Graz University of Technology
Institute of Applied Mechanics

Preprint Series

Preprint No 01/2020

Fast Boundary-Domain Integral Method with the $\mathscr{H}^2$-matrix formulation for large scale numerical investigations

Jan Tibaut, Martin Schanz
Institute of Applied Mechanics, Graz University of Technology
Jure Ravnik
Faculty of Mechanical Engineering, University of Maribor

Submitted to: *Engineering Analysis with Boundary Elements*
Latest revision: June 22, 2020

Abstract

The implementation of the Boundary Element Method depends on the partial differential equation type. The type of the partial differential equation determines the form of the fundamental solution. However, we do not know the fundamental solution for every partial differential equation. Thus, the Boundary-Domain Integral Method is employed. The Boundary-Domain Integral Method is based on the Boundary Element Method. Fully populated matrices have to be computed. By transforming the fully populated matrix into smaller matrices with an approximation technique on each matrix, we can overcome the excessive storage requirements and the large number of operations, that are needed for the numerical calculation.

In this study, we focus on the domain integral kernel that is present in the Boundary-Domain Integral method. The computation of the Boundary-Domain integrals uses a substantial amount of computational resources. We solve the modified Helmholtz equation. The kernel was approximated with the $\mathscr{H}^2$-matrix formulation and the ACA algorithm. By observing the impact of the approximation on the integral kernel, we were able to asses the influence of the approximation on the solution of the modified Helmholtz equation.

1 Introduction

Large scale numerical analysis in engineering and science is difficult and time-consuming. However, usually, the cost of numerical investigations is lower than the cost of experiments. Numerical simulation plays an important role in practical engineering computation, such as solid mechanics, fluid mechanics, acoustics, electromagnetism, etc. Practical problems are often governed by non-linear partial differential equations. In computational fluid dynamics, partial differential equations have a diffusion, convection, and source part. For unsteady simulations also a transient part is present. For such problems mostly Finite Volume Methods or the Finite Element Method are used. But, as well the Boundary Element Method (BEM) may have advantages and is used to handle non-linear partial differential equations. Such a formulation requires to solve besides the boundary integrals also a domain integral, why it is often denoted Boundary-Domain Integral Method (BDIM).

The BDIM is based on Green's second identity, where a domain integral kernel is present. Even though the numerical method has higher computational costs as conventional BEM, the method is employed by several authors. Mikhailov and Mohamed [1] applied the BDIM to a Neuman boundary value problem for a scalar elliptic partial differential equation with a variable coefficient. Portillo [2] employed the method on a mixed boundary value problem for the diffusion equation with non-homogeneous right hand side. To reduce the computational costs for evaluating the domain integral several methods have been proposed, which are mostly based on an approximation. The dual reciprocity method transforms the domain integral into a boundary integral operator. Partridge and Brebbia [3] used the dual reciprocity method to solve the Poisson equation. Cheng et al. [4] presented global interpolation functions within the dual reciprocity BEM. They obtained a better convergence behavior for their approximation of the transformation. An extension is the triple reciprocity method. Ochiai [5] implemented the triple reciprocity method to solve the heat equation with heat sources. Guo et al. [6] presented an improved implementation of the triple reciprocity BEM for three-dimensional steady-state heat conduction problems. This formulation has reduced computation time and storage space. However, the approximation of the domain integral is highly complex.

The heat equation contains a partial derivative with respect to time, which either requires a time-domain BE formulation (see, e.g. [7, 8]) or it may be discretised by a difference quotient. The latter mentioned semi-discretisation results in a domain integral similar to the non-linear terms and is called false transient. Malinson and Davis [9] presented this approach for the solution of a coupled elliptic equation. Stella and Guj [10] applied the false transient to solve the lid-driven cavity test case with a finite difference scheme. Behnia et al. [11] implemented the same procedure to simulate three-dimensional natural convection flow. The governing equation of the false transient is the inhomogeneous modified Helmholtz equation, which we focus on. It is also referred to as the Yukawa equation [12]. Hriberšek and Škerget [13] employed the modified Helmholtz equation to compute incompressible fluid flow problems. Cui et al. [14] employed the Radial Integration Boundary Element Method for the solution of transient heat conduction problems with heat sources and variable thermal conduction.

Independent which variant of the above briefly discussed BE formulations is used, the bottleneck of every BEM is the complexity with order $\mathcal{O}(n^2)$. The latter holds in principle for the domain integrals. To overcome this restriction fast methods have been developed. The fast mul-

tipole method (FMM) is may be one of the first (see, e.g., [15, 16]). Within the field of the heat equation the formulation by Messner et al. [17] may be mentioned. As well the wavelet transform [18] can be used to accelerate the BEM. Alternatively, hierarchical matrices ($\mathscr{H}$-matrices) with some data compression techniques can be applied. Hackbusch [19] introduced a recursive hierarchical decomposition for the approximation of an asymptotic smooth function. The recursive hierarchical procedure is known as the $\mathscr{H}$-structure. After the $\mathscr{H}$-structure is formed an approximation method is employed. The most efficient method in storage is the Singular Value Decomposition Method (SVD) [20]. However, the computational cost of the method scales $\mathscr{O}(n^3)$. Adaptive Cross Approximation (ACA) presented by Bebendorf [21] is an approximate alternative to the SVD and reduces the complexity to $\mathscr{O}(n\log n)$ for most elliptic operators. Two versions of the algorithm were presented: the partial pivoting and full pivoting ACA [22], where only the first makes sense to accelerate BEM. A variant called Hybrid Cross Approximation (HCA) has been presented by [23]. For numerical analysis in solid mechanics, Bebendorf and Grzibovski [21] employed the ACA to solve a linear elasticity problem with the Galerkin BEM. Heider and Schanz [24] implemented the ACA to solve elasticity problem based on an extension of the ACA given by Rjasanow and Weggler [25]. Grytsenko and Galybin [26] solved multi-crack large-scale problems with the ACA and the $\mathscr{H}$-matrix. Rjasanow and Weggler [27] employed the ACA with the $\mathscr{H}$-matrix formulation to solve Maxwell problems. Tamayo et al. [28] applied the multi-level ACA for electromagnetic and radiation examples. Campos et al. [29] presented an isogeometric BEM that was accelerated with the ACA for the analysis of potential problems. Recently Rodopulos et al. [30] employed the ACA and BEM to solve the chaotic protection problem on a large scale. Chaotic protection techniques are widely used to avoid corrosion in offshore structures. For the problem class treated here, Tibaut et al. [31] employed the $\mathscr{H}$-matrix and the ACA algorithm to accelerate the BDIM. Ravnik et al. [32] performed stochastic modeling of nanofluid flow using the BDIM. Tibaut and Ravnik [33] accelerated the BDIM with the modified Helmholtz equation and the ACA to solve a heat transfer problem.

An improvement of the hierarchical matrix concept has been presented by Börm [34] and is called $\mathscr{H}^2$-matrix. The $\mathscr{H}^2$-matrix form is based on nested cluster basis functions [35]. In the $\mathscr{H}^2$-matrix, the integral kernel is mostly interpolated in the far field with polynomials. Börm and Hackbusch [36] approximated the integral kernel with Lagrangian polynomials. This approach is used here for the modified Helmholtz equation.

The paper is split into five sections. Firstly, we present the governing equation that we used for the numerical investigation. In the second section, we present the integral equation with its discretisation. Thirdly, we present the $\mathscr{H}^2$-matrix and the approximation algorithm ACA. In section four, we discuss the results and in the last section, we conclude the content of this paper.

2 Governing equations

In the present work, we consider a time dependent inhomogeneous partial differential equation for a scalar field of temperature or mass concentration $u(\vec{r}',t)$

$$\frac{\partial u(\vec{r}',t)}{\partial t} = \mathscr{K}\nabla^2 u(\vec{r}',t) + b(\vec{r}',t), \tag{1}$$

where t presents the time, $\vec{r}'$ the spatial coordinate and $b(\vec{r}',t)$ is a source term. This equation can be found, e.g., in investigations on heat transfer or mass transport in fluids and solids. The time derivative $\frac{\partial u(\vec{r}',t)}{\partial t}$ is discretized with a finite difference scheme, here the backward Euler method. Thus, the term is $\frac{1}{\mathscr{D}}\frac{u(\vec{r}',t)-u_{F-1}(\vec{r}',t)}{\Delta t}$, where $u_{F-1}(\vec{r}',t)$ presents the value of the scalar filed potential at the previous time step and $\frac{1}{\mathscr{D}}$ is a relaxation parameter. This results in the time discrete form of Eq.(1)

$$\frac{1}{\mathscr{D}}\frac{u(\vec{r}',t)-u_{F-1}(\vec{r}',t)}{\Delta t} = \mathscr{K}\nabla^2 u(\vec{r}',t) + b(\vec{r}',t). \tag{2}$$

With the reformulation

$$\left(\mathscr{K}\nabla^2 - \frac{1}{\mathscr{D}\Delta t}\right) u(\vec{r}',t) + \frac{1}{\mathscr{D}\Delta t}u_{F-1}(\vec{r}',t) + b(\vec{r}',t) = 0. \tag{3}$$

and $\mu^2 = \frac{1}{\mathscr{D}\Delta t}$, $b'(\vec{r}',t) = \frac{1}{\mathscr{D}\Delta t}u_{F-1}(\vec{r}',t) + b(\vec{r}',t)$ we obtain the modified Helmholtz equation

$$\left(\mathscr{K}\nabla^2 - \mu^2\right) u(\vec{r}',t) + b'(\vec{r}',t) = 0. \tag{4}$$

In case of large time steps $\Delta t \to \infty$, the parameter $\mu^2 \to 0$ is equal to zero. Thus, the modified Helmholtz equation is transformed into the Poisson equation.

2.1 Integral form of the modified Helmholtz equation

In order to write the integral form of the modified Helmholtz equation, we employ Green's second identity. The fundamental solution of the modified Helmholtz equation is the solution of

$$\left(\nabla^2 - \mu^2\right) u^*(\vec{\xi},\vec{r}') + \delta(\vec{\xi},\vec{r}') = 0, \tag{5}$$

where $\delta(\vec{\xi},\vec{r}')$ is the Dirac delta distribution. Its explicit form is

$$u^*(\vec{\xi},\vec{r}') = \frac{e^{-\mu r}}{4\pi r} \tag{6}$$

with $r = \sqrt{(\xi_x - r'_x)^2 + (\xi_y - r'_y)^2 + (\xi_z - r'_z)^2}$. The normal derivative can be written as

$$q^*(\vec{\xi},\vec{r}') = \vec{n}\cdot\vec{\nabla}u^*(\vec{\xi},\vec{r}') = \frac{\vec{n}\cdot\vec{r}'}{4\pi r^3}(1+\mu r)e^{-\mu r}. \tag{7}$$

In case of large time steps the parameter $\mu^2 \to 0$ and the fundamental solutions in Eqs. (6) and (7) revert into those of the Laplace equation. Properties of the employed fundamental solution were discussed in [33]. Taking into account the presented fundamental solution in Green's second identity we can write the integral form of the modified Helmholtz equation for a domain Ω with boundary $\Gamma = \partial\Omega$

$$\begin{aligned} c(\vec{\xi})u(\vec{\xi}) + \int_\Gamma \left[u(\vec{r}',t)\vec{\nabla}u^*(\vec{\xi},\vec{r}') - u^*(\vec{\xi},\vec{r}')\vec{\nabla}u(\vec{r}',t)\right]\cdot d\vec{\Gamma} = \\ \int_\Omega u^*(\vec{\xi},\vec{r}')\left[\mu^2 u(\vec{r}',t)_{F-1} + b(\vec{r}',t)\right] d\Omega. \end{aligned} \tag{8}$$

In the following, we present the discretization of the modified Helmholtz equation and the implementation of the approximation procedure on the full matrices that evolve from the discretization.

2.2 Discretization of the integral formulation

In this subsection we focus on the discretization of Eq.(8). In order to discretize the equation, the boundary is divided into N boundary elements and the domain Ω is divided into M domain cells. Each boundary element has a surface area and each domain cell has a volume. Thus, the sum of boundary elements is equal to the surface area of the boundary and the sum of the cells is equal to the volume of the domain:

$$\Gamma = \sum_{i}^{N} \Gamma_i, \quad \Omega = \sum_{j}^{M} \Omega_j. \tag{9}$$

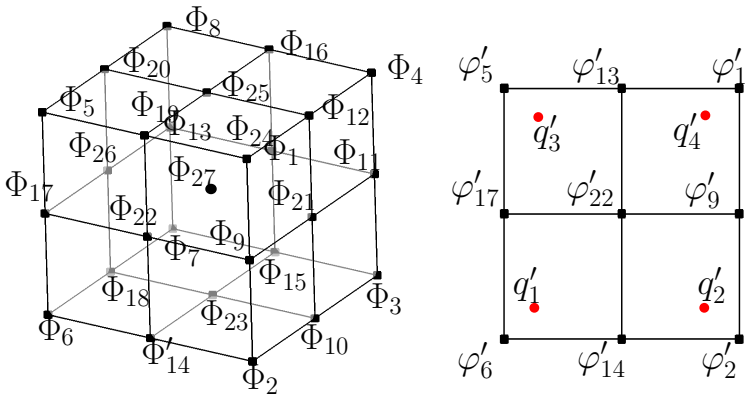

Figure 1: Interpolation functions Φ for the domain cell and φ', q' for the boundary element.

To discretize the integral form of the modified Helmholtz equation, the boundary and domain integral in Eq.(8) were replaced with Eq.(9):

$$\begin{aligned} c(\vec{\xi})u(\vec{\xi}) + \sum_{i}^{N} \int_{\Gamma_i} \left[u(\vec{r}',t)\vec{\nabla}u^*(\vec{\xi},\vec{r}') - u^*(\vec{\xi},\vec{r}')\vec{\nabla}u(\vec{r}',t) \right] \cdot d\vec{\Gamma}_i = \\ \sum_{j}^{M} \int_{\Omega_j} u^*(\vec{\xi},\vec{r}') \left[\mu^2 u(\vec{r}',t)_{F-1} + b(\vec{r}',t) \right] d\Omega_j. \end{aligned} \tag{10}$$

The values of u_b are interpolated on the boundary of the element Γ_i: $u(\vec{r},t) = \sum_b^B u_b \varphi'_b$ and the values of the normal derivative q_b are interpolated inside the element $q(\vec{r},t) = \sum_b^{B'} q_b q'_b$, where φ'_b are quadratic Lagrange interpolation functions and q'_b are linear discontinuous Lagrange interpolation functions on the boundary element Γ_i. The latter choice avoids undefined values of the normal vector on corners.

The values of the previous time step u_{dF-1} are interpolated inside of a cell Ω_j with quadratic Lagrange interpolation functions $u(\vec{r}',t)_{F-1} = \sum_d^D u_{d(F-1)}\Phi_d$ and the values of the source b_d as well $b(\vec{r}',t) = \sum_d^D b_d\Phi_d$. In Fig.1, we present the position of each interpolation function Φ, φ' and q', on the boundary element and the domain cell. Considering the approximation of the fields $u(\vec{r}',t)$, $q(\vec{r}',t)$, $u(\vec{r}',t)_{F-1}$ and $b(\vec{r}',t)$ we obtain the flowing form of Eq.(10):

$$\begin{aligned} c(\vec{\xi}_k)u(\vec{\xi}_k) + u_b\sum_i^N\sum_b^B\int_{\Gamma_i}\varphi'_b q^*(\vec{\xi}_k,\vec{r}'_b)d\Gamma_i = \\ q_b\sum_i^N\sum_b^{B'}\int_{\Gamma_i} q'_b u^*(\vec{\xi}_k,\vec{r}'_b)d\Gamma_i + \left[\mu^2 u_{d(F-1)} + b_d\right]\sum_j^M\sum_d^D\int_{\Omega_j}\Phi_d u^*(\vec{\xi}_k,\vec{r}'_d)d\Omega_j. \end{aligned} \tag{11}$$

The values u_b and q_b are unknown. They are determined by applying the boundary conditions. Using a collocation method we write Eq.(11) for each source point $\vec{\xi}$ and domain point $\vec{r}'$. The system of equations can be written with the matrices

$$h_{kb} = \delta_{kb}c(\vec{\xi}_k) + \int_{\Gamma_i}\varphi'_b q^*(\vec{\xi}_k,\vec{r}'_b)d\Gamma_i, \tag{12}$$

$$g_{kb} = \int_{\Gamma_i} q'_b u^*(\vec{\xi}_k,\vec{r}'_b)d\Gamma_i, \tag{13}$$

$$b_{kd} = \int_{\Omega_j}\Phi_d u^*(\vec{\xi}_k,\vec{r}'_d)d\Omega_j. \tag{14}$$

Elements h_{kb} in Eq.(12) form the matrix $[H]$, elements g_{kb} in Eq.(13) are part of matrix $[G]$ and elements b_{kd} in Eq.(14) form matrix $[B]$. The variables u_b, q_b and $u_{d(F-1)}$ and the values of the source points b_d are formulated into vectors. After assembling the matrices, the modified Helmholtz equation can be presented in this form:

$$[H]\{u\} - [G]\{q\} = [B]\left(\mu^2\left\{u_{(F-1)}\right\} + \{b\}\right). \tag{15}$$

Matrices $[H]$ and $[G]$ are of the size $n \times n$, where n is the number of unknowns on the boundary. Matrix $[B]$ is of the size $n \times m$ where m is the number of nodes in the domain. In case of very large time steps $\mu^2 \to 0$, Eq.(15) transforms into the matrix form of the integral equation

$$[H]\{u\} - [G]\{q\} = [B]\{b\} \tag{16}$$

corresponding to Poisson's equation.

3 $\mathscr{H}$-structure

Let us consider a matrix $[B]$ that is of the form $m \times n$. In order to form a $\mathscr{H}^2$-matrix, we have to build cluster tress with the bottom-up approach. Next, a block cluster tree is formed of the cluster trees and lastly, the admissibility test is performed to determine the matrices $[\hat{B}]_{\hat{m}\times\hat{n}}$.

3.1 The Cluster tree

The computational domain is meshed with hexahedral elements. The smallest parts of the mesh are boundary elements and the domain cells. The cluster trees are formed using the bottom-up approach (bounding box method) [31]. The bottom-up approach combines the neighboring cells and forms new clusters on a new level. The procedure is repeated until one cluster that presents the whole domain is left. We will denote the cluster tree that is built from the boundary elements as T_J and the domain cluster tree that is built from the domain cells as T_I. The cluster trees have a number of levels p and each level is built from a number of clusters c.

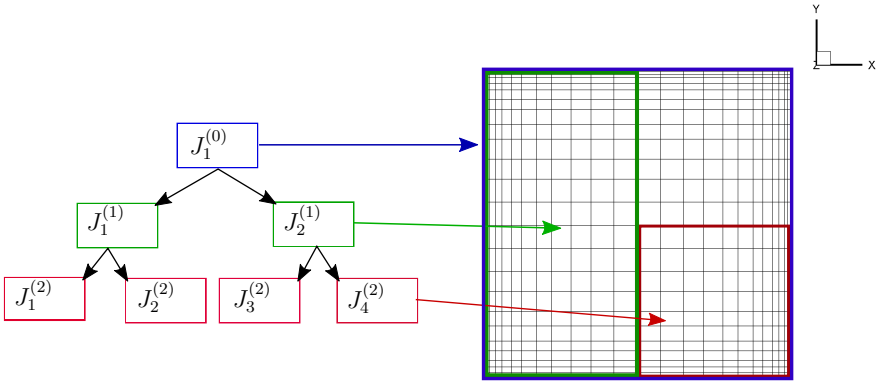

Figure 2: Illustration of a surface on the computational domain and the cluster tree T_J that is formed from clusters J_n^p.

In Fig.2, we illustrate clusters that form a boundary element cluster tree T_J. Moving from bottom to top level the size of clusters increases. The number of clusters in each level decreases by the function $c = 2^{p-1}$. On the top level, $p = 0$ the size of the cluster equals the size of the domain. After the cluster trees are built a block cluster tree is formed.

3.2 Block cluster tree

The block cluster tree $T_{J\times I}$ is a combination of clusters that are in the cluster tree T_I and T_J. To form a block cluster $(I_j^{(i)} \times J_k^{(i)})$, clusters $I_j^{(i)}$ and $J_k^{(i)}$ are combined together and attached to a level of the block cluster tree. In Fig.3, two cluster trees are presented. The cluster I_2^1 in cluster tree T_I is combined with the clusters in T_J, to form block clusters.

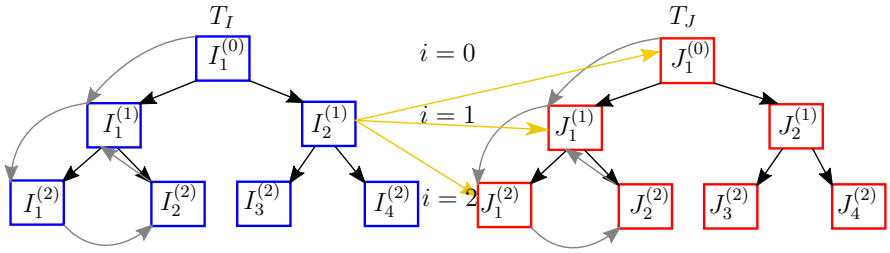

Figure 3: Block cluster tree $T_{J\times I}$ of the clusters I and J. A combination of the clusters $I_j^{(i)}$ and $J_k^{(i)}$ forms a block cluster $I_j^{(i)} \times J_k^{(i)}$. The formed block cluster is tested for admissibility.

After the block clusters are formed, each one is tested for admissibility. The admissibility condition is of this form:

$$min\{dim(I_i^{(i)}), dim(J_k^{(i)})\} \leq \eta \; dist(I_j^{(i)}, J_k^{(i)}), \tag{17}$$

where $dim(I_i^{(i)})$ and $dim(J_k^{(i)})$ are the cluster sizes and $dist(I_j^{(i)}, J_k^{(i)})$ is the minimal distance between the clusters. The parameter η is defined by the user. The condition in Eq.(17) denotes the admissibility or inadmissibility of the block cluster. If the block cluster is inadmissible then it is split into smaller block clusters. Block clusters that are at the lowest level of the block cluster tree and do not fulfill the admissibility condition are considered inadmissible. Lastly, the tested block clusters are assembled into an $\mathscr{H}^2$-matrix form of $[\hat{B}]$, which is of the size $\hat{m} \times \hat{n}$. All matrices $[\hat{B}]_{\hat{m}\times\hat{n}}$ in a group, present the full matrix $[B]_{m\times n}$. However, some matrices $[\hat{B}]_{\hat{m}\times\hat{n}}$ have elements with very small values. Increasing μ in the modified Helmholtz fundamental solution, the shape of the fundamental solution changes to a more local shape. Thus, the values of elements in some matrices are getting smaller. In order to reduce the number of matrices and to reduce the computational cost of the method an additional condition was employed to eliminate matrices that have elements with small values:

$$||\hat{B}_{\hat{m}\times\hat{n}}|| \leq 10^{-15}, \tag{18}$$

where $||\hat{B}_{\hat{m}\times\hat{n}}||$ is the Frobenius norm. If the norm of the matrix $[\hat{B}]_{\hat{m}\times\hat{n}}$ is smaller or equal to the prescribed minimal norm value, the matrix is deallocated from the memory.

For admissible matrices, $[\hat{B}]_{\hat{m}\times\hat{n}}$ that have a Frobenius norm higher then the prescribed value in Eq.(18), an approximation of the integral kernel is employed. Even though the approximation of the integral kernel can be implemented on all elements of the admissible matrices, the expected accuracy of the approximation would be poor. Thus, another admissibility condition is employed to obtain a better approximation accuracy of the integral kernel:

$$dist(I_j^{(i)}, J_k^{(i)}) \geq dist_m, \tag{19}$$

where $dist_m$ is the minimal distance between the clusters. The minimal distance is the absolute value of the minimal distance between the clusters $dist(I_j^{(i)}, J_k^{(i)})$ in the admissible condition Eq.(17). For this study we set a constant value $dist_m = 0.25$.

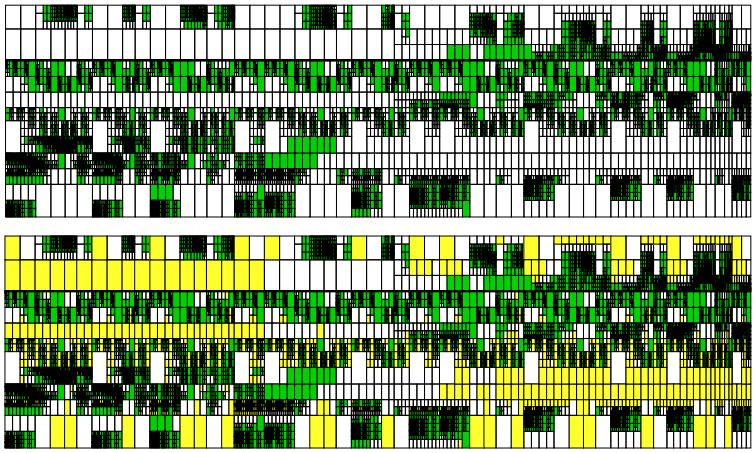

Figure 4: Two $\mathscr{H}^2$-matrices $[B]_{m\times n}$. Three parameters were used $\eta = 5$, then $\mu = 20$ (top) and $\mu = 50$ (bottom) for the determination of the fundamental solution Eq(6) and $dist_m = 0.25$.

In Fig.4, we present two $\mathscr{H}^2$-matrices. For the top matrix the value of the parameter $\mu = 20$ and bottom matrix the value was $\mu = 50$. The matrices that form the two $\mathscr{H}^2$-matrices have different colors. Inadmissible matrices are black. In order to obtain a better approximation accuracy, we divide the admissible matrices into two groups: green matrices where the integral kernel is not approximated and white matrices where integral kernel is approximated. Lastly we present yellow matrices that have a low Frobenius norm. The amount of Yellow matrices increases with a bigger μ. Thus, the memory space that is needed to store the matrices can be deallocated.

3.3 Approximation of the integral kernel

Let us consider the integral kernel formulation that was presented in Eq.(12), Eq.(13) and Eq.(14). In order to approximate the integral kernel function we have to approximate the fundamental solution $u^*\left(\vec{\xi},\vec{r}'\right)$ with the Lagrange interpolation function [36]:

$$u^*\left(\vec{\xi},\vec{r}'\right) \approx \sum_{\iota}^{\alpha}\sum_{\kappa}^{\beta} u^*\left(\vec{\xi}_\iota,\vec{r}'_\kappa\right)\mathscr{L}_\iota\left(\vec{\xi}\right)\mathscr{L}_\kappa\left(\vec{r}'\right), \tag{20}$$

where α is the number of interpolation points on the boundary, β is the number of interpolation points in the domain, $\mathscr{L}_\iota(\vec{\xi})$ Lagrange interpolation function for the boundary and $\mathscr{L}_\kappa(\vec{r})$ Lagrange interpolation function for the domain. $\mathscr{L}_\iota(\vec{\xi})$ is defined like this:

$$\mathscr{L}_\iota(\vec{\xi}) = \prod_{e=1}^{3}\prod_{k=0,\,\iota\neq k}^{\alpha}\frac{\xi^e-\xi^e_k}{\xi^e_\iota-\xi^e_k}, \tag{21}$$

where ξ^e are the interpolation points in the selected boundary interval $\hat{m}$. For the domain interpolation interval $\hat{n}$ the interpolation function $\mathscr{L}_\kappa(\vec{r})$ is of the same form.

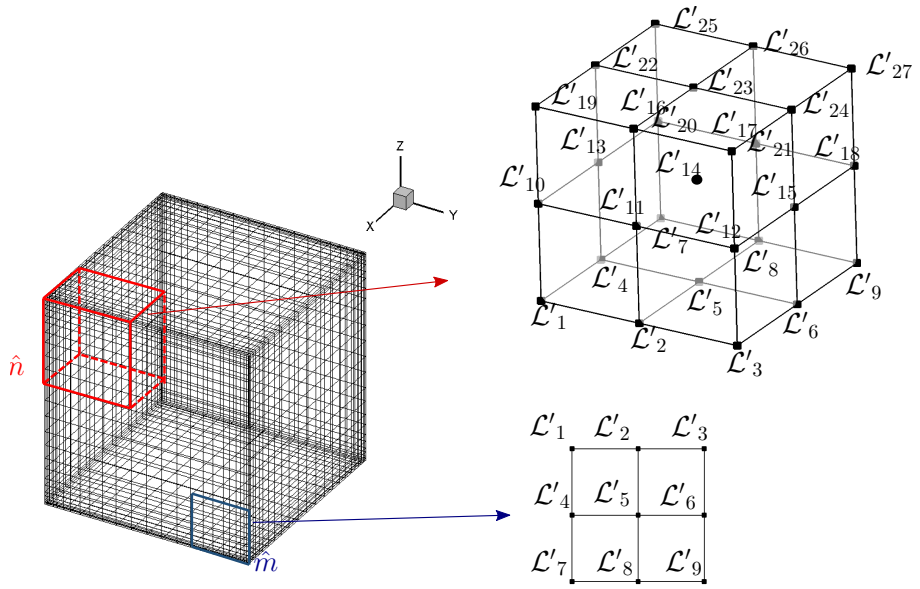

Figure 5: Illustration of the selected cell $\hat{m}$ and boundary $\hat{n}$ interval. The approximation of $\mathscr{L}$ and $\vec{\nabla}\mathscr{L}$ on each cell and boundary element that is in the selected interpolation interval $\hat{m}\times\hat{n}$. The the number of interpolation points was $\gamma = 3$.

In Fig.5 we illustrate the interpolation intervals that are denoted as $\hat{m}$ and $\hat{n}$. In Eq.(12) the normal derivative $q^* = \vec{n}\cdot\vec{\nabla}u^*(\vec{\xi},\vec{r}')$ of the fundamental solution $u^*(\vec{\xi},\vec{r}')$ is present. The fundamental solution was approximated with the Lagrange interpolation function Eq.(20). Thus, the normal derivative of the fundamental solution $q^*(\vec{\xi},\vec{r}')$ is approximated with the derivation of Eq. (20):

$$\vec{\nabla}u^*\left(\vec{\xi},\vec{r}'\right) \approx \sum_{\iota}^{\alpha}\sum_{\kappa}^{\beta} u^*\left(\vec{\xi}_\iota,\vec{r}'_\kappa\right)\mathscr{L}_\iota\left(\vec{\xi}\right)\vec{\nabla}\mathscr{L}_\kappa\left(\vec{r}'\right). \tag{22}$$

The approximation of the fundamental solution Eq.(20) is inserted into Eq.(13) and Eq.(14). The normal derivative of the fundamental solution Eq.(21) is inserted into Eq.(12). Thus, the integral kernel is defined with the Lagrange interpolation functions:

$$h_{kb} = \delta_{kb}c(\vec{\xi}_k) + \sum_{\iota}^{\alpha}\sum_{\kappa}^{\beta} u^*\left(\vec{\xi}_\iota,\vec{r}'_\kappa\right)\mathscr{L}_\iota\left(\vec{\xi}\right)\vec{n}\cdot\int_{\Gamma_i}\varphi'_b\vec{\nabla}\mathscr{L}_\kappa\left(\vec{r}'\right)d\Gamma_i, \tag{23}$$

$$g_{kb} = \sum_{\iota}^{\alpha}\sum_{\kappa}^{\beta} u^*\left(\vec{\xi}_\iota,\vec{r}'_\kappa\right)\mathscr{L}_\iota\left(\vec{\xi}\right)\int_{\Gamma_i} q'_b\mathscr{L}_\kappa\left(\vec{r}'\right)d\Gamma_i, \tag{24}$$

$$b_{kd} = \sum_{\iota}^{\alpha}\sum_{\kappa}^{\beta} u^*\left(\vec{\xi}_\iota,\vec{r}'_\kappa\right)\mathscr{L}_\iota\left(\vec{\xi}\right)\int_{\Omega_j}\Phi_d\mathscr{L}_\kappa\left(\vec{r}'\right)d\Omega_j. \tag{25}$$

To decrease the complexity further the nested cluster bases are employed. Nested cluster bases interpolates the Lagrange interpolation functions $\mathscr{L}_\kappa$ and $\vec{\nabla}\mathscr{L}_\kappa$ on each boundary element Γ_i and domain cell Ω_j [35]:

$$\mathscr{L}_\kappa\left(\vec{r}'\right) = \sum_{\lambda'}^{\gamma} \mathscr{L}'_{\lambda'}\left(\vec{r}'\right) \mathscr{L}''_{\lambda'}, \tag{26}$$

$$\vec{\nabla}\mathscr{L}_\kappa\left(\vec{r}'\right) = \sum_{\lambda'}^{\gamma} \vec{\nabla}\mathscr{L}'_{\lambda'}\left(\vec{r}'\right) \mathscr{L}''_{\lambda'}. \tag{27}$$

where $\mathscr{L}'_{\lambda'}$ and $\vec{\nabla}\mathscr{L}'_{\lambda'}$ are the coefficients of the Lagrange interpolation function. Because we use the collocation scheme we only approximate $\mathscr{L}_\iota$ [37]. In Fig. 5 we illustrate the position of the coefficients $\mathscr{L}'_{\lambda'}$ for a selected polynomial order γ. To determine the coefficients the values of interpolation points that define the boundary element and domain cell are inserted into Eq. (21). $\mathscr{L}''_{\lambda'}$ is the interpolation function for each local point:

$$\mathscr{L}''_{\lambda'} = \prod_{e}^{3} \prod_{k=0,\, k\neq\lambda'}^{\gamma} \frac{\varphi'^e - \varphi'^e_k}{\varphi'^e_{\lambda'} - \varphi'^e_k}, \tag{28}$$

φ'_e is the local coordinate of the interpolation point. Eq.(27) is inserted into Eq.(23), Eq.(26) is inserted into Eq. (24) and Eq.(26) is inserted into Eq.(25). Thus, we approximated the integral kernel function:

$$h_{kb} = \delta_{kb} c(\vec{\xi}_k) + \sum_{\iota}^{\alpha}\sum_{\kappa}^{\beta} u^*\left(\vec{\xi}_\iota, \vec{r}'_\kappa\right) \mathscr{L}_\iota\left(\vec{\xi}_k\right) \sum_{\lambda'}^{\gamma} \vec{n}\cdot\vec{\nabla}\mathscr{L}'_{\lambda'}\left(\vec{r}'_b\right) \int_{\Gamma_i} \varphi'_b \mathscr{L}''_{\lambda'} d\Gamma_i. \tag{29}$$

$$g_{kb} = \sum_{\iota}^{\alpha}\sum_{\kappa}^{\beta} u^*\left(\vec{\xi}_\iota, \vec{r}'_\kappa\right) \mathscr{L}_\iota\left(\vec{\xi}_k\right) \sum_{\lambda'}^{\gamma} \mathscr{L}'_{\lambda'}\left(\vec{r}'_b\right) \int_{\Gamma_i} q'_b \mathscr{L}''_{\lambda'} d\Gamma_i, \tag{30}$$

$$b_{kd} = \sum_{\iota}^{\alpha}\sum_{\kappa}^{\beta} u^*\left(\vec{\xi}_\iota, \vec{r}'_\kappa\right) \mathscr{L}_\iota\left(\vec{\xi}_k\right) \sum_{\lambda'}^{\gamma} \mathscr{L}'_{\lambda'}\left(\vec{r}'_d\right) \int_{\Omega_j} \Phi_d \mathscr{L}''_{\lambda'} d\Omega_j. \tag{31}$$

The elements h_{kb} are assembled into three matrices:

$$[U]^{\hat{n}\times\alpha} = \mathscr{L}_\iota\left(\vec{\xi}_k\right), \tag{32}$$

$$[S]^{\alpha\times\beta} = u^*\left(\vec{\xi}_\iota, \vec{r}'_\kappa\right), \tag{33}$$

$$[V]^{\beta\times\hat{n}} = \sum_{\lambda'}^{\gamma} \vec{n}\cdot\vec{\nabla}\mathscr{L}'_{\lambda'}\left(\vec{r}'_b\right) \int_{\Gamma_i} \varphi'_b \mathscr{L}''_{\lambda'} d\Gamma_i. \tag{34}$$

Together the matrices form the matrix $[H] = [U]^{\hat{n}\times\alpha}[S]^{\alpha\times\beta}[V]^{\beta\times\hat{n}}$. Next the matrices $[U]^{\hat{n}\times\alpha}$, $[S]^{\alpha\times\beta}$ and $[W]^{\beta\times\hat{n}}$ are assembled from Eq. (30). Matrix $[W]^{\beta\times\hat{n}}$ is of this form:

$$[W]^{\beta\times\hat{n}} = \sum_{\lambda'}^{\gamma} \mathscr{L}'_{\lambda'}\left(\vec{r}'_b\right) \int_{\Gamma_i} q'_b \mathscr{L}''_{\lambda'} d\Gamma_i. \tag{35}$$

The three matrices form $[G] = [U]^{\hat{n}\times\alpha}[S]^{\alpha\times\beta}[W]^{\beta\times\hat{n}}$. Lastly Eq.(31) is split into matrices $[U]^{\hat{n}\times\alpha}$, $[S]^{\alpha\times\beta}$ and $[V^D]^{\beta\times\hat{n}}$, where the matrix $[V^D]^{\beta\times\hat{n}}$ is of the form:

$$[V^D]^{\beta\times\hat{m}} = \sum_{\lambda'}^{\gamma} \mathscr{L}'_{\lambda'}\left(\vec{r}'_d\right) \int_{\Omega_j} \Phi_d \mathscr{L}''_{\lambda'} d\Omega_j. \tag{36}$$

The matrices are assembled into matrix $[B] = [U]^{\hat{n}\times\alpha}[S]^{\alpha\times\beta}[V^D]^{\beta\times\hat{m}}$. The approximation of the integral kernel for the collocation numerical scheme was presented. The accuracy of the method depends on the number of interpolation points α, β and the degree of the interpolation polynomial $\mathscr{L}''_{\lambda'}$ [22]. The approximation of the integral kernel can not be employed for every value of the fundamental solution.

3.4 The ACA algorithm

After constructing the $\mathscr{H}^2$-matrix and the approximation of the integral kernel, the admissible matrices are approximated with the ACA algorithm. Let us denote $[B]_{n\times m}$ as a matrix with entries $[B]_{n\times m} \in \mathbb{R}_{n\times m}$ (n is the number of nodes at the boundary and m is the number of nodes in the domain). The matrix $[B]_{n\times m}$ was reformed into a $\mathscr{H}^2$-matrix. Thus, the matrix $[B]_{n\times m}$ is separated into smaller matrices $[\hat{B}]_{\hat{n}\times\hat{m}}$. The matrices that meet the admissibility condition are reformed into $[\hat{B}] = [U]^{\hat{n}\times\alpha}[S]^{\alpha\times\beta}[V^D]^{\beta\times\hat{m}}$. ACA is a method that transforms the matrix $[S]^{\alpha\times\beta} = [A]^{\alpha\times r}[B]^{r\times\beta}$, into a low rank matrix form [19], where r is the approximation rank of the matrix. Thus, a reduction of the memory storing space and numerical complexity is reached.

The approximation algorithm is written below:

- State $R^0 = [S]^{\alpha\times\beta}$
- For $r = 0, 1, 2, 3, \ldots, k$
  1. $(i^*, j^*)^r = ArgMax|(R^r)|$
  2. $\tau^{r+1} = (R^r_{i^*,j^*})^{-1}$
  3. $\vec{a}_{r+1} = \tau^{r+1} R^r_{i,j^*}$, $\vec{b}_{r+1} = (R^r_{i^*,j})^T$
  4. $R^{r+1} = R^r - \vec{a}_{r+1}\vec{b}_{r+1}$
- If $(\left\|\vec{a}^T_{r+1}\right\|_F \left\|\vec{b}^T_{r+1}\right\|_F \leq \varepsilon \left\|S^{r+1}\right\|_F \vee r = k)$ Stop
- EndFor

The ACA does six steps to approximate the matrix. In the first step, the residual matrix R^0 is set. Secondly, the maximal element in the matrix is determined. Thirdly the value of τ for the maximal element is calculated. After that, the vectors $\vec{a}_{r+1}$ and $\vec{b}_{r+1}$ are allocated around the maximal element. Fifth the residual matrix R^{r+1} that is one rank higher then the starting residual matrix is built. In the last step, the Frobenious norm is calculated. The steps from two to six are repeated until the stopping condition is satisfied or the maximal rank of matrix $[S]^{\alpha\times\beta}$ is

reached. The value ε is determined by the user. $\left\|S^{r+1}\right\|_F$ is the Frobenious norm of the matrix S^{r+1} [38]:

$$\left\|S^{r+1}\right\|_F = \|S^r\|_F + \sum_{j=1}^{r} \vec{a}_{r+1}^T \vec{a}_r \vec{b}_r \vec{b}_{r+1}^T + \left\|\vec{a}_{r+1}^T\right\|_F \left\|\vec{b}_{r+1}^T\right\|_F. \tag{37}$$

4 Numerical test

In this section, we present the numerical tests that were performed to validate the developed algorithm. To integrate the kernel we used the Gaussian quadrature. Two kernel forms were employed: in Eq.(14), the exact kernel was integrated and in Eq.(25) the approximated kernel was integrated. Two tests were carried out. For the first test, we validated the approximation algorithm. We replaced the coefficients of the evaluated fundamental solution $u^*\left(\vec{\xi}_\iota, \vec{r}'_\kappa\right)$ in Eq.(20) with the coefficients of this polynomial:

$$f(x_\kappa) = x_\kappa^z, \tag{38}$$

where z determines the degree of polynomial and x_κ is the x-coordinate. The integral for the first case is:

$$\int f(x)\Phi d\Omega. \tag{39}$$

For the second test case, we approximated the integral kernel that is defined with the Helmholtz fundamental solution. The original algorithm integrates the integral kernel Eq.(14) with the Gauss quadrature, while the accelerated method approximates the integral kernel. We observed the accuracy of the approximated integral kernel in matrix $[\hat{B}]_{\hat{n}\times\hat{m}}$ with the relative root mean square error as:

$$RMS_{[B]} = \left(\frac{\sum_i^p \sum_k^{\hat{m}} \sum_b^{\hat{n}} (b_{kd} - \tilde{b}_{kd})_i^2}{\sum_i^p \sum_k^{\hat{m}} \sum_b^{\hat{n}} (b_{kd})_i^2}\right)^{\frac{1}{2}}, \tag{40}$$

where p is the number of matrices, b_{kd} are the elements of the matrix $[\hat{B}]$, where exact integral was solved and $\tilde{b}_{kd}$ are the elements of the same matrix, where the approximation of the integral was employed.

The computational domain was a unit cube with a structured nonuniform mesh. The domain cells are formed from hexahedral elements. To validate the algorithm and examine the method, we increased the number of interpolation points α, β and γ, from 1 to 10. Secondly, the ACA algorithm was employed to observe the impact of the approximation on the approximated integral kernel. The stopping condition of ACA was set from 10^{-12} to 10^{-4}. The mesh density of the computational domain was set from 9^3 to 46^3 nodes.

4.1 Results

In this section, we examine the accuracy of the approximated integral kernel. In the first test, we changed the degree of the polynomial in Eq.(38) to observe the $RMS_{[B]}$ of the integral Eq.(39). Secondly, we performed the approximation of the integral kernel.

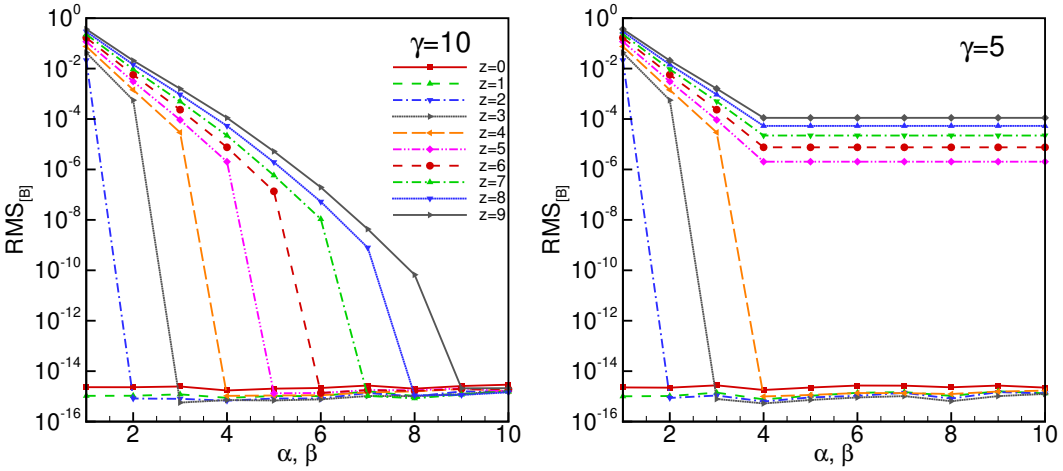

Figure 6: The approximation of the integral kernel in Eq.(38). The norm $RMS_{[B]}$ depending on the polynomial degree and the approximation accuracy of the integral depending on the number of interpolation points γ. The mesh density was 9^3 nodes.

In Fig.6, we present the results of the first test. The results show that a larger interpolation order increases the accuracy of the approximated integral. In the case that the number of the interpolation points equals the degree of the polynomial, the solution of the approximated integral is equal to machine precision. The accuracy depends on the number of interpolation points and the degree of the polynomial. However, if the number of interpolation points γ is constant, the solution of the approximated integral is no longer exact, when the degree of the polynomial is larger then γ. This is presented in Fig.6 (right panel), where the number of interpolation points $\gamma = 5$. When the degree of the polynomial is larger than γ, the approximation of the integral no longer converged to the exact solution. Because the approximation accuracy of the Lagrange interpolation functions $\mathscr{L}_\kappa$ depends of γ. The $\mathscr{L}_\kappa$ is approximated on the domain cell Eq.(26). In order to increase the approximation accuracy of the integral a larger number of interpolation points γ has to be employed.

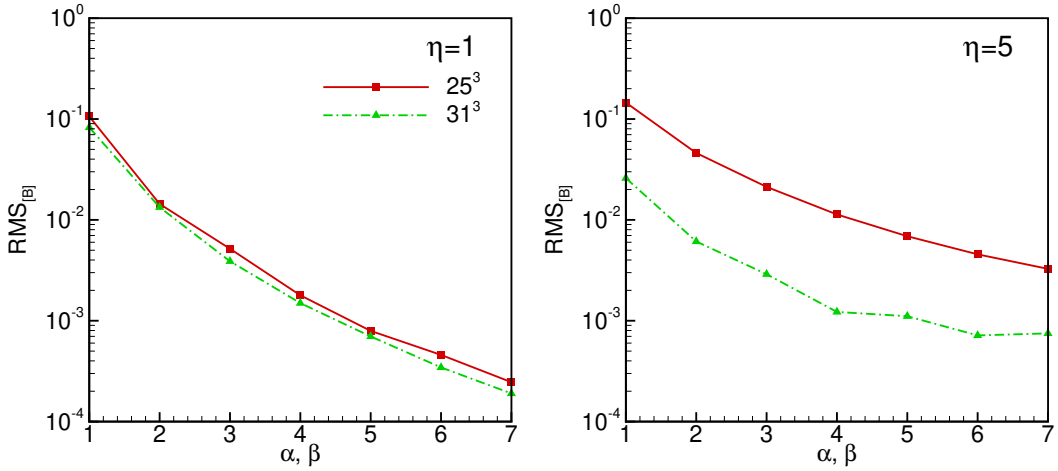

Figure 7: The approximation accuracy of the integral kernel, for the modified Helmholtz solution, when the time step is very large $\mu \to 0$. Two different $\mathscr{H}$-structures, where parameter $\eta = 1$ (left) and $\eta = 5$ (right), for the mesh density 25^3 and 31^3. The number of interpolation points α, β, and γ were set from 1 to 7.

In Fig.7, we present the accuracy of the approximated integral kernel, for the modified Helmholtz fundamental solution $u^*(\vec{\xi},\vec{r}')$, when $\mu \to 0$. We observed the impact of the $\mathscr{H}$-structure at an increasing interpolation order α β and γ, on the approximation of the integral kernel. Results show that the $RMS_{[B]}$ decreases with a larger interpolation order. The $\mathscr{H}$-structure is dependent on the parameter η. The parameter determines the size of approximation. A bigger η increases the approximation of the matrix. The accuracy of the approximated integral kernel depends on the parameter η. Comparing the left and right panel we observe that increasing the parameter η lowers the accuracy of the approximated integral kernel. Hence, a larger order of interpolation is needed to reach the same approximation accuracy when the parameter η is low.

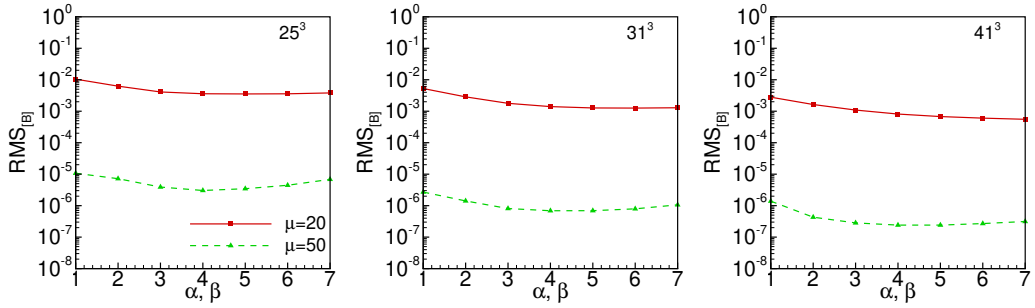

Figure 8: The $RMS_{[B]}$ for two values of parameter μ. Number of interpolation points for the interval $\gamma = 3$. The mesh densities were 25^3 (left), 31^3 (middle) and 41^3 (right).

In Fig.8, we present the $RMS_{[B]}$ depending on the mesh density and the shape of the fundamental solution. The character of the fundamental solution influence the approximation accuracy. A bigger value of μ changes the global character of the modified Helmholtz solution to local.

Thus, a better approximation accuracy can be reached. However, the interpolation order of γ is constant and the accuracy of the approximated kernel does not increase with a larger number of interpolation points α and β. If we increase the number of interpolation points γ, the computational cost of the presented method increases.

In Fig.9, we present the accuracy of the approximated integral kernel depending on the mesh density. For this test the number of interpolation points γ was constant and α, β were increased from 1 to 7. Four different mesh densities were employed. We observed that the accuracy of the approximated integral kernel is depended on the mesh density. The $RMS_{[B]}$ is higher for the mesh density 46^3 compared to 41^3. The mesh density has an influence on the construction of the $\mathscr{H}$-structure. A better-resolved mesh increases the number and size of block clusters. Thus, a larger number of elements in the matrix are approximated. The accuracy of the integral kernel approximation is depended on the interpolation order. A larger interpolation order would be needed to increase the accuracy of the approximated integral kernel on a denser mesh.

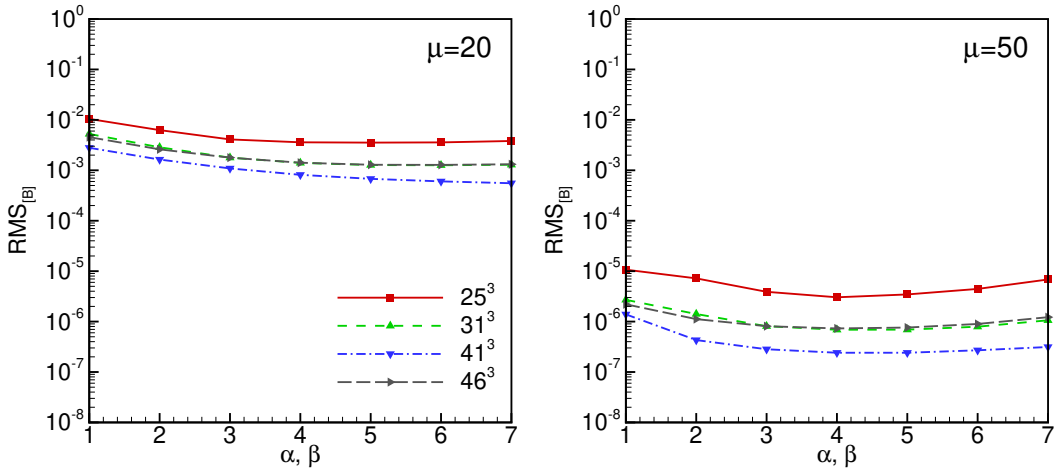

Figure 9: The norm $RMS_{[B]}$ depending on the mesh density and the value of parameter μ. The number of interpolation points for the interval of $\gamma = 3$.

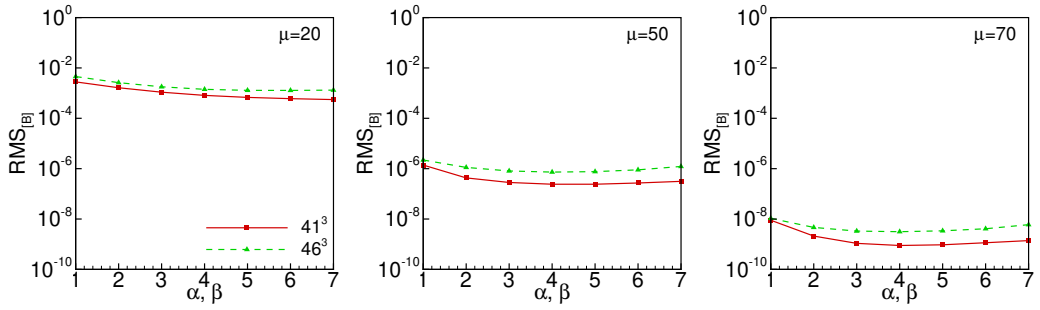

Figure 10: The norm $RMS_{[B]}$ for the mesh density 41^3 and 46^3. Three cases were considered $\mu = 20$ (left), $\mu = 50$ (right) and $\mu = 70$ (middle). Number of interpolation points for the interval $\gamma = 3$. The $\mathscr{H}$-structure was formed with $\eta = 5$.

In Fig.10, we present the $RMS_{[B]}$ norm, which depends on two mesh densities and the parameter μ. The parameter μ was increased from 20 to 70. The modified Helmholtz fundamental solution has a more local character and the approximation of the integral kernel reaches single-precision accuracy. The number of interpolation points γ was constant. Even though the local character of the modified Helmholtz fundamental solution contributes to a better approximation of the integral kernel, one has to account that the values of elements in the matrices decrease with a more local character of the fundamental solution. Hence, fewer elements are approximated. Matrices that have a lower Frobenius norm, then the prescribed condition in Eq.(18) are deallocated from the memory. Furthermore, increasing the parameter μ means that only the main diagonal matrices that are not admissible will stay in the $\mathscr{H}$-structure formulation.

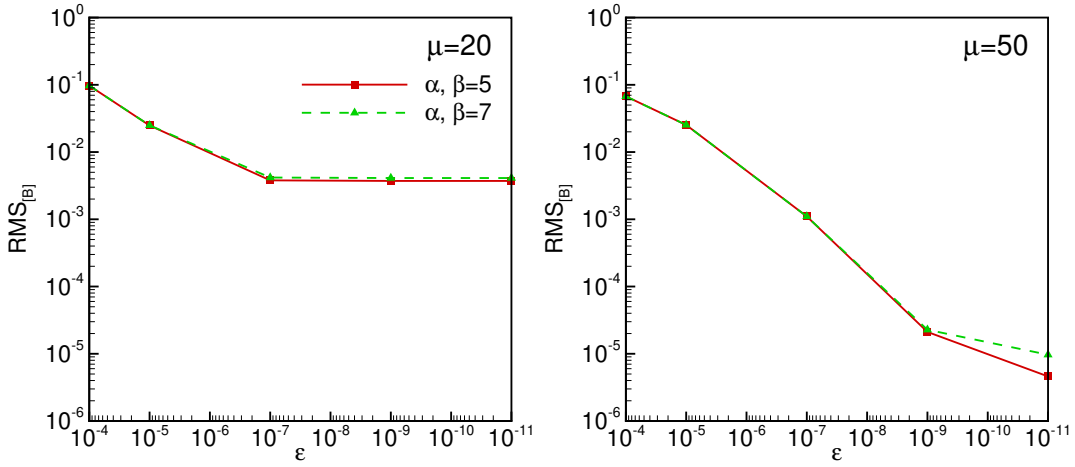

Figure 11: The $RMS_{[B]}$ dependent of the ACA stopping condition ε for $\mu = 20$ (left) and $\mu = 50$ (right). The approximation interval for the integral kernel $\gamma = 3$. The parameter $\eta = 5$ and mesh density 46^3.

The ACA algorithm was employed on admissible matrices to further approximate. In Fig.11 we present the norm $RMS_{[B]}$ to measure the influence of the ACA on the approximation of the integral kernel. The ACA algorithm was employed on matrix $[S]^{\alpha\times\beta}$, that is present in Eq.(31). The rank of matrix $[S]^{\alpha\times\beta}$ is determined with the number of interpolation points α and β. The ACA stopping condition is determined by the user-defined parameter ε (subsection 3.4). The value of the parameter ε was decreased from 10^{-4} to 10^{-11}. Results show that the approximation accuracy increases with a smaller parameter ε.

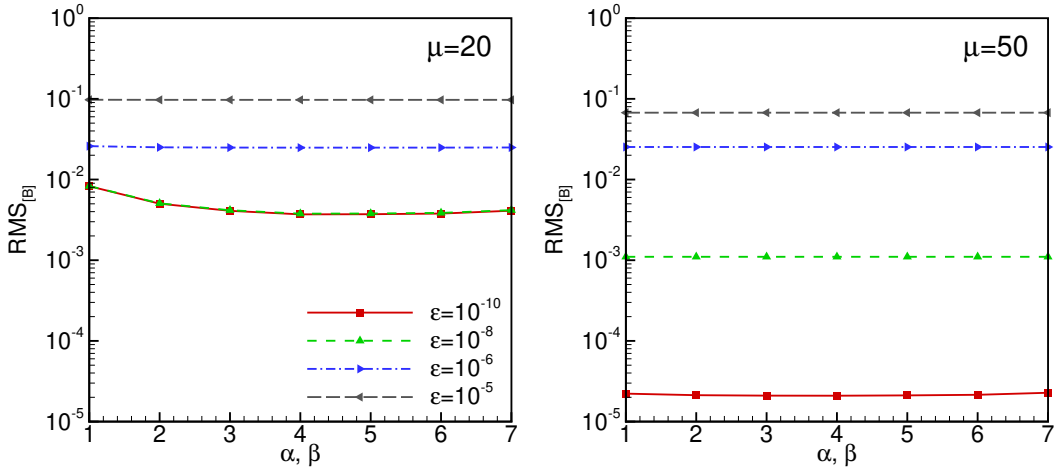

Figure 12: The influence of ACA on the $RMS_{[B]}$, for $\mu = 20$ (left), $\mu = 50$ (right) and different stopping condition ε. Mesh density 25^3. The approximation interval for the integral kernel $\gamma = 3$.

In Fig.12, we present the norm $RMS_{[B]}$ depending of the ACA stopping condition. When the parameter ε is low, the norm $RMS_{[B]}$ is of the same accuracy as the approximation error of the integral kernel. The approximation accuracy does not equal ε. However, a larger value for parameter ε lowers the accuracy of the approximation. Results show a lower ε gives a better approximation. The ACA approximation error is larger when the value of ε is larger. In Fig.12, we observed that at a point the ACA error is larger, then the approximation accuracy of the integral kernel. Thus, increasing the interpolation order does not lead to a better approximation accuracy of the integral kernel.

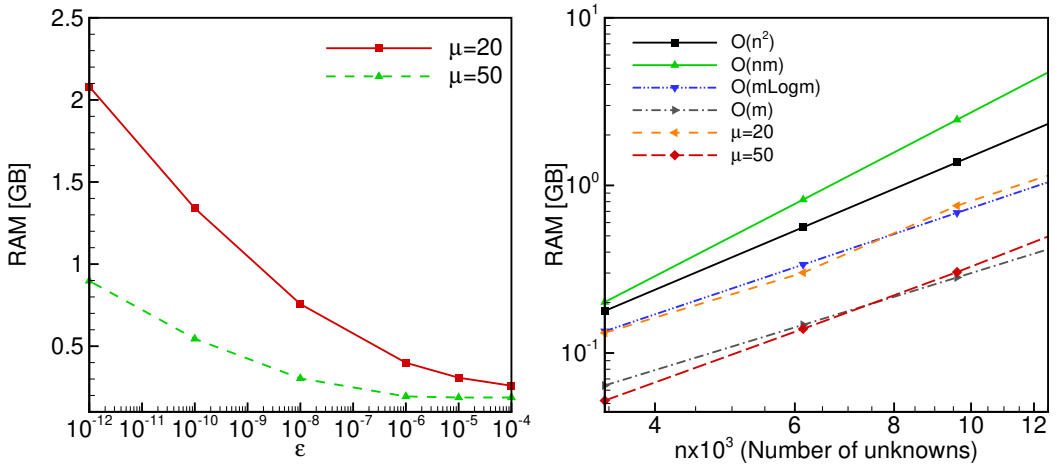

Figure 13: The amount of memory is needed to store the matrix [B]. We changed the stopping condition of ACA (left) and the number of unknowns n (right), the stopping condition was set to $\varepsilon = 10^{-8}$. The order of interpolation is $\alpha, \beta = 5$ and $\gamma = 3$.

The ACA algorithm in combination with the $\mathscr{H}^2$-matrix, reduces the cost of storing the domain integral kernel, which is assembled in matrix $[B]$ in memory. In Fig.13, we present the memory usage depending on the ACA stopping condition (left) and the number of unknowns n (right). The developed method reduced the computational cost from $\mathscr{O}(nm)$ to a logarithmic linear $\mathscr{O}(m\log m)$. The change of the modified Helmholtz fundamental solution from global characteristics to a more local also reduces memory consumption.

5 Conclusions

The poroelastodynamic BEM in Laplace- or frequency domain is well established. For the u-p-formulation four degrees of freedom per node are required. This fact and the quadratic complexity of a standard approach using an iterative solver does not allow to compute even medium sized problems in 3D. Fast methods have been developed for BE formulations, which reduces the complexity to almost linear order. Here, the fast multipole method (FMM) is applied to reduce the complexity of the poroelastodynamic BEM also to an almost linear order. Due to the complicated fundamental solutions a black-box type FMM is preferable.

The presented black-box FMM-BEM uses the Chebyshev interpolation-based FMM. This, firstly, reduces the number of evaluations of the computationally expensive fundamental solution significantly. Secondly, for large problems, the storage requirement of the FMM-BEM operators is almost linear. A significant compression, though, was only achieved using the SVD to compress the M2L-operators. In the tests, asymptotically, an almost linear behavior with respect to the CPU-time has been observed. It must be remarked that the iterative solver works only if the problem is transformed to dimensionless variables. The presented results show that the proposed FMM can be adjusted such that the convergence rates of the standard BEM can be obtained. An open point of the proposed method is to formulate an effective preconditioner to reduce the iteration numbers. However, this is independent of the proposed FMM.

Acknowledgement The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0196).

References

[1] S. E. Mikhailov, N. A. Mohamed, Numerical solution and spectrum of boundary-domain integral equation for the Neumann BVP with a variable coefficient, International Journal of Computer Mathematics 89 (11) (2012) 1488–1503.

[2] C. Fresneda-Portillo, Boundary-domain integral equations for the diffusion equation in inhomogeneous media based on a new family of parametrices, Complex Variables and Elliptic Equations (2019) 0–15.

[3] P. W. Partridge, C. A. Brebbia, Computer implementation of the BEM dual reciprocity method for the solution of general field equations, Communications in applied numerical methods 6 (1990) 83–92.

[4] A. H. Cheng, O. Lafe, S. Grille, Dual-reciprocity BEM based on global interpolation functions, Engineering Analysis with Boundary Elements 13 (1994) 303–311.

[5] Y. Ochiai, Three-dimensional steady thermal stress analysis by triple-reciprocity boundary element method, International journal for numerical methods in engineering 63 (2005) 1741–1756.

[6] S. Guo, Q. Wu, J. Gu, W. Wang, X. Li, An improved implementation of triple reciprocity boundary element method for three-dimensional steady state heat conduction problems, Engineering Analysis with Boundary Elements 107 (2019) 1–11.

[7] C. Lubich, R. Schneider, Time discretization of parabolic boundary integral equations, Numerische Mathematik 63 (1992) 455–481.

[8] M. Messner, M. Schanz, J. Tausch, A fast Galerkin method for parabolic space – time boundary integral equations, Journal of Computational Physics 258 (2014) 15–30.

[9] G. D. Mallinson, G. de Vahl Davis, The method of the false transient for the solution of coupled elliptic equations, Journal of Computational Physics 12 (1973) 435–461.

[10] G. Guj, F. Stella, A vorticity-velocity method for the numerical solution of 3D incompressible flows, Journal of Computational Physics 106 (2) (1993) 286–298.

[11] M. Behnia, F. Stella, G. Guj, A numerical study of three-dimensional combined buoyancy and thermocapillary convection, International Journal of Multiphase Flow 21 (3) (1995) 529–542.

[12] H. Cheng, J. Huang, T. J. Leiterman, An adaptive fast solver for the modified Helmholtz equation in two dimensions, Journal of Computational Physics 211 (2) (2006) 616–637.

[13] M. Hriberšek, L. Škerget, Fast boundary-domain integral algorithm for the computation of incompressible fluid flow problems, International Journal for Numerical Methods in Fluids 31 (5) (1999) 891–907.

[14] M. Cui, B. B. Xu, W. Z. Feng, Y. Zhang, X. W. Gao, H. F. Peng, A radial integration boundary element method for solving transient heat conduction problems with heat sources and variable thermal conductivity, Numerical Heat Transfer, Part B: Fundamentals 73 (2018) 1–18.

[15] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, Journal of Computational Physics 73 (2) (1987) 325–348.

[16] E. Darve, The Fast Multipole Method: Numerical Implementation, Journal of Computational Physics 160 (1) (2000) 195–240.

[17] M. Messner, M. Schanz, J. Tausch, An Efficient Galerkin Boundary Element Method for the Transient Heat Equation, SIAM Journal on Scientific Computing 37 (3) (2014) 1554–1576.

[18] I. Daubechies, Ten Lectures of Wavelets, 1992.

[19] W. Hackbusch, Sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices, Computing 62 (2) (1999) 89–108.

[20] D. Kalman, A Singularly Valuable Decomposition: The SVD of a Matrix, The College Mathematics Journal 27 (1) (1996) 2.

[21] M. Bebendorf, R. Grzibovski, Accelerating Galerkin BEM for Linear Elasticity using Adaptive Cross Approximation, Mathematical Methods in the Applied Sciences 29 (14) (2006) 1721–1747.

[22] S. Rjasanow, O. Steinbach, The Fast Solutuion of Boundary Integral Equations, 2007.

[23] S. Börm, L. Grasedyck, Hybrid cross approximation of integral operators, Numerische Mathematik 101 (2) (2005) 221–249.

[24] A. M. Haider, M. Schanz, Adaptive Cross Approximation for BEM in elasticity, Journal ofTheoretical and Computational Acoustics 27 (01) (2019) 1–22.

[25] S. Rjasanow, L. Weggler, Matrix valued adaptive cross approximation, Mathematical Methods in the Applied Sciences 40 (7) (2017) 2522–2531.

[26] T. Grytsenko, A. N. Galybin, Numerical analysis of multi-crack large-scale plane problems with adaptive cross approximation and hierarchical matrices, Engineering Analysis with Boundary Elements 34 (5) (2010) 501–510.

[27] S. Rjasanow, L. Weggler, ACA accelerated high order BEM for Maxwell problems, Computational Mechanics 51 (4) (2013) 431–441.

[28] J. M. Tamayo, A. Heldring, J. M. Rius, Application of Multilevel Adaptive Cross Approximation (MLACA) to electromagnetic scattering and radiation problems, Proceedings of the 2009 International Conference on Electromagnetics in Advanced Applications, ICEAA '09 (2009) 178–181.

[29] L. S. Campos, É. L. de Albuquerque, L. C. Wrobel, An ACA accelerated isogeometric boundary element analysis of potential problems with non-uniform boundary conditions, Engineering Analysis with Boundary Elements 80 (2017) 108–115.

[30] D. C. Rodopoulos, T. V. Gortsas, S. V. Tsinopoulos, D. Polyzos, ACA/BEM for solving large-scale cathodic protection problems, Engineering Analysis with Boundary Elements 106 (2019) 139–148.

[31] J. Tibaut, L. Škerget, J. Ravnik, Acceleration of a BEM based solution of the velocity–vorticity formulation of the Navier–Stokes equations by the cross approximation method, Engineering Analysis with Boundary Elements 82 (2017) 17–26.

[32] J. Ravnik, A. Šušnjara, J. Tibaut, D. Poljak, M. Cvetkovi, Stochastic modelling of nanofluids using the fast Boundary-Domain Integral Method, Engineering Analysis with Boundary Elements 107 (2019) 185–197.

[33] J. Tibaut, J. Ravnik, Fast boundary-domain integral method for heat transfer simulations, Engineering Analysis with Boundary Elements 99 (2019) 222–232.

[34] S. Börm, Approximation of integral operators by H2-matrices with adaptive bases, Computing 74 (3) (2005) 249–271.

[35] M. Bebendorf, R. Venn, Constructing nested bases approximations from the entries of non-local operators, Numerische Mathematik 121 (4) (2012) 609–635.

[36] S. Börm, W. Hackbusch, H2-matrix approximation of integral operators by interpolation, Applied Numerical Mathematics 43 (1-2) (2002) 129–143.

[37] S. Börm, Efficient Numerical Methods for Non-local Operators, 2010.

[38] S. Börm, L. Grasedyck, W. Hackbusch, Hierarchical Matrices, 2006.