H^{1/2}(\Gamma)$ are linear continuous operators, additionaly, the former is $H^{-1/2}(\Gamma)$-elliptic. For $t, u \in L^\infty(\Gamma)$ these operators take the following forms:

$$V(t)(x) := \int_\Gamma t(y) G(x,y)\, dS(y), \quad K(u)(x) := \int_\Gamma u(y) \frac{\partial G}{\partial n(y)}(x,y)\, dS(y),$$

respectively, where $x \in \Gamma$ and the integrals are considered in the Lebesgue sense. Let $\langle ., . \rangle$ denote the duality pairing between $H^{-1/2}(\Gamma)$ and $H^{1/2}(\Gamma)$ with respect to the pivot space $L^2(\Gamma)$. We shall find the missing Neumann datum $t \in H^{-1/2}(\Gamma)$ by means of the following well-posed weak formulation:

$$\langle v, V(t) \rangle = \langle v, ((1/2)I + K)(g) \rangle \quad \forall v \in H^{-1/2}(\Gamma). \tag{3}$$

Now we shall approximate the solution of (3) by a boundary element method. Without a loss of generality, let Γ be polygonal and let $\tau^h = (T_i)_{i=1}^n$ be its shape-regular triangulation into n triangles. We approximate the space $H^{-1/2}(\Gamma)$ by its finite-dimensional subspace V^h consisting of the discontinuous functions that are piecewise constant on τ^h. We introduce the standard element base $(\psi_i^h(x))_{i=1}^n$ of V^h. For the sake of simplicity, we approximate the space $H^{1/2}(\Gamma)$ by V^h again. The approximation is nonconforming in sense of $V^h \not\subset H^{1/2}(\Gamma)$. Assume we are given $g^h \in V^h$ as an approximation of g. The latter introduces an additional error, which can be analyzed by means of Strang's lemma. Let us denote the coordinates of g^h in the element base by $\mathbf{g} \in \mathbb{R}^n$. The Galerkin approximation of (3) then leads to the following linear system of equations:

$$\mathbf{V}\,\mathbf{t} = ((1/2)\mathbf{M} + \mathbf{K})\,\mathbf{g}, \tag{4}$$

where $\mathbf{t} \in \mathbb{R}^n$ are the coordinates of the approximated solution $t^h(x) \in V^h$, the diagonal matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ with entries $(\mathbf{M})_{i,i} = |T_i|$, where by $|T_i|$ we denote the area of T_i, is the Galerkin approximation of the identity I, and the matrices $\mathbf{V}, \mathbf{K} \in \mathbb{R}^{n \times n}$ are the Galerkin approximations of V, K

$$(\mathbf{V})_{i,j} = \int_{T_i} \int_{T_j} G(x,y)\, dS(y)\, dS(x), \; (\mathbf{K})_{i,j} = \int_{T_i} \int_{T_j} \frac{\partial G}{\partial n(y)}(x,y)\, dS(y)\, dS(x). \tag{5}$$

3 Fast boundary element methods

The matrices $\mathbf{V}$ and $\mathbf{K}$ are densely populated and we shall approximate them by means of hierarchical matrices [Beb08]. For this purpose we introduce a

hierarchical clustering of τ^h. On the first level, we decompose τ^h into two disjoint clusters $\mathcal{C}_1, \mathcal{C}_2$ by a separating plane. The plane crosses the boundary mass center

$$c(\tau^h) := \frac{1}{|\Gamma|} \int_\Gamma x \, dS(x) = \frac{1}{m\,|\Gamma|} \sum_{T_k \in \tau^h} |T_k| \, c_k, \tag{6}$$

where c_k denotes the mid-point of T_k. The normal vector to the plane is an eigenvector related to the largest eigenvalue of the 3-by-3 covariance matrix

$$\begin{aligned} (\mathbf{C}(\tau^h))_{ij} &:= \sum_{T_k \in \tau^h} |T_k| \, (c_k - c(\tau^h))_i \, (c_k - c(\tau^h))_j \\ &\approx \int_\Gamma (x - c(\tau^h))_i \, (x - c(\tau^h))_j \, dS(x). \end{aligned} \tag{7}$$

In other words, the normal direction is, up to a quadrature error, the least inertia axis. Further we proceed recursively. At the next level the decomposition is applied to the cluster $\mathcal{C}_1$, i.e. the separating plane is determined by $c(\mathcal{C}_1)$ and $\mathbf{C}(\mathcal{C}_1)$ using (6) and (7), respectively, so that we arrive at $\mathcal{C}_{11}$, $\mathcal{C}_{12}$ and the decomposition of $\mathcal{C}_2$ results in $\mathcal{C}_{21}$, $\mathcal{C}_{22}$. The recursion stops when the number of triangles in a cluster to be decomposed is less than or equal to a prescribed $n_{\min}$.

The resulting binary tree of clusters generates a quad-tree of submatrices of $\mathbf{V}$. The idea of sparse approximation of $\mathbf{V}$ by a hierarchical matrix relies on the observation that submatrices of $\mathbf{V}$ related to well-separated cluster pairs can be well-approximated by a low-rank matrix. Such pairs of clusters $(\mathcal{C}_x, \mathcal{C}_y)$ are called admissible and we indicate the admissibility by the following condition:

$$\min\{\operatorname{diam} \mathcal{C}_x, \operatorname{diam} \mathcal{C}_y\} \le \eta \operatorname{dist}(\mathcal{C}_x, \mathcal{C}_y),$$

where $\eta < 1$ is given, $\operatorname{diam} \mathcal{C}$ and $\operatorname{dist}(\mathcal{C}_x, \mathcal{C}_y)$ respectively denote the diameter of $\mathcal{C}$ and the distance between $\mathcal{C}_x$ and $\mathcal{C}_y$. Unfortunately, evaluation of the condition has a quadratic complexity, therefore, we replace it by the following stronger admissibility condition with a linear complexity:

$$2 \min\{\operatorname{rad} \mathcal{C}_x, \operatorname{rad} \mathcal{C}_y\} \le \eta \, (\operatorname{dist}(c(\mathcal{C}_x), c(\mathcal{C}_y)) - \operatorname{rad} \mathcal{C}_x - \operatorname{rad} \mathcal{C}_y), \tag{8}$$

where $\operatorname{rad} \mathcal{C} := \max_{T_k \in \mathcal{C}} \max_{x \in T_k} \|x - c(\mathcal{C})\|$.

Based on (8) and the binary tree of clusters we decompose the matrix $\mathbf{V}$ into blocks. They are either nonadmissible, in the case a block is related to a nonadmissible pair of leaves of the cluster tree, or admissible if it is related to two vertices of the cluster tree that satisfy (8). Moreover, in order to sparsify $\mathbf{V}$ efficiently, the admissible blocks should be as large as possible, it means that the parents of the related admissible pairs of clusters creates a nonadmissible pair. The nonadmissible blocks are assembled as full matrices, however, they can be glued together and stored as a sparse matrix $\mathbf{V}^{\text{non}} \in \mathbb{R}^{n\times n}$.

3.1 Fast multipole method

We shall approximate each admissible block by a low-rank matrix. First we describe the fast multipole method (FMM) [Rok85]. It relies on an expansion of the integral kernel $G(x,y)$ into spherical harmonics. For a point x^* with $\|x-x^*\| < \|y-x^*\|$ we have

$$G(x,y) = \frac{1}{4\pi\|y-x^*\|} \sum_{k=0}^{\infty} \left(\frac{\|x-x^*\|}{\|y-x^*\|}\right)^k P_k\left(\frac{x-x^*}{\|x-x^*\|}, \frac{y-x^*}{\|y-x^*\|}\right),$$

where the Legendre polynomials $P_k(e(x), e(y))$ admit separation of variables

$$P_k(e(x), e(y)) = \sum_{m=-k}^{k} Y_k^m(e(x))\, Y_k^{-m}(e(y)),$$

with Y_k^m being the spherical harmonic functions. We approximate $G(x,y)$ by a finite sum $G_p(x,y)$, for which we have the following error estimate:

$$\left| G(x,y) - \underbrace{\frac{1}{4\pi\,\|y-x^*\|} \sum_{k=0}^{p} \left(\frac{\|x-x^*\|}{\|y-x^*\|}\right)^k P_k\left(\frac{x-x^*}{\|x-x^*\|}, \frac{y-x^*}{\|y-x^*\|}\right)}_{=:G_p(x,y)} \right| \leq \frac{1}{4\pi\,(\|y-x^*\| - \|x-x^*\|)} \left(\frac{\|x-x^*\|}{\|y-x^*\|}\right)^{p+1}.$$

Provided $\operatorname{rad}\mathcal{C}_x \leq \operatorname{rad}\mathcal{C}_y$, $x^* := c(\mathcal{C}_x)$, the error can further be estimated by

$$\frac{1}{4\pi\,(\|c(\mathcal{C}_x) - c(\mathcal{C}_y)\| - \operatorname{rad}\mathcal{C}_y)} \left[\eta\left(1 - \frac{\operatorname{rad}\mathcal{C}_x + 2\operatorname{rad}\mathcal{C}_y}{\|c(\mathcal{C}_x) - c(\mathcal{C}_y)\| + \operatorname{rad}\mathcal{C}_y}\right)\right]^{p+1}.$$

The approximation with a precision $\varepsilon > 0$ requires $p = O(\log \varepsilon / \log \eta)$ terms in the sum, each of which admits the separation of variables

$$G_p(x,y) = \frac{1}{4\pi} \sum_{k=0}^{p} \sum_{m=-k}^{k} \left(\|x - x^*\|^k \, Y_k^m(e(x)) \right) \left(\|y - x^*\|^{-k-1} \, Y_k^{-m}(e(y)) \right).$$

Therefore, the double integral can be separated into a product of two single integrals and the matrix-vector multiplication $\mathbf{s} = \mathbf{V}\,\mathbf{t}$ can be evaluated by

$$(\mathbf{s})_i = \sum_{j \in \mathrm{NF}(i)} (\mathbf{V})_{i,j} \, (\mathbf{t})_j + \sum_{k=0}^{p} \sum_{m=-n}^{n} \hat{M}_k^m(O, \psi_i) \tilde{L}_k^m(O, \mathrm{FF}(i)).$$

Here

$$\hat{M}_k^m(O, \psi_i) = \int_\Gamma \|x - x^*\|^{k-1} \, Y_k^m(e(x)) \psi_i(x) dS(x)$$

are the multipole coefficients associated with the element T_i, and

$$\tilde{L}_k^m(O, \mathrm{FF}(i)) = \sum_{j \in \mathrm{FF}(i)} \frac{(\mathbf{t})_j}{4\pi} \int_\Gamma \|y - x^*\|^{-k} \, Y_k^{-m}(e(y)) \psi_j(y) ds(y)$$

are the coefficients of local expansion. The set of admissible and nonadmissible clusters to the cluster containing the element T_i is denoted by $\mathrm{FF}(i)$ and $\mathrm{NF}(i)$, respectively. The efficient computation of $\tilde{L}_k^m(O, \mathrm{FF}(i))$ exploits the existing tree structure:

1. *Upward pass* – multipole moments are computed on the finest level of the tree and translated to the higher levels by multipole-to-multipole translations.

2. *Downward pass* – coefficients of a local expansion are computed on the highest posssible level by translation of multipole moments, and translated to the lower levels of the tree by local-to-local translations.

Since the multipole coefficients depend on the vector $\mathbf{t}$, these tree traversals have to be repeated in each iteration of an iterative solver. For details see e.g. [Of07].

The overall algorithmic complexity for the approximate assembling and an action of $\mathbf{V}$ is $O(p^2\, n \log n)$. A similar procedure is applied to the case of $\mathbf{K}$, where one additionaly differentiate each y-term in $G_p(x,y)$ subject to the normal direction $n(y)$.

3.2 Adaptive cross approximation

Another method that we describe is the adaptive cross approximation (ACA) [Beb00]. It approximates an admissible block $\mathbf{A} := \mathbf{V}_{\mathcal{C}_x,\mathcal{C}_y} \in \mathbb{R}^{n_x \times n_y}$ by the product of low-rank matrices

$$\mathbf{A} \approx \mathbf{U}_p \, \mathbf{V}_p^T,$$

where $\mathbf{U} \in \mathbb{R}^{n_x \times p}$, $\mathbf{V} \in \mathbb{R}^{n_y \times p}$ with $p\,(n_x + n_y) < n_x\, n_y$ to guarantee a memory reduction. If the latter condition cannot be fulfilled, the block is classified as nonadmissible. The best low-rank matrix approximation in the spectral norm is the truncated singular value decomposition. However, this is impractical as the computational complexity is cubic.

Instead the ACA can be thought as a Lagrange interpolation of the matrix entries subject to properly chosen pivot rows and columns. Let (i_1, j_1), …, (i_p, j_p) be indices of the pivot rows and columns. By $\mathbf{P}_x \in \mathbb{R}^{n_x \times n_x}$ we denote a permutation of the unit matrix, which reorders rows in $\mathbf{A}$ so that they start with $(i_1, \ldots, i_p)$. Similarly $\mathbf{P}_y \in \mathbb{R}^{n_x \times n_x}$ denotes the column permutation of the unit matrix, which shifts the columns $(j_1, \ldots, j_p)$ of $\mathbf{A}$ forward. The ACA then approximates the admissible block $\mathbf{A}$ as follows:

$$\begin{aligned}\mathbf{P}_x \, \mathbf{A} \, \mathbf{P}_y =: \begin{pmatrix} \widetilde{\mathbf{A}}_{11} & \widetilde{\mathbf{A}}_{12} \\ \widetilde{\mathbf{A}}_{21} & \widetilde{\mathbf{A}}_{22} \end{pmatrix} &\approx \begin{pmatrix} \widetilde{\mathbf{A}}_{11} & \widetilde{\mathbf{A}}_{12} \\ \widetilde{\mathbf{A}}_{21} & \widetilde{\mathbf{A}}_{21} \, \widetilde{\mathbf{A}}_{11}^{-1} \, \widetilde{\mathbf{A}}_{12} \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} \widetilde{\mathbf{A}}_{11} \\ \widetilde{\mathbf{A}}_{21} \end{pmatrix}}_{=:\mathbf{U}_p} \underbrace{\left\{ \widetilde{\mathbf{A}}_{11}^{-1} \left(\widetilde{\mathbf{A}}_{11}, \widetilde{\mathbf{A}}_{12} \right) \right\}}_{=:\mathbf{V}_p^T}. \end{aligned} \tag{9}$$

The ACA approximation (9) is constructed adaptively using a partial pivoting in the actual remainder $\mathbf{R}_p := \widetilde{\mathbf{A}} - \mathbf{U}_p \, \mathbf{V}_p^T$, which is the Schur complement with respect to $\widetilde{\mathbf{A}}_{22}$ completed by zero blocks. The method starts to search in the first row $i_1 := 1$ of $\mathbf{R}_0 := \widetilde{\mathbf{A}}$ for the first pivot $(\mathbf{R}_0)_{i_1,j_1}$ being the largest entry in modulus. The first approximation is the cross of the column $\mathbf{U}_1 = \mathbf{u}_1 := (\mathbf{R}_0)_{*,j_1}$ and the row $\mathbf{V}_1 = \mathbf{v}_1 := (1/(\mathbf{R}_0)_{i_1,j_1})\,(\mathbf{R}_0)_{i_1,*}$. Then, we proceed iteratively by searching for further crosses. The second pivoting row i_2 is determined by an entry in the column j_1 of $\mathbf{R}_1$ having the maximal modulus. The pivoting column j_2 is then given by an entry with the maximal modulus in the row i_2 of $\mathbf{R}_1$. The ACA approximation is updated by

the cross of the column $\mathbf{u}_2 := (\mathbf{R}_1)_{*,j_2}$ and the row $\mathbf{v}_2 := (1/(\mathbf{R}_1)_{i_2,j_2})\,(\mathbf{R}_1)_{i_2,*}$ so that $\mathbf{U}_2 := (\mathbf{U}_1, \mathbf{u}_2)$ and $\mathbf{V}_2 := (\mathbf{V}_1, \mathbf{v}_2)$. Notice that ACA does not need $\widetilde{\mathbf{A}}_{22}$. The approximation error is measured in the Frobenius norm. In [BK05] it is shown that, provided $\|\mathbf{R}_k\|_F \leq \eta \|\mathbf{R}_{k-1}\|_F$, the stopping criterion

$$\|\mathbf{u}_p\,\mathbf{v}_p^T\|_F \leq \frac{1-\eta}{1+\varepsilon}\|\mathbf{U}_{p-1}\,\mathbf{V}_{p-1}^T\|_F$$

implies the relative error reduction $\|\mathbf{R}_p\|_F \leq \varepsilon\,\|\widetilde{\mathbf{A}}\|_F$. This makes the cross approximation adaptive. The complexity of the ACA approximation to $\mathbf{V}_{\mathcal{C}_x,\mathcal{C}_y}$ is again $O(p^2\,(n_x+n_y))$.

A straightforward application of ACA to $\mathbf{K}$ may fail, as in some admissible blocks $\mathbf{K}_{\mathcal{C}_x,\mathcal{C}_y}$ there might be zero parts when some pairs of triangles in $\mathcal{C}_x$ and $\mathcal{C}_y$ belong to a common plane. As a remedy, which we were advised by Professor Mario Bebendorf, we apply ACA to another matrix

$$\left(\int_{T_i^x}\int_{T_j^y}\frac{1}{|x-y|^2}\,dS(y)\,dS(x)\right)_{i,j} \in \mathbb{R}^{n_x\times n_y},$$

where T_i^x and T_j^y is the i-th triangle in $\mathcal{C}_x$ and the j-th triangle in $\mathcal{C}_y$, respectively. We remember the resulting pivot indices $(i_1,j_1),\ldots,(i_p,j_p)$. The pivots are then used to calculate the Lagrange interpolation of $1/|x-y|^3$ with the idea of separating x and y in an approximation of the kernel of $\mathbf{K}$ via a product of low-rank functions as follows:

$$\frac{\partial G}{\partial n(y)}(x,y) = \underbrace{\underbrace{(x-y)\,n(y)}_{\text{rank-4 function}}\quad\underbrace{\frac{1}{|x-y|^3}}_{\text{ACA}\,\rightsquigarrow\,\text{rank-}p\text{ function}}}_{\text{rank-}4p\text{ function}}.$$

Since $(x-y)\,n(y)$ is reproduced exactly, the modified ACA preserves the zero parts. The action of $\mathbf{K}_{\mathcal{C}_x,\mathcal{C}_y}$ is performed as follows:

$$\mathbf{K}_{\mathcal{C}_x,\mathcal{C}_y}\cdot\mathbf{g}_{\mathcal{C}_y} \approx \frac{1}{4\pi}\sum_{r=1}^{4}\mathbf{U}_p^r\cdot\left((\mathbf{M}_p)^{-T}\cdot\left((\mathbf{V}_p^r)^T\cdot\mathbf{g}_{\mathcal{C}_y}\right)\right),$$

where the entries of $\mathbf{M}_p \in \mathbb{R}^{p\times p}$, $\mathbf{U}_p^r \in \mathbb{R}^{m_x\times p}$, and $\mathbf{V}_p^r \in \mathbb{R}^{m_y\times p}$ reads

$$(\mathbf{M}_p)_{i,j} := \frac{1}{\left|c(T_{i_k}^x) - c(T_{j_k}^y)\right|^3}, \quad (\mathbf{U}_p^r)_{i,k} := \int_{T_i^x} \frac{f^r(x)}{\left|x - c(T_{j_k}^y)\right|^3}\, dS(x), \quad \text{and}$$
$$(\mathbf{V}_p^r)_{j,k} := \int_{T_j^y} \frac{g^r(y)}{\left|c(T_{i_k}^x) - y\right|^3}\, dS(y),$$

respectively, with

$$\sum_{r=1}^{4} f^r(x)\, g^r(y) := x_1\, n_1(y) + x_2\, n_2(y) + x_3\, n_3(y) + (-1)\, y\, n(y) = (x-y)\, n(y).$$

4 Cyclic decompositions of undirected graphs

We wish to assign $N \times N$ matrix blocks to N processes such that each process is assigned exactly one diagonal block, which we expect to balance the load per process when employing a fast BEM. At the same time, we wish the maximum of the number of block row or column indices per process to be minimized, which minimizes the memory per process. We shall formulate this combinatorial problem in terms of graph theory. For certain values of N an optimal solution using a cyclic decomposition of the complete undirected graph on N vertices into complete subgraphs will be obtained. Decompositions that are not cyclic are rarely to be obtained in an easy and systematic way. Nevertheless, any decomposition of a complete graph into complete subgraphs leads to a perfect distribution of memory blocks among processes.

We recall a few notions. By a simple undirected graph G we understand a pair (V, E), where V is a set of vertices and E is a set of two element subsets of V called edges. The simple undirected graph with each pair of vertices connected by an edge is called a complete graph and denoted by K_N. We set the correspondence between the $N \times N$ block matrix and the complete graph K_N as follows. The N row (and column) indices of the matrix correspond to the N vertices of the complete graph K_N and each edge $\{w, z\} \in E(K_N)$ corresponds to a pair of off-diagonal blocks of the $N \times N$ matrix, first block with indices z, w and second block with indices w, z.

A graph can be decomposed into smaller graphs with respect to edges. By a G-decomposition of a complete graph K_N we understand such a system of pairwise edge disjoint subgraphs $G_0, G_1, \ldots, G_q$, where each G_i is isomorphic to G, that each edge of K_N belongs to exactly one copy G_i of G. A

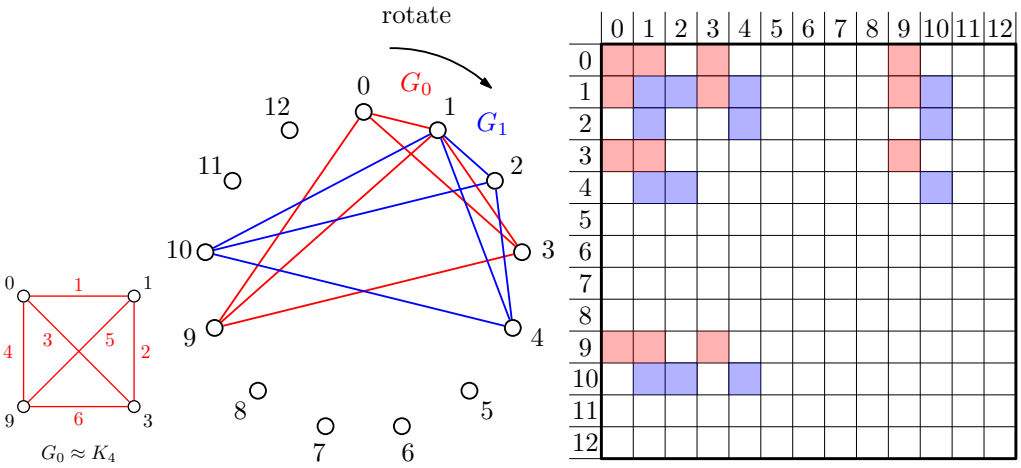

Figure 1: Example of a ρ-labeling of K_4 with induced edge labels (left); a cyclic decomposition of K_{13} into 13 copies of K_4 (middle); a 13×13 block matrix with highlighted blocks corresponding to copies G_0, G_1 of K_4 with ρ-labeling $\{0, 1, 3, 9\}$ (right).

decomposition is cyclic if there exists an ordering $z_1, z_2, \ldots, z_N$ of vertices of K_N and there exist isomorphisms $\phi_i : G_0 \to G_i$, where $i = 0, 1, \ldots, q$, such that $\phi_i(z_j) = z_{j+i}$ for every $j = 1, 2, \ldots, N$, where the indices are taken modulo N. We adopt the usual convention $N = 0$ of modular arithmetic.

Among popular tools used for graph decompositions there are graph labelings. Rosa [Ros67] proved, that there exists a cyclic G-decomposition of K_{2k+1} into $2k + 1$ copies of G with k edges if and only if G has the so called ρ-labeling of $V(G)$. Loosely saying a vertex labeling is a mapping of non-negative integers to the vertices of a graph, while labels of each edge can be induced from the labels of its end vertices, see Fig. 1 (left). A ρ-labeling of a graph G with k edges is such an injective mapping $f : V(G) \to \{0, 1, \ldots, 2k\}$ that the set of edge labels induced by $l(z, w) = \min\{|f(w) - f(z)|, 2k + 1 - |f(w) - f(z)|\}$ is $\{1, 2, \ldots, k\}$. We refer to Fig. 1 (middle) for an example of a cyclic K_4-decomposition of K_{13} based on a ρ-labeling, which is depicted in Fig. 1 (left).

Recall that our objective is to distribute all the $N \times N$ blocks to N parallel processes each with as few indices as possible. In terms of graphs, we employ a cyclic G-decomposition of K_N, which exists only for certain odd N, as follows. Each copy G_i, where $i = 0, 1, \ldots, N - 1$, assigns to the

i-th process one diagonal block and $N-1$ off-diagonal blocks with indices given by the labels of end-vertices of the edges in G_i. To minimize the total memory consumption the optimal graph G shall have $(N-1)/2$ edges and minimum vertices. Obviously, G is again a complete graph K_M with

$$\frac{M(M-1)}{2} = \frac{N-1}{2} \tag{10}$$

edges. Provided K_M has a ρ-labeling, we can construct a cyclic K_M-decomposition of K_N into $N = M^2 - M + 1$ copies of K_M. The difficulty is that a complete graph K_M allows a ρ-labeling only for certain values of M. Unfortunately, a complete classification is not known, see [CD07, Gal11]. E.g. the complete graphs K_7 and K_{11} do not have a ρ-labeling, thus no K_7-decomposition of K_{43} nor a K_{11}-decomposition of K_{111} exist. The blocks highlighted in Fig. 1 (right) correspond to the first two copies G_0 and G_1 of a K_4-decomposition of K_{13} from Fig. 1 (middle).

On the other hand, there are sufficient conditions known based on which a K_M-decomposition of K_N into N copies can be constructed for infinitely many values of N. A perfect difference set introduced by Singer in [Sin37] with M elements corresponds immediately to a ρ-labeling of a complete graph K_M. If we restrict ourselves to the case when $M-1$ is a prime power, we can use the construction of perfect difference sets [Sin37], see Tab. 1, to find a ρ-labeling of K_M and construct a cyclic K_M-decomposition of K_N.

Notice, that each copy of the complete subgraph G_i in the cyclic decomposition corresponds to blocks, that differ in their indices by $1, 2, \ldots, (N-1)/2$, while each of the differences appears precisely once. Therefore, the corresponding submeshes are likely to be distributed throughout the mesh in a sense evenly. This supports a good load balance in average.

5 Parallel implementation, scalability

The idea of the parallel implementation is now straightforward. It relies on ρ-labelings, examples of which are given in Tab. 1. For a feasible N we decompose the mesh τ^h consisting of n triangles into N parts $\tau_0^h, \ldots, \tau_{N-1}^h$ so that the elements in a submesh τ_i^h are geometrically close and the sizes of the submeshes and the numbers of elements do not differ much from n/N. For this purpose we employ the software package Metis [KK99]. Each process with an index $P \in \{0, 1, \ldots, N-1\}$ then translates the set of labels

N	M	set of labels in a ρ-labeling of K_M
3	2	$\{0,1\}$
7	3	$\{0,1,3\}$
13	4	$\{0,1,3,9\}$
21	5	$\{0,1,4,14,16\}$
31	6	$\{0,1,3,8,12,18\}$
57	8	$\{0,1,3,13,32,36,43,52\}$
73	9	$\{0,1,3,7,15,31,36,54,63\}$
91	10	$\{0,1,3,9,27,49,56,61,77,81\}$
133	12	$\{0,1,3,12,20,34,38,81,88,94,104,109\}$
183	14	$\{0,1,3,16,23,28,42,76,82,86,119,137,154,175\}$
273	17	$\{0,1,3,7,15,31,63,90,116,127,136,181,194,204,233,238,255\}$

Table 1: Selected set of labels for K_M-decompositions of K_N, where $M = \frac{1}{2}\left(1+\sqrt{4N-3}\right)$.

$\{i_1,\ldots,i_M\}$ to the index vector of submeshes

$$\mathbf{s}^P := ((i_1+P) \operatorname{rem} N, \ldots, (i_M+P) \operatorname{rem} N),$$

where $a \operatorname{rem} b$ gives the remainder of the division of a by b. This simple expression takes the advantage of the cyclic decomposition.

The setup phase starts so that each process P concurrently uploads the associated submeshes $\mathcal{S}_i^P := \tau^h_{(\mathbf{s}^P)_i}$ for $i = 0,\ldots,M-1$. Next, nodes and edges (pairs of nodes) on boundaries of submeshes are identified and sent to the master $P := 0$. The master process introduces a global sorting of nodal indices. The setup phase is completed by creating trees of boundary elements.

The second phase is the assembling. For $P \in \{0,1,\ldots,N-1\}$ and $i,j \in \{0,1,\ldots,M-1\}$ we denote by $\mathbf{V}^P_{i,j}$, $\mathbf{K}^P_{i,j}$ the blocks of the matrices $\mathbf{V}$, $\mathbf{K}$, respectively, associated to the process P and related to the submeshes $\mathcal{S}_i^P$ and $\mathcal{S}_j^P$. Similarly, we denote by $\mathbf{g}_i^P$ the block of the Dirichlet datum $\mathbf{g}$. Each P assembles the following blocks by means of a fast BEM:

- one diagonal block $\mathbf{V}^P_{0,0}$,
- $M(M-1)$ or $M(M-1)/2$ off-diagonal blocks $\mathbf{V}^P_{i,j}$, for $i \neq j$ or $i < j$, respectively, in case of FMM or ACA,
- one diagonal block $\mathbf{K}^P_{0,0}$,

- $M(M-1)$ off-diagonal blocks $\mathbf{K}^P_{i,j}$, $i \neq j$,
- and one block $\mathbf{g}^P_0$.

After the assembling phase the master process gathers the vector $\mathbf{g}$ and requests for the action $\mathbf{K} \cdot \mathbf{g}$. This is summed up from the following parallel contributions:

$$\sum_{j=0}^{M-1} \mathbf{K}^P_{0,j} \cdot \mathbf{g}^P_j \quad \text{and} \quad \sum_{\substack{j=0 \\ j \neq i}}^{M-1} \mathbf{K}^P_{i,j} \cdot \mathbf{g}^P_j, \ i = 1, \ldots, M-1,$$

and transformed by the master to the right-hand side of (4). Finally, for a solution of (4) the conjugate gradient method is employed so that in each iteration the master asks for the action $\mathbf{V} \cdot \mathbf{r}$. Each process P calculates the contributions

$$\sum_{j=0}^{M-1} \mathbf{V}^P_{0,j} \cdot \mathbf{r}^P_j \quad \text{and} \quad \sum_{\substack{j=0 \\ j \neq i}}^{M-1} \mathbf{V}^P_{i,j} \cdot \mathbf{r}^P_j, \ i = 1, \ldots, M-1,$$

while, in case of the ACA, it makes use of the symmetry of $\mathbf{V}$.

We shall estimate computational time for the setup, assembling, and matrix action as well as memory-per-process consumption. We can derive these for a slightly modified variant of the implementation, which in practice delivers still optimal, though a bit worse time and memory demands. During the assembling phase, identically to the implementation above, each process assembles the diagonal blocks $\mathbf{V}^P_{0,0}$ and $\mathbf{K}^P_{0,0}$. This enjoys the complexity $O((n/N)\log(n/N))$. Now the difference is that rather than assembling the remaining off-diagonal blocks, each process assembles only two larger hierarchical matrices $\mathbf{V}^P$ and $\mathbf{K}^P$ related to the union $\mathcal{S}^P$ of the submeshes $\mathcal{S}^P_0, \ldots, S^P_{M-1}$. The action $\mathbf{V} \cdot \mathbf{r}$ comprises the following contributions $\mathbf{v}^P$ of processes $P \in \{0, 1, \ldots, N-1\}$:

$$\mathbf{v}^P := \mathbf{V}^P \cdot \mathbf{r}^P, \quad \mathbf{v}^P_0 := \mathbf{v}^P_0 - (M-1)\, \mathbf{V}^P_{0,0} \cdot \mathbf{r}^P_0.$$

The latter subtraction is due to that each diagonal block is repeated in M matrices $\mathbf{V}^P$. The action $\mathbf{K} \cdot \mathbf{g}$ proceeds similarly. During the setup phase, we still use the same cyclic decompositions as in the implementation above. The only difference is that we construct different trees. We conclude that

the theoretical complexity of the CPU-time for the setup, assembling the hierarchical matrices, a matrix action as well as the memory-per-process is

$$O\left(\frac{Mn}{N}\log\frac{Mn}{N}+\frac{n}{N}\log\frac{n}{N}\right)=O\left(\frac{n}{\sqrt{N}}\log\frac{n}{\sqrt{N}}\right). \tag{11}$$

6 Numerical results

We shall document the parallel efficiency of our method. In our experiments the theoretical estimate (11) of the memory-per-process consumption will be confirmed, while the same theoretical complexity of the setup, assembling, and matrix action will be surpassed. For a fixed number of elements n we measure the time and memory parallel efficiency, respectively, as a function of the number of processes N as follows:

$$E_{\mathrm{CPU}}(N):=\frac{N'\,\mathrm{CPU}(N')}{N\,\mathrm{CPU}(N)}\,100\% \quad\text{and}\quad E_{\mathrm{Mem}}(N):=\frac{\sqrt{N'}\,\mathrm{Mem}(N')}{\sqrt{N}\,\mathrm{Mem}(N)}\,100\%,$$

where N' is the smallest number of processes for which we can run the programme. Note that the efficiency can exceed 100%, since different N lead to different trees, thus, in slightly different approximations of (4).

The numerical experiments were carried out on the cluster Anselm at VŠB-Technical University of Ostrava, Czech Republic. The cluster consists of 209 compute nodes, each equipped with two eight-core 2.4 GHz Intel Sandy Bridge processors and 64 GB of RAM. Compute nodes are interconnected by InfiniBand network. The theoretical peak performance is 82 Tflop/s.

In Tab. 2 we illustrate that a parallel method based on scheduling the jobs with respect to their costs, the so-called largest process time (LPT), suffers from the setup phase as well as from parallely unscalable memory-per-process consumption. The computational domain was $\Omega:=(0,1)^3$ discretized into $n=3072$, 12288, 49152, and 196608 triangles at the levels $l=0$, 1, 2, and 3, respectively. We employed the ACA method with the following parameters: $\eta:=1.1$, $\varepsilon:=10^{-4}$, and $n_{\min}:=10\,(l+1)$ for the respective levels l. The matrix entries (5) were calculated by the second order semi-analytical quadrature method, see [RS07].

In all of our following experiments we discretize the boundary of the shaft depicted in Fig. 2 into $n=10722$, 42888, 171552, 686208, and 2744832 triangles that we denote by uniform refinement levels $l=0$, 1, 2, 3, and 4, respectively.

	setup time [s]; its parallel efficiency E_{CPU} [%] assembling time [s]; E_{CPU} [%] time [s] for 100 matrix actions; E_{CPU} [%] average memory per process [MB]; E_{Mem} [%]					
l	$N := 1$	3	7	13	21	31
	1;100	0;-	0;-	0;-	1;5	0;-
	8;100	3;89	1;114	0;-	0;-	0;-
0	1;100	0;-	0;-	1;8	0;-	0;-
	194;100	234;48	229;32	228;24	238;18	233;15
	4;100	3;44	2;29	2;15	2;10	2;6
	53;100	18;98	8;95	4;102	2;126	2;85
1	7;100	2;117	1;100	1;54	1;33	0;-
	332;100	304;63	279;45	271;34	273;27	271;22
	39;100	26;50	25;22	22;14	19;10	21;6
	315;100	105;100	45;100	26;93	16;94	11;92
2	38;100	13;97	6;90	3;97	2;90	2;61
	1069;100	633;97	492;82	444;67	428;55	416;46
		448;100	382;50	311;33	291;22	298;15
		520;100	224;99	112;107	76;98	54;93
3		65;100	32;87	17;88	12;77	10;63
		2111;100	1412;98	1170;87	1068;75	1011;65

Table 2: A parallel ACA for $\mathbf{V}$ using the LPT method.

	setup time [s]; its parallel efficiency E_{CPU} [%]						
	assembling time [s]; E_{CPU} [%]						
	time [s] for 100 matrix actions; E_{CPU} [%]						
	time [s] for 100 matrix actions free of communication; E_{CPU} [%]						
	average memory per process [MB]; E_{Mem} [%]						
l	$N := 1$	3	13	21	73	133	273
	1.1;100	0.9;43	1.3;7	1.4;4	2.5;1	4.3;0	5.7;0
	106;100	35;102	7;111	4;119	1;133	1;139	0;119
0	9.9;100	4.0;82	1.2;65	0.8;59	0.3;41	0.4;21	0.4;8
	9.9;100	4.0;82	1.1;69	0.7;70	0.2;77	0.1;78	0.1;59
	503;100	321;91	237;59	235;47	228;26	229;19	230;13
	10.9;100	4.2;87	2.0;42	1.9;28	2.6;6	5.0;2	6.1;1
	551;100	192;96	43;98	26;100	6;132	7;61	2;130
1	50;100	20;81	6;61	4;58	2;41	2;19	1;14
	50;100	20;81	6;63	4;62	1;71	1;35	0;61
	1749;100	724;139	334;145	301;127	260;79	255;59	254;42
	158;100	54;97	12;103	8;98	4;50	5;22	7;9
	2381;100	855;93	204;90	123;92	29;111	15;118	8;111
2	212;100	88;80	27;61	17;58	7;44	5;33	5;16
	212;100	88;80	26;63	16;62	5;64	3;65	1;58
	6861;100	2461;161	764;249	594;252	399;201	366;182	352;118
				95;100	26;103	17;86	13;59
				465;100	123;109	67;109	38;96
3				63;100	25;72	20;51	20;24
				59;100	16;105	10;94	6;74
				1775;100	972;98	830;85	758;65
					371;100	198;103	99;100
					508;100	290;96	148;92
4					101;100	79;70	85;32
					65;100	41;87	27;64
					3365;100	2755;90	2436;71

Table 3: A parallel ACA for $\mathbf{V}$ using the cyclic graph decompositions.

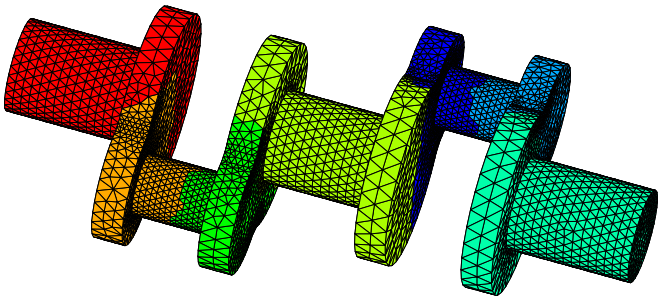

Figure 2: Triangulation of the shaft decomposed into 7 subdomains.

In Tab. 3 we display scalability of the parallel ACA using the cyclic decompositions applied to the matrix $\mathbf{V}$. The parameters and quadrature method remain the same as in Tab. 2. In contrary to the LPT parallelization, we observe that the overall time of the setup and assembling enjoys the parallel scalability $O(1/N)$ as far as the local number of elements n/N is large enough. The parallel complexity of the memory-per-process consumption follows the theoretical estimate $O(1/\sqrt{N})$ as far as it is sufficiently larger than 200 MB, which was always consumed by the system. The parallel scalability is satisfactory up to the scalability of the matrix action, which deteoriates because the communication dominates the computation. While the computational time of the matrix action without the communication is $O(1/N)$, see the fourth lines in cells of the table, this is not the case when including the communication, see third lines, since time for the communication dominates.

Similarly to Tab. 3, in Tab. 4 we present scalability of the parallel FMM using the cyclic decompositions applied to the matrix $\mathbf{V}$. The FMM parameters were chosen similarly with the only difference that the ACA precision ε is replaced by the FMM expansion order $p := 4$. We do not present setup times as they were equal to those in Tab. 3. By construction only the nonadmissible blocks are assembled while assembling the admissible blocks is replaced by the FMM expansions during each matrix action. The FMM assembling times are higher than ACA since we cannot make use of the symmetry. Un-

	assembling time [s]; its parallel efficiency E_{CPU} [%] time [s] for 100 matrix actions without communication; E_{CPU} [%] average memory per process [MB]; E_{Mem} [%]						
l	$N := 1$	3	13	21	73	133	273
0	26;100	11;78	3;60	2;63	2;67	0;67	0;60
	53;100	26;68	10;41	8;31	6;13	5;8	5;4
	335;100	287;67	235;39	236;31	231;17	232;12	235;9
1	226;100	89;85	29;59	19;57	4;74	2;76	1;66
	131;100	67;65	30;34	24;26	18;10	16;6	15;3
	988;100	532;107	315;87	294;73	266;44	263;33	266;23
2	1067;100	418;85	127;65	81;63	23;63	12;66	7;55
	440;100	201;73	98;35	84;25	63;10	59;6	57;3
	3712;100	1603;34	654;157	544;149	424;106	391;82	390;58
3				297;100	88;98	52;91	34;68
				291;100	235;36	221;21	215;10
				1558;100	991;84	916;68	899;48
4					385;100	197;103	99;100
					890;100	855;57	837;28
					3445;100	3057;83	2959;60

Table 4: A parallel FMM for $\mathbf{V}$ using the cyclic graph decompositions.

	time [s] for assembling $\mathbf{V}$ and $\mathbf{K}$; its parallel efficiency E_{CPU} [%] time [s] for the CG-iterations; E_{CPU} [%] **overall time [s]; E_{CPU} [%]** **average memory per process [MB]; E_{Mem} [%]**						
l	$N := 1$	7	13	31	73	133	273
	376;100	53;102	33;88	12;103	5;107	2;116	3;47
	49;100	9;74	5;69	2;69	1;54	1;39	1;14
0	**427;100**	**63;97**	**40;81**	**16;87**	**9;67**	**8;40**	**11;14**
	882;100	**301;111**	**260;94**	**243;65**	**236;44**	**237;32**	**240;22**
	4307;100	745;83	489;68	172;81	70;85	37;87	20;79
	351;100	68;74	43;62	18;63	9;53	6;44	6;23
1	**4671;100**	**816;82**	**535;67**	**192;78**	**82;78**	**48;73**	**32;53**
	4016;100	**693;219**	**465;239**	**341;211**	**293;160**	**283;123**	**283;86**
		3605;100	2224;87	875;93	391;88	257;74	184;50
		370;100	239;83	99;84	48;74	31;62	40;24
2		**3999;100**	**2478;87**	**981;92**	**445;86**	**295;71**	**236;44**
		2454;100	**1389;130**	**780;150**	**539;141**	**475;119**	**458;86**
					7229;100	1811;219	722;359
					1385;100	515;148	213;174
3					**8649;100**	**2353;202**	**957;242**
					2683;100	**1633;122**	**1293;107**

Table 5: A parallel solution to (4) using the ACA for $\mathbf{V}$, the FMM for $\mathbf{K}$, and the cyclic graph decompositions.

fortunately, unlike ACA, the FMM action without communication behaves rather as $O(1/\sqrt{N})$ than $O(1/N)$, which might be caused by the fact that the ACA rank adapts block by block while it is globally fixed to p in FMM.

In Tab. 5 we show the scalability of the parallel fast BEM using the cyclic decompositions to solution of (4) by the conjugate gradient (CG) method. The right-hand side corresponds to the chosen analytical solution $u(x) := 1/|x - x^*|$, where $x^* \notin \overline{\Omega}$. We apply the ACA to the matrix $\mathbf{V}$ and the FMM to assemble the right-hand side using the matrix $\mathbf{K}$. This combination turned to be most efficient. The ACA parameters for the respective levels $l = 0, 1, 2, 3$ were chosen as follows: $\varepsilon := 10^{-8}, 10^{-8}, 10^{-8}, 10^{-10}$, $\eta := 1.1$, $n_{\min} := 10\,(l+1)$. We employed the second order semi-analytical quadrature method. The FMM expansion orders were $p := 5, 6, 6, 7$. The entries of

K were calculated using Sauter-Schwab quadrature method, see [SS97], of orders 3,4,4,4. The CG relative precision was 10^{-6}. The approximation error

$$\text{error} := \sqrt{\frac{(t - t^h, t - t^h)_{L^2(\Gamma)}}{(t, t)_{L^2(\Gamma)}}}. \tag{12}$$

took the values $3.76{\cdot}10^{-2}$, $1.71{\cdot}10^{-2}$, $9.41{\cdot}10^{-3}$, and $5.73{\cdot}10^{-3}$, which was typically achieved in 280, 350, 420, and 500 CG iterations almost independently of N at the respective levels. Note that the time for the CG-solution, which is not optimally scalable, is marginal when compared to the well-scalable time of the assembling phase.

7 Conclusion

We developed a novel method of a parallel distribution of hierarchical matrices arising in fast BEM methods, namely, in the ACA and FMM. Our method relies on a distribution of $N \times N$ matrix blocks among N processes to be assembled concurrently so that to each process exactly one diagonal block is assigned and the amount of block-row or column indices per process is minimal. This problem turns out to be equivalent to a decomposition of the complete graph K_N into complete subgraphs K_M, where N and M are related by (10). We restricted ourselves to cyclic decompositions, the constructions of which are known for an infinite number of N. Under reasonable assumptions we prove that the method enjoys the parallel scalability $O(1/\sqrt{N})$ of the memory-per-process consumption as well as the overall computational time. However, in our numerical experiments up to $n = 2744832$ boundary elements and $N = 273$ computational cores on a real-world geometry the computational time scales with $O(1/N)$, while the memory demands correspond to theory.

Our method partly relies on heuristics. A rigorous analysis of the load balance, which is documented only numerically, is missing. Another open question is the parallel scalability of the matrix action, which in our experiments deteoriates due to the fact that the communication dominates over the short computational phase.

References

[Beb00] Bebendorf, M.: Approximation of boundary element matrices. Numer. Math. **86**, 565–589 (2000)

[BK05] Bebendorf, M. and Kriemann, R.: Fast parallel solution of boundary integral equations and related problems. Comp. Vis. Sci. **8**, 121–135 (2005)

[Beb08] Bebendorf, M.: Hierarchical Matrices. Springer (2008)

[BH86] Burnes, J., Hut, P.: A hierarchical $O(N \log N)$ force calculation algorithm. Nature **324**: 446–449 (1986)

[CD07] Colbourn, C.J. and Dinitz, J.H.: The CRC Handbook of Combinatorial Designs. 2nd ed. Chapman & Hall/CRC (2007)

[Duf82] Duffy, M.G.: Quadrature over a pyramid or cube of integrands with a singularity at a vertex. SIAM J. Numer. Anal. **19**, 1260–1262 (1982)

[EH06] Eppler, K., Harbrecht, H.: Second-order shape optimization using wavelet BEM. Optim. Methods Softw. **21**, 135–153 (2006)

[Gal11] Gallian, J.A.: Graph Labeling. Electron. J. Comb., Dynamic Survey **6** (2013)

[GKS98] Grama, A., Kumar, V., Same, A.: Parallel hierarchical solvers and preconditioners for boundary element methods. SIAM J. Sci. Comput. **20**, 337–358 (1998)

[HN89] Hackbusch, W., Nowak, Z.P.: On the fast matrix multiplication in the boundary element methods by panel clustering. Numer. Math. **54**, 463–491 (1989)

[KK99] Karypis, G., Kumar, V.: A fast and highly quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20**, 359–392 (1999)

[LPZ12] Lukáš, D., Postava, K., Životský, O.: A shape optimization method for nonlinear axisymmetric magnetostatics using a coupling of finite and boundary elements. Math. Comp. **82**, 1721–1731 (2012)

[MT97] McLean, W., Tran, T.: A preconditioning strategy for boundary element Galerkin methods. Numer. Meth. Partial Differential Equations **13**, 283–301 (1997)

[Of07] Of, Gunter: Fast multipole methods and applications. Lecture notes in applied and computational mechanics. **29**, 135–160 (2007)

[OM95] Olstad, B. and Manne, F.: Efficient partitioning of sequences. IEEE Trans. Comp. **44**, 1322—1325 (1995)

[PS92] Petersdorff, T. von, Stephan, E.: Multigrid solvers and preconditioners for first kind integral equations. Numer. Meth. Partial Differential Equations **8**, 443–450 (1992)

[RS07] Rjasanow, S., Steinbach, O.: The Fast Solution of Boundary Integral Equations. Springer (2007)

[Rok85] Rokhlin, V.: Rapid solution of integral equations of classical potential theory. J. Comput. Phys. **60**, 187–207 (1985)

[Ros67] Rosa, A.: On certain valuations of the vertices of a graph. In Theory of Graphs, International Symposium, Rome, July 1966. Gordon and Breach, 349–355 (1967)

[Sag94] Sagan, H.: Space-Filling Curves. Springer (1994)

[SS97] Sauter, S., Schwab, C.: Quadrature for hp-Galerkin BEM in $\mathbb{R}^3$. Numer. Math. **78**, 211–258 (1997)

[SS10] Sauter, S., Schwab, C.: Boundary Element Methods. Springer (2010)

[Sin37] Singer, J.: A theorem in finite projective geometry and some applications to number theory. Trans. AMS **43**, 377–385 (1937)