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

Published in: *Engineering Analysis with Boundary Elements, Volume 138, May 2022, pp 1-12, 2022.*

The final authenticated version is available online at :
https://doi.org/10.1016/j.enganabound.2022.01.019

Abstract

In engineering, several physical models result in inhomogeneous partial differential equations. A prototype of such an equation is the modified Helmholtz equation or also called Yukawa equation. It may result from fluid mechanics (false transient approach) or heat transfer if a semi-discretisation in time with a finite difference schema is applied. Using the Boundary Element Method for the numerical solution of such problems requires to solve a boundary-domain integral equation. The main drawback of all boundary element methods is the quadratic complexity, which exists as well for boundary-domain element methods.

Here, a fast approach based on the $\mathscr{H}^2$-concept is proposed. The focus is on the discretisation of the domain integral. Respective cluster trees for the domain and the boundary nodes are established. The integral kernels in admissible blocks are approximated with Lagrange interpolation. Further, a recompression is applied, which is here performed with a fully pivoted adaptive cross approximation. The numerical results show that the memory used to store the approximated matrices is logarithmic linear. Considering the matrix formulation of the integral kernel approximation one can reduce the storing space needed in memory to linear complexity.

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 than 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 a non-homogeneous right-hand side. Verhnjak et al. [3] presented a novel approach for two-way coupled simulations of multiphase flows within an Euler-Lagrange framework. 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 [4] used the dual reciprocity method to solve the Poisson equation. Cheng et al. [5] 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 [6] implemented the triple reciprocity method to solve the heat equation with heat sources. Guo et al. [7] 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. [8, 9]) or a semi-discretisation in time with a finite difference schema. The latter results in a domain integral similar to the treatment of non-linear terms and is called in the framework of fluid mechanics false transient. Malinson and Davis [10] presented this approach for the solution of a coupled elliptic equation. Stella and Guj [11] applied the false transient to solve the lid-driven cavity test case with a finite difference scheme. Behnia et al. [12] 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 [13]. Hriberšek and Škerget [14] employed the Yukawa equation to compute incompressible fluid flow problems. Cui et al. [15] 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 multipole method (FMM) is maybe one of the first (see, e.g., [16, 17]). Within the field of the heat equation, the formulation by Messner et al. [18] may be mentioned. As well the wavelet transform [19] can be used to accelerate the BEM. Alternatively, hierarchical matrices ($\mathcal{H}$-matrices) with some data compression techniques can be applied. Hackbusch [20] introduced a recursive hierarchical decomposition for the approximation of an asymptotic smooth function. The recursive hierarchical procedure is known as the $\mathcal{H}$-structure. After the $\mathcal{H}$-structure is formed an approximation method is employed. The most efficient compression method is the Singular Value Decomposition Method (SVD) [21]. However, the computational cost of the method scales $\mathcal{O}(n^3)$. Adaptive Cross Approximation (ACA) presented by Bebendorf [22] is an approximate alternative to the SVD and reduces the complexity to $\mathcal{O}(n \log n)$ for most elliptic operators. Two versions of the algorithm were presented: the partial pivoting and full pivoting ACA [23], where only the first makes sense to accelerate BEM. A variant called Hybrid Cross Approximation (HCA) has been presented by [25]. For numerical analysis in solid mechanics, Bebendorf and Grzibovski [22] employed the ACA to solve a linear elasticity problem with the Galerkin BEM. Heider and Schanz [26] implemented the ACA to solve elasticity problem based on an extension of the ACA given by Rjasanow and Weggler [27]. Grytsenko and Galybin [28] solved multi-crack large-scale problems with the ACA and the $\mathcal{H}$-matrix. Rjasanow and Weggler [29] employed the ACA with the $\mathcal{H}$-matrix formulation to solve Maxwell problems. Tamayo et al. [30] applied the multi-level ACA for electromagnetic and radiation examples. Campos et al. [31] presented an isogeometric BEM that was accelerated with the ACA for the analysis of potential problems. Recently Rodopulos et al. [32] 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. [33, 34] employed the $\mathcal{H}$-matrix and the ACA algorithm to accelerate the BDIM. Ravnik and Tibaut [35] employed the accelerated BDIM with the modified Helmholtz equation and ACA to solve the unsteady convection-diffusion problem with variable diffusion .

An improvement of the hierarchical matrix concept has been presented by Börm [36] and is called $\mathcal{H}^2$-matrix. The $\mathcal{H}^2$-matrix form is based on nested cluster basis functions [37]. In the $\mathcal{H}^2$-matrix, the integral kernel is mostly interpolated in the far-field with polynomials. Börm and Hackbusch [38] approximated the integral kernel with Lagrangian polynomials. For Helmholtz problems, Börm et al. [41] presented directional $\mathcal{H}^2$-matrices. The authors included in their analysis also the case of dissipative Helmholtz kernels.

The paper is split into five sections. Firstly, we present the governing equation that we used for the numerical investigation. Next, we present the integral equation with its discretisation. For this set of equations, we present the $\mathcal{H}^2$-matrix with a special focus on the domain integral. As well, a recompression with ACA is introduced. In section four, we discuss the results and in the last section, we summarise the findings of this paper.

2 Governing equations

Several physical problems result in parabolic inhomogeneous partial differential equations. An example from fluid mechanics is incompressible, laminar flow of a Newtonian fluid, where often the so-called false transient approach results in such an equation (see, e.g., [34]). Here, we consider a model problem, which may either be heat transfer or the false transient approach. Certainly, the same considerations hold for the other physical problems governed by such equations.

2.1 Governing equations

Let $\Omega \subset \mathbb{R}^3$ be a bounded Lipschitz domain and $\Gamma := \partial\Omega$ its boundary with the outward normal $\vec{n}$. The governing equation for the scalar field $u(\vec{x},t)$, e.g., the temperature field, is given with

$$\frac{\partial u(\vec{x},t)}{\partial t} = \nabla_x^2 u(\vec{x},t) + b^*(\vec{x},t) \quad \forall\, (\vec{x},t) \in \Omega \times (0,T) \tag{1}$$

where t presents the time with the final time T and $b^*(\vec{x},t)$ is a source term. Any material data are set to unity for simplicity. As usual, ∇_x denotes the Nabla-operator with respect to the spatial coordinate $\vec{x}$. To complete the physical setting initial and boundary conditions have to be described. Here, as model a Dirichlet problem with vanishing initial condition is used

$$u(\vec{x},t=0) = 0 \qquad \forall\, \vec{x} \in \Omega \;\; t=0 \tag{2a}$$

$$u(\vec{x},t) = \bar{u}(\vec{x},t) \qquad \forall\, \vec{x} \in \Gamma \times (0,T)\,, \tag{2b}$$

with the prescribed boundary data $\bar{u}(\vec{x},t)$. Let the time be discretised in steps $t_0 = 0, t_1, \dots, t_N = T$ with constant time steps Δt. A time discrete form of Eq.(1) is obtained by approximating the time derivative $\frac{\partial u(\vec{x},t)}{\partial t}$ by a first order finite difference scheme. This results for $n = 1,\dots,N$ in

$$\frac{u(\vec{x},t_n) - u(\vec{x},t_{n-1})}{\Delta t} = \nabla_x^2 u(\vec{x},t_n) + b^*(\vec{x},t_n)\,, \tag{3}$$

where $u(\vec{x},t_{n-1})$ denotes the value of the scalar field at the previous time step. Rearrangement and the abbreviations $\mu^2 = \frac{1}{\Delta t}$, $b(\vec{x},t_n) = \frac{1}{\Delta t}u(\vec{x},t_{n-1}) + b^*(\vec{x},t_n)$ results in the Yukawa equation

$$\left(\nabla_x^2 - \mu^2\right) u(\vec{x},t_n) + b(\vec{x},t_n) = 0\,. \tag{4}$$

This equation is also called modified Helmholtz equation. In case of large time steps $\Delta t \to \infty$, the parameter μ^2 tends to zero, i.e., $\mu^2 \to 0$. Thus in the limit, the Poisson equation is obtained.

In order to find the integral form of the Yukawa equation (4), we will employ Green's second identity. As a preliminary step the fundamental solutions are formulated. For a given point $\vec{\xi}$ and the Dirac distribution as source term, i.e., $b(\vec{x},t_n) = \delta(\vec{\xi} - \vec{x})$ the fundamental solution of the Yukawa equation (4) is given

$$u^*(\vec{\xi},\vec{x}) = \frac{e^{-\mu r}}{4\pi r}\,, \tag{5}$$

with $r = |\vec{\xi} - \vec{x}|$. The flux fundamental solution is as well needed and can be obtained by the normal derivative

$$q^*(\vec{\xi},\vec{x}) = \vec{n}(\vec{x}) \cdot \vec{\nabla}_x u^*(\vec{\xi},\vec{x}) = \frac{\vec{n}(\vec{x}) \cdot (\vec{\xi} - \vec{x})}{4\pi r^3}(1+\mu r)e^{-\mu r} \,. \tag{6}$$

As discussed above, in case of large time steps the parameter $\mu^2 \to 0$ and the fundamental solution in (6) tends to the fundamental solution of the Laplace equation. In Fig. 1, $u^*(\vec{\xi},\vec{x})$ is plotted versus the argument of the exponential function $r\mu$ for different values of μ. It can

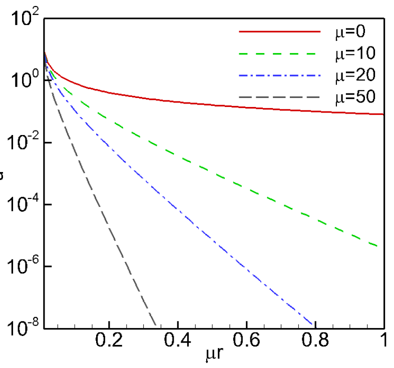

Figure 1: Fundamental solution u^* for different values of the μ

be observed that for small values of μ the fundamental solution behaves as that of the Laplace equation denoted with $\mu = 0$. The more interesting observation is the strong decay of the solution for large μ, i.e., for small time step sizes. This property will be used subsequently.

Either starting from Green's second identity or via a weighted residual statement the usual steps with partial integration and a suitable limit of the load point to the boundary results in the well known integral equation

$$c(\vec{\xi})u(\vec{\xi},t_n) + \int\limits_\Gamma \left[u(\vec{x},t_n)q^*(\vec{\xi},\vec{x}) - u^*(\vec{\xi},\vec{x})q(\vec{x},t_n)\right] d\Gamma = \int\limits_\Omega u^*(\vec{\xi},\vec{x})b(\vec{x},t_n)d\Omega \;\; \forall \vec{\xi} \in \Gamma, \tag{7}$$

where $q(\vec{x},t_n) = \vec{n}(\vec{x}) \cdot \vec{\nabla} u(\vec{x},t_n)$ is the flux of the scalar field $u(\vec{x},t_n)$. The boundary integral has kernels with a weak singularity and the integral free term $c(\vec{\xi})$ has the usual form (see, e.g., [23]). Note, the integral equation has to be evaluated at each time step t_n and the influence of the last time step is hidden in $b(\vec{x},t_n) = \frac{1}{\Delta t}u(\vec{x},t_{n-1}) + b^*(\vec{x},t_n)$. Including as well the boundary condition, we arrive at the integral equation to be solved

$$\int\limits_\Gamma u^*(\vec{\xi},\vec{x})q(\vec{x},t_n)d\Gamma = c(\vec{\xi})\bar{u}(\vec{\xi},t_n) + \int\limits_\Gamma \bar{u}(\vec{x},t_n)q^*(\vec{\xi},\vec{x})d\Gamma - \int\limits_\Omega u^*(\vec{\xi},\vec{x})b(\vec{x},t_n)d\Omega \;\; \forall \vec{\xi} \in \Gamma. \tag{8}$$

In the following, the argument t_n, indicating the actual time step, is skipped for the sake brevity.

2.2 Spatial discretization of the integral formulation

For the spatial discretisation of the integral equation (8), first, the boundary is divided into N boundary elements and the domain Ω is divided into M domain cells

$$\Gamma = \bigcup_{i=1}^{N} \Gamma_i, \quad \Omega = \bigcup_{j=1}^{M} \Omega_j . \tag{9}$$

Second, the boundary data are approximated with a continuous quadratic (9 nodes) interpolation for the primary field and a discontinuous linear ansatz (4 nodes) for the flux. The right hand side is approximated with a quadratic (27 nodes) interpolation, however, in this case a volume cell must be used, i.e., hexahedrons are assumed. The respective shape functions are

$$u(\vec{x}) \approx \sum_{a=1}^{9} u_a \varphi_a(\vec{x}), \qquad q(\vec{x},t) \approx \sum_{b=1}^{4} q_b \psi_b(\vec{x}), \qquad b(\vec{x}) \approx \sum_{c=1}^{27} b_c \Phi_c(\vec{x}) , \tag{10}$$

which are defined locally on each element. The locations of the nodes in these shape functions are sketched in Fig. 2. The distance of the nodes of the discontinuous shape functions to the

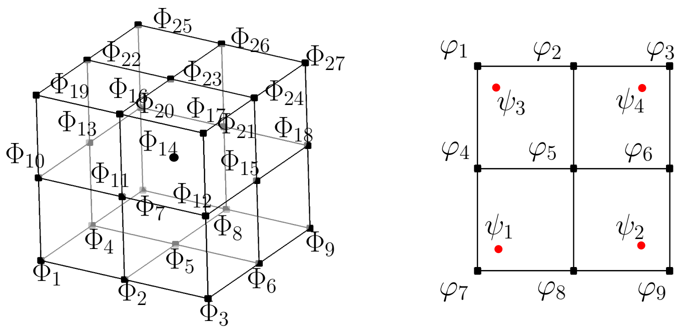

Figure 2: Locations of the nodes in the boundary elements and volume cells. These nodes are used in the shape functions in (10).

boundary is 0.25. The usage of discontinuous shape functions avoids any discussions on defining the normal vector. Further, it fits into the function spaces required for this elliptic problem. Inserting these shape functions and the panelisation of the geometry in (8) results in the discrete integral equation

$$\begin{aligned}\sum_{i=1}^{n} q_i \int\limits_{supp(\psi_i)} \psi_i(\vec{x}) u^*(\vec{\xi},\vec{x}) d\Gamma_i = \sum_{i=1}^{n^*} c(\vec{\xi}) u_i \varphi_i(\vec{\xi}) + u_i \int\limits_{supp(\varphi_i)} \varphi_i(\vec{x}) q^*(\vec{\xi},\vec{x}) d\Gamma_i \\ - \sum_{j=1}^{m} b_j \int\limits_{supp(\Phi_j)} \Phi_j(\vec{x}) u^*(\vec{\xi},\vec{x}) d\Omega_j ,\end{aligned} \tag{11}$$

where index i restricts the shape functions φ_i and ψ_i to the boundary element Γ_i and j restricts the shape function Φ_j to the domain cell Ω_j. The upper limit of the second sum n^* indicates that this is not the number of elements but the number of nodes. Further, $n = 4N$ as discontinuous shape functions are used. In the last sum, m is the number of domain nodes. Hence, the sums over shape functions and nodes are collected to one sum and the respective integration domain is marked with the support of the shape functions.

The integral equation will be solved with a collocation approach. For the selected Dirichlet problem, the collocation points are at the nodes of the shape functions of the flux (ψ_i in Fig. 2) and are denoted with $\vec{\xi}_k$. Hence, the total number of collocation points is $n = 4N$. Writing this in a matrix form the discrete equation system is

$$[G]\{q\} = [H]\{u\} - [B]\{b\}, \tag{12}$$

with the matrix elements

$$\begin{aligned} h_{ki} &= c(\vec{\xi}_k)\varphi_i(\vec{\xi}_k) + \int\limits_{supp(\varphi_i)} \varphi_i(\vec{x})q^*(\vec{\xi}_k,\vec{x})d\Gamma_i \\ g_{ki} &= \int\limits_{supp(\psi_i)} \psi_i(\vec{x})u^*(\vec{\xi}_k,\vec{x})d\Gamma_i \\ b_{kj} &= \int\limits_{supp(\Phi_j)} \Phi_j(\vec{x})u^*(\vec{\xi}_k,\vec{x})d\Omega_j\,. \end{aligned} \tag{13}$$

Hence, the matrices are of size $[G] = [n \times n], [H] = [n \times n^*]$, and $[B] = [n \times m]$.

3 $\mathscr{H}$-structure

To obtain a fast BE formulation one option is to use hierarchical matrices. Here, the $\mathscr{H}^2$ approach will be applied, where the focus is on the domain integral, i.e., the approximation of matrix $[B]$ with the size $n \times m$. We form two different cluster trees, one for the domain and the other for the boundary. Note that we have in the domain hexahedrons and on the boundary quadrilaterals.

3.1 The Cluster tree

To form uniform cluster trees we use a bottom-up approach based on equal sized bounding boxes [33]. The bottom-up approach combines the neighboring cells and forms new clusters on a new level. For each level higher, cluster pairs form a new cluster. The procedure is repeated until one cluster that presents the whole domain is on the top level. We will denote the cluster tree built from the boundary elements as T_J and the domain cluster tree as T_I. The cluster tree T_J has pb levels and T_I has pd levels. The domain tree has more levels $(pb < pd)$ because the number of domain cells is larger than the number of boundary elements (see the example in Fig. 4). In Fig.3, we illustrate clusters that form the boundary element cluster tree T_J with their index sets $J_i^{(k)}$. In the following, we will denote the clusters by their index sets. Moving from top to bottom level each cluster is split into son clusters. The bottom level is called leaf level.

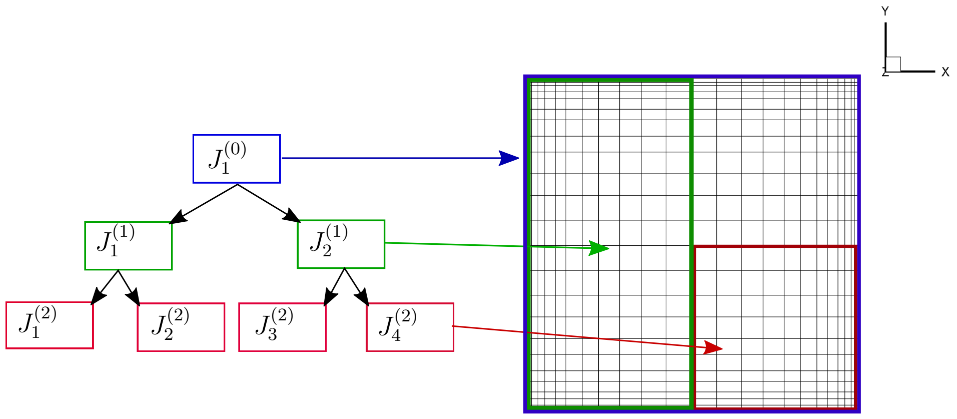

Figure 3: Illustration of the cluster size on each level of the cluster tree T_J. From top to the bottom of the cluster tree the size of clusters decreases.

3.2 Block cluster tree

The block cluster tree $T_{J\times I}$ is a combination of clusters in the cluster tree T_I and T_J. The block clusters in $T_{J\times I}$ form rectangular blocks $[\hat{B}]_{\hat{n}\times\hat{m}}$ with size $\hat{n}\times\hat{m}$, where $\hat{n}<\hat{m}$ holds for sufficiently large meshes. However, square blocks would be preferable. If the clusters are combined at different levels of both cluster trees it is possible to get nearly square blocks. Hence, cluster $I_i^{(k)}$ is combined either with a cluster in T_j one level higher $J_j^{(k-1)}$, or on the same level $J_j^{(k)}$, or one level lower $J_j^{(k+1)}$. In Fig.4, the three yellow arrows indicate these choices exemplarily. In the following, the indication of the level of the index sets are skipped for the sake of brevity. Clusters at the leaf level of the cluster tree T_I are combined to block clusters with the clusters

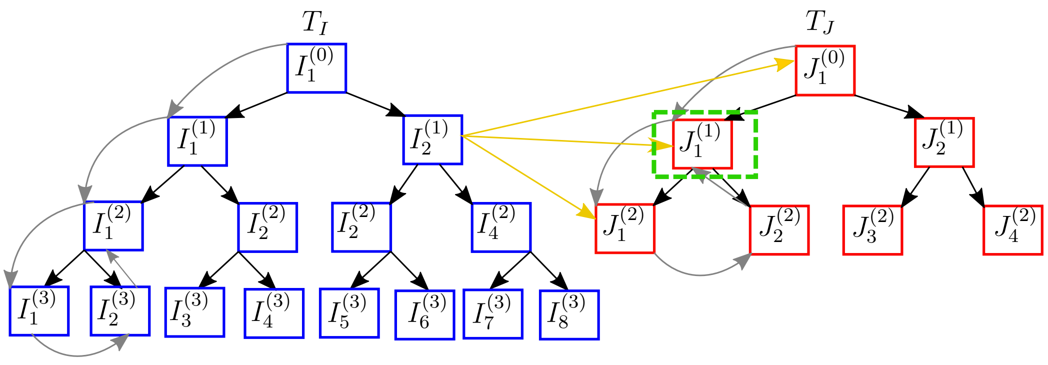

Figure 4: Domain block cluster tree $T_{J\times I}$ of the clusters T_I and T_J. Yellow arrows indicate the different choices to form nearly square blocks.

at the leaf level of T_J. Similar to the above sketched procedure for the domain block cluster tree $T_{J\times I}$ the block cluster tree for the boundary $T_{J\times J}$ is build. Two identical T_J cluster trees are combined to form blocks $J_i\times J_j$. Different to the above, the block clusters are only build within

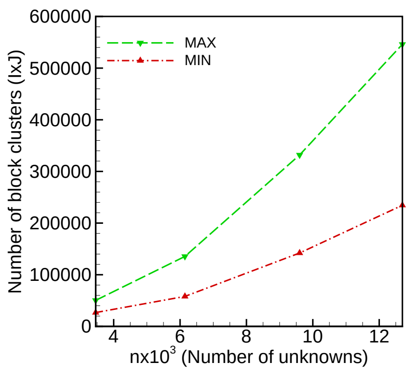

Figure 5: Number of bock cluster $J \times I$ that form the $\mathscr{H}$-structure depending on the admissibility condition (15) (MIN) and (14) (MAX).

one level.

The cluster pairs constructed as described above form nearly a square block ($\hat{n} \approx \hat{m}$) and are tested on admissibility. Following the literature [42], for $\mathscr{H}^2$-matrices the criterion

$$max\{dim(I_i), dim(J_j)\} \leq \eta \; dist(I_i, J_j) \tag{14}$$

is used, where $dim(I_i)$ and $dim(J_j)$ are the cluster sizes determined by the largest diagonal of the cluster. $dist(I_i, J_j)$ is the minimal distance between the clusters and parameter η is defined by the user. Each block cluster fulfilling (14) is called admissible. Inadmissible blocks are split into smaller block clusters, i.e., one goes one level down. Block clusters on the lowest level of the block cluster tree that do not fulfill the admissibility condition are considered inadmissible and constitute the near field. For $\mathscr{H}$-matrices usually a different condition based on the minimum of both clusters is applied

$$min\{dim(I_i), dim(J_j)\} \leq \eta \; dist(I_i, J_j) \; . \tag{15}$$

The criterium (15) creates larger admissible blocks compared to (14), which is exemplarily shown in Fig.5. There, the number of block clusters $J \times I$ is presented for matrix $[B]$ for a unit cube. Four mesh densities were chosen. It can be observed that the admissibility condition (15) forms less block clusters than (14). Hence, condition (14) formed smaller block cluster than condition (15). However it must be remarked that the minimum based condition (15) has no mathematical basis if the kernel is interpolated as subsequently done. Nevertheless, both versions will be compared in the following.

In case of small block clusters, it may happen that a kernel interpolation produces a larger matrix block as it would be without interpolation, i.e., to store the dense matrix block needs less storage than the interpolated matrix block. The reason is that the interpolation degree is to the power of three. To avoid such situations an additional criterion is introduced. If an admissible block cluster is smaller than a predefined value, i.e.,

$$dist(I_i, J_j) \leq dist_m, \tag{16}$$

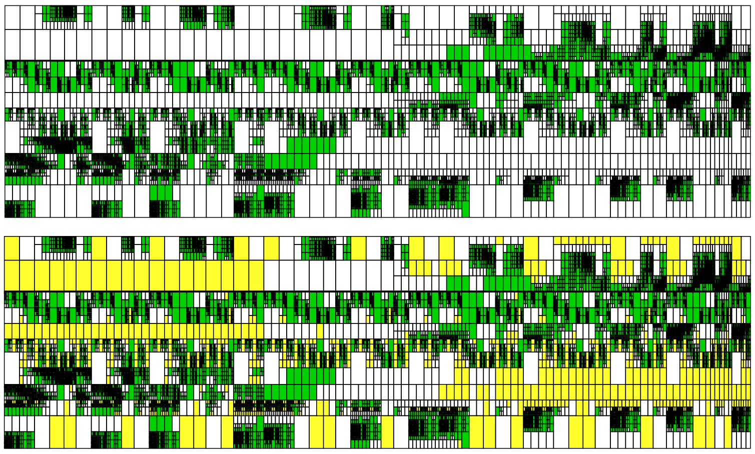
Figure 6: Sketch of two matrices for $\eta = 5, dist_m = 0.25$ and $\mu = 20$ (top), $\mu = 50$ (bottom). The inadmissible blocks are black, admissible blocks from condition (17) are yellow, admissible blocks from condition (16) are green and white admissible blocks are to be approximated. Admissibility condition (15) was employed.

holds, no kernel interpolation will be applied but the fundamental solution itself is used. Hence, a dense matrix block would be stored, however ACA is used to reduce storage. For this study we set a constant value $dist_m = 0.25$. Condition (16) is only checked after the admissibility condition holds to ensure that a low rank representation with ACA is possible.

Additionally to the above given clustering some block clusters have elements with very small values. This is caused by the fundamental solution (5) for larger values of μ. In Fig. 1, the values of the fundamental solution have been plotted with respect to the distance and μ. It can be observed that the matrix entries decrease by several decades if the distance r is increased, hence become negligible compared to the other entries. Further, such small values may even cause trouble in a kernel expansion as, essentially, a zero is approximated. To avoid such problems and to save storage, blocks which fulfil

$$||[\hat{B}]_{\hat{m}\times\hat{n}}|| \leq 10^{-15} \tag{17}$$

are set to zero. The condition is realized in the code by checking the fundamental solution for the smallest distance of this cluster to the boundary.

The above conditions are checked in a hierarchy, first (17) then (15) or (14) and lastly (16). In Fig.6, we present two examples of $\mathscr{H}^2$-matrices constructed in this way. For the top matrix, the parameter $\mu = 20$ is set and below the value is $\mu = 50$. Inadmissible matrix blocks are coloured in black. Matrix blocks fulfilling (17) are marked yellow and are not stored. Those blocks fulfilling (16) are green and are compressed with ACA. For the remaining white matrix blocks a kernel interpolation is applied with recompression (see below). Obviously, the amount of yellow blocks increases with a larger μ.

3.3 Approximation of the integral kernel

A low-rank representation of the kernel is obtained by interpolation in admissible blocks, which do not fulfill (17) or (16). Such blocks exist for all three matrices $[G],[H]$, or $[B]$ of (12), which are detailed in (13). The blocks in $[H]$ and $[G]$ stem from block clusters $J_i \times J_j$ and those in $[B]$ stem from block clusters $J_j \times I_i$. Within such blocks we approximate the fundamental solution $u^*\left(\vec{\xi},\vec{x}\right)$ with the Lagrange interpolation function [38]

$$u^*\left(\vec{\xi},\vec{x}\right) \approx \sum_{\iota=1}^{\alpha^3} \sum_{\kappa=1}^{\beta^3} \mathscr{L}_\iota\left(\vec{\xi}\right) u^*\left(\vec{\xi}_\iota,\vec{x}_\kappa\right) \mathscr{L}_\kappa(\vec{x}), \tag{18}$$

where α^3 is the number of interpolation points in the cluster J_j, β^3 is the number of interpolation points in the cluster I_i. Note, the interpolation is for all three coordinate directions, which gives overall α^3 interpolation points, when each coordinate axis has α interpolation points. The Lagrange interpolation function is defined as usual

$$\mathscr{L}_\iota(\vec{\xi}) = \prod_{\iota_1 \neq \ell} \frac{\xi^1 - \xi^1_\ell}{\xi^1_{\iota_1} - \xi^1_\ell} \times \prod_{\iota_2 \neq \ell} \frac{\xi^2 - \xi^2_\ell}{\xi^2_{\iota_2} - \xi^2_\ell} \times \prod_{\iota_3 \neq \ell} \frac{\xi^3 - \xi^3_\ell}{\xi^3_{\iota_3} - \xi^3_\ell}, \ \iota_1,\iota_2,\iota_3,\ell = 1,\ldots,\alpha^3, \tag{19}$$

with the zeros of the Chebyschev polynomial $\vec{\xi}_\ell$. The second Lagrange interpolation in (18) is defined analogously but there β^3 interpolation points are used. In the matrix $[H]$ in (13), the normal derivative $q^* = \vec{n}(\vec{x}) \cdot \vec{\nabla}_x u^*(\vec{\xi},\vec{x})$ of the fundamental solution is present, which is realised by applying the gradient operator on the Lagrange interpolation

$$q^*\left(\vec{\xi},\vec{x}\right) \approx \sum_{\iota=1}^{\alpha^3} \sum_{\kappa=1}^{\beta^3} \mathscr{L}_\iota\left(\vec{\xi}\right) u^*\left(\vec{\xi}_\iota,\vec{x}_\kappa\right) \left(\vec{n}(\vec{x}) \cdot \vec{\nabla}_x \mathscr{L}_\kappa(\vec{x})\right). \tag{20}$$

These approximations of the fundamental solutions are used in the block cluster tree. Considering the block clusters $J_i \times J_j$ with the size $\hat{n} \times \hat{n}$ and $J_j \times I_i$ with the size $\hat{n} \times \hat{m}$, then the entries are

$$\begin{aligned}
\hat{h}_{ki} &= \sum_{\iota=1}^{\alpha^3} \sum_{\kappa=1}^{\beta^3} \mathscr{L}_\iota\left(\vec{\xi}_k\right) u^*\left(\vec{\xi}_\iota,\vec{x}_\kappa\right) \int\limits_{supp(\varphi_i)} \varphi_i(\vec{x})(\vec{n}(\vec{x}) \cdot \vec{\nabla}_x \mathscr{L}_\kappa(\vec{x}))d\Gamma_i, \\
\hat{g}_{ki} &= \sum_{\iota=1}^{\alpha^3} \sum_{\kappa=1}^{\beta^3} \mathscr{L}_\iota\left(\vec{\xi}_k\right) u^*\left(\vec{\xi}_\iota,\vec{x}_\kappa\right) \int\limits_{supp(\psi_i)} \psi_i(\vec{x}) \mathscr{L}_\kappa(\vec{x})\, d\Gamma_i, \\
\hat{b}_{kj} &= \sum_{\iota=1}^{\alpha^3} \sum_{\kappa=1}^{\beta^3} \mathscr{L}_\iota\left(\vec{\xi}_k\right) u^*\left(\vec{\xi}_\iota,\vec{x}_\kappa\right) \int\limits_{supp(\Phi_j)} \Phi_j(\vec{x}) \mathscr{L}_\kappa(\vec{x})\, d\Omega_j.
\end{aligned} \tag{21}$$

where index k determines the $\hat{n}$-th row, i the $\hat{n}$-th column and j is the $\hat{m}$-th column. For block cluster $J_i \times J_j$ cluster J_i has β^3 and J_j has α^3 interpolation points. While block cluster $I_i \times J_j$ has β^3 and α^3 interpolation points. Note, for the elements $\hat{h}_{ki}$ there is no integral free term. The integral free term is located on the main diagonal of $[H]$ and such blocks can never be admissible. Writing the above in a matrix notation for the whole block cluster results in

$$[\hat{H}] = [\hat{U}][\hat{S}][\hat{V}_H], \qquad [\hat{G}] = [\hat{U}][\hat{S}][\hat{V}_G], \qquad [\hat{B}] = [\hat{U}][\hat{S}][\hat{V}_B], \tag{22}$$

with the elements

$$\hat{S}_{\iota\kappa} = u^*\left(\vec{\xi}_\iota, \vec{x}_\kappa\right), \qquad \hat{U}_{k\iota} = \mathscr{L}_\iota\left(\vec{\xi}_k\right), \qquad (\hat{V}_H)_{\kappa i} = \int\limits_{supp(\varphi_i)} \varphi_i(\vec{x})(\vec{n}(\vec{x})\cdot\vec{\nabla}_x\mathscr{L}_\kappa(\vec{x}))d\Gamma_i,$$

$$(\hat{V}_G)_{\kappa i} = \int\limits_{supp(\psi_i)} \psi_i(\vec{x})\mathscr{L}_\kappa(\vec{x})\,d\Gamma_i, \qquad (\hat{V}_B)_{\kappa j} = \int\limits_{supp(\Phi_j)} \Phi_j(\vec{x})\mathscr{L}_\kappa(\vec{x})\,d\Omega_j\,.$$

Those matrices collect the entries for all nodes in the block cluster, i.e., the sizes are $[\hat{U}] = [\hat{n}\times\alpha^3], [\hat{S}] = [\alpha^3\times\beta^3]$ and for the right interpolation matrices $[\hat{V}_H] = [\beta^3\times\hat{n}^*], [\hat{V}_G] = [\beta^3\times\hat{n}], [\hat{V}_B] = [\beta^3\times\hat{m}]$.

Following the $\mathscr{H}^2$-strategy, a nested cluster basis is introduced to end up with an almost linear complexity [37]. With nested cluster basis we can express polynomials corresponding to clusters J in terms of polynomials corresponding to boundary elements on the leaf cluster. The same holds for cluster I. Essentially, the Lagrange interpolant in (21) is again interpolated

$$\mathscr{L}_\iota(\vec{x}) = \sum_{\lambda=1}^{\gamma^3}\mathscr{L}_\iota(\vec{x}_\lambda)\mathscr{L}'_\lambda(\vec{x}), \qquad \vec{n}(\vec{x})\cdot\vec{\nabla}_\eta\mathscr{L}_\kappa(\vec{x}) = \sum_{\lambda=1}^{\gamma^3}\mathscr{L}_\kappa(\vec{x}_\lambda)\left(\vec{n}(\vec{x})\cdot\vec{\nabla}_x\mathscr{L}'_\lambda(\vec{x})\right), \tag{23}$$

where $\mathscr{L}'$ is of this form (19). The equal sign holds only if the interpolation order γ is large enough with respect to α,β [42]. The nested cluster basis replace the Lagrange interpolation functions in the elements $\hat{U}_{k\iota}$, $(\hat{V}_H)_{\kappa i}$, $(\hat{V}_G)_{\kappa i}$ and $(\hat{V}_B)_{\kappa j}$ of (22)

$$\begin{aligned}
\hat{U}_{k\iota} &= \sum_{\lambda=1}^{\gamma^3}\mathscr{L}_\iota(\vec{x}_\lambda)\mathscr{L}'_\lambda(\vec{x}),\\
(\hat{V}_H)_{\kappa i} &= \sum_{\lambda=1}^{\gamma^3}\mathscr{L}_\kappa(\vec{x}_\lambda)\int\limits_{supp(\varphi_i)}\varphi_i(\vec{x})(\vec{n}(\vec{x})\cdot\vec{\nabla}_x\mathscr{L}'_\lambda(\vec{x}))d\Gamma_i,\\
(\hat{V}_G)_{\kappa i} &= \sum_{\lambda=1}^{\gamma^3}\mathscr{L}_\kappa(\vec{x}_\lambda)\int\limits_{supp(\psi_i)}\psi_i(\vec{x})\mathscr{L}'_\lambda(\vec{x})\,d\Gamma_i,\\
(\hat{V}_B)_{\kappa j} &= \sum_{\lambda=1}^{\gamma^3}\mathscr{L}_\iota(\vec{x}_\lambda)\int\limits_{supp(\Phi_j)}\Phi_j(\vec{x})\mathscr{L}'_\lambda(\vec{x})d\Omega_j\,.
\end{aligned} \tag{24}$$

The elements under the boundary and domain integral are stored separately in leaf basis matrices

$$\begin{aligned}
T_\lambda &= \mathscr{L}'_\lambda(\vec{x}), \qquad (T_H)_{i\lambda} = \int\limits_{supp(\varphi_i)}\varphi_i(\vec{x})(\vec{n}(\vec{x})\cdot\vec{\nabla}_x\mathscr{L}'_\lambda(\vec{x}))d\Gamma_i,\\
(T_G)_{i\lambda} &= \int\limits_{supp(\psi_i)}\psi_i(\vec{x})\mathscr{L}'_\lambda(\vec{x})\,d\Gamma_i, \qquad (T_B)_{j\lambda} = \int\limits_{supp(\Phi_j)}\Phi_j(\vec{x})\mathscr{L}'_\lambda(\vec{x})d\Omega_j\,,
\end{aligned} \tag{25}$$

where $[T] = T_\lambda$, $[T_H] = (T_H)_{i\lambda}$, $[T_G] = (T_G)_{i\lambda}$ and $[T_B] = (T_B)_{j\lambda}$. For the leaf basis matrices the elements have to be integrated corresponding to the boundary element and domain cell. The sizes of the leaf basis matrices are $[T] = [\gamma^3], [T_H] = [T_G] = [N \times 9\gamma^3]$ and $[T_B] = [M \times 27\gamma^3]$. Please note, only the leaf basis matrices are stored in the memory. The remaining polynomials can either be computed on the fly during the matrix-vector product, which increases its computation time or these interpolations are pre-computed and stored per level.

To speed up the matrix-vector product and to save storage the matrix $[\hat{S}]$ from equation (22) is compressed with ACA, which is denoted recompression because it is already an approximation. This matrix consists of the fundamental solution evaluated at a set of interpolation nodes (18). As mentioned above, the size of this matrix is the interpolation order to the power of three. Here, the fully pivoted ACA is used (see, e.g., [23]). Essentially, a low-rank decomposition is sought after

$$[\hat{S}]_{\alpha^3 \times \beta^3} = [A^*]_{\alpha^3 \times k} [B^*]_{k \times \beta^3} \ , \tag{26}$$

where $k < \alpha, \beta$ should hold. The ACA approximates these low-rank matrices with $[\hat{S}^k] = \sum_{m=1}^{k} \vec{a}_m \vec{b}_m^T$ and the approximation algorithm can be sketched as follows:

```
• Set $R^0 = [\hat{S}]$
• For $\ell = 1, 2, \dots, k$
    1. $(i^*, j^*)^\ell = ArgMax|R^{\ell-1}|$
    2. $\tau^\ell = (R^{\ell-1}_{i^*,j^*})^{-1}$
    3. $\vec{a}_\ell = \tau^\ell R^{\ell-1}_{:,j^*}$, $\vec{b}_\ell = (R^{\ell-1}_{i^*,:})^T$
    4. $R^\ell = R^{\ell-1} - \vec{a}_\ell \vec{b}_\ell$ ,$\hat{S}^\ell = \hat{S}^{\ell-1} + \vec{a}_\ell \vec{b}_\ell$
• If $(\left\|R^\ell\right\|_F \leq \varepsilon \left\|\hat{S}^\ell\right\|_F \vee \ell = k)$ Stop
• EndFor
```

The notation $R_{i,:}$ denotes the i-th column of R and analogously $R_{:,j}$ the row. 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 $\vec{a}_\ell$ and $\vec{b}_\ell$ are set as the row and column of the residual matrix related to the maximal element. Then the residual matrix R^ℓ is computed by subtracting the outer product of $\vec{a}$ and $\vec{b}$, i.e., its rank is reduced by one. In the last step, the Frobenius norm of the residuum $\left\|R^\ell\right\|_F$ is calculated. The steps from two to six are repeated until the stopping condition is satisfied or the maximal rank of matrix $[\hat{S}]$ is reached. The value of the stopping condition ε is set a priori by the user.

4 Numerical test

In this section, we present numerical tests to show the performance of the proposed methodology. The considerations are restricted to the matrix $[B]$ in (12) as this matrix is the largest of the involved matrices. Further, both other matrices $[H]$ and $[G]$ are based only on the boundary discretisation and have been studied in other publications on $\mathscr{H}^2$-matrices for the Laplace operator. Essentially, three test are presented. First, to validate the code, the fundamental solution $u^*\left(\vec{\xi}_\iota, \vec{x}_\kappa\right)$ in $[B]$ is replaced by a polynomial

$$f(x) = x^z, \tag{27}$$

with degree z. Hence, the domain integral in this test is

$$\int f(x)\Phi(x)d\Omega = \int x^z\Phi(x)d\Omega\,. \tag{28}$$

We expected an exact solution for $z = \alpha = \beta = \gamma$ because in this case the kernel expansion is exact. The second and third test show the behavior of the approximation of $[B]$ for the Yukawa fundamental solution as proposed above. In the second test, the $\mathscr{H}^2$-methodology is applied without recompression, whereas in the third test the ACA is used for recompression of the matrix $[\hat{S}]_{\alpha^3\times\beta^3}$.

In all tests, the relative root mean square error defined by

$$RMS_{[B]} = \left(\frac{\left(\|B - \tilde{B}\|\right)^2}{\|B\|^2}\right)^{\frac{1}{2}} \tag{29}$$

is used to measure the approximation. $\|B\|$ is the Frobenius norm of the original matrix $[B]$ and $\tilde{B}$ denotes the approximated matrix.

The test geometry is a three-dimensional unit cube. Hexahedral elements are used in the domain cells. The mesh density of the computational domain was varied from $9^3 = 729$ to $91^3 = 753571$ nodes.

4.1 Code validation

As written above, first, we does not use the fundamental solution but a polynomial of degree z as kernel function. Please, note that the integral (28) consists of the polynomial and the shape functions. In Fig.7, the $RMS_{[B]}$ is displayed versus the approximation, i.e., the degree of the Lagrange polynomials. When the number of interpolation points α and β is equal to the degree of the polynomial z the error drops to a very small value close to machine precision. As expected, the accuracy increases with increasing the number of interpolation points.

The parameter γ governs the interpolation (23) within the levels. Its influence is studied in the right panel of Fig.7, where $\gamma = 5$ was set. Also as expected, γ has to grow with α and β, else an error is introduced, which gives the straight line after $\alpha = \beta > \gamma$ for a $z < \gamma$.

4.2 Results for the Yukawa kernel

In the following tests, the kernel is the fundamental solution $u^*(\vec{\xi}, \vec{x})$ from (5). Please, remember that the case $\mu = 0$ represents the case of a Poisson equation with a right hand side $\vec{b}^*(\vec{x})$. This right hand side is approximated by the shape function in (10). The following results are again

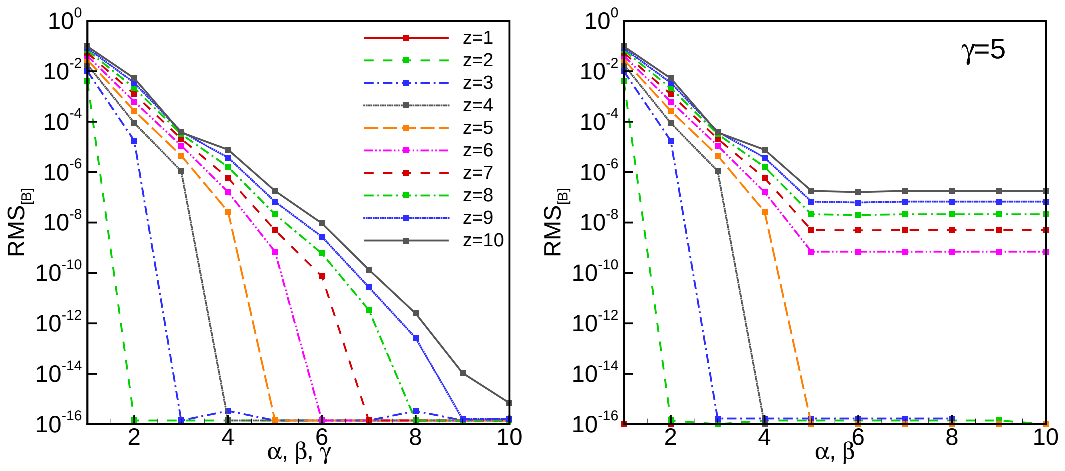

Figure 7: The norm $RMS_{[B]}$ is plotted versus the number of interpolation points $\alpha = \beta$ for a polynomial as integral kernel (27). In the left figure $\gamma = \alpha = \beta$ is set, whereas in the right figure $\gamma = 5$ is used. The mesh density was 9^3 nodes.

restricted to show the approximation of matrix $[B]$ and, hence, are valid for any right hand side which can be approximated by these shape functions. Please note, for these results the criterium (16) was disabled.

In Fig.8 and Fig.9, we present the accuracy of the approximated $[B]$ for different interpolation orders without recompression. Results are displayed for four different meshes and two parameter choices η. In both figures, the results in the upper line are produced with the maximum based admissibility criterium (14) and the lower line with the minimum based admissibility criterium (15). Obviously, the results depend on η, where the larger value produce slightly worse results. This is reasonable as an increased η results in more approximated clusters. Further, the different mesh sizes show a similar behavior. On the other hand the different admissibility conditions show for $\eta = 1$ a better approximation quality for the maximum based criterion. However, for $\eta = 5$ this effect is reduced. Nevertheless, it must be remarked that the minimum criterion produce less cluster and, as shown later, results in a faster matrix-vector product.

Next, the influence of the parameter $dist_m$ in the condition (16) is studied. In Fig.10, we present the $RMS_{[B]}$ depending on the distance $dist_m$ for different μ and different meshes. Note, now we have the Yukawa type kernel and criterium (16) decides whether the kernel is interpolated or not. For this test these dense blocks are not compressed with ACA and, hence, are not approximated. Decreasing the distance $dist_m$ increases the number of admissible block clusters where the kernel interpolation is applied. Thus, more elements in the matrix are approximated and, consequently, the accuracy of $[B]$ decreases. Further, for higher values of μ this effect is stronger as a lot of entries are set to zero following (17) and the overall amount of block clusters with kernel interpolation is smaller. Finally, it may be remarked that the influence of the interpolation order is not affected by this additional condition.

Next, the influence of μ and different mesh sizes is studied. In Fig.11, we present the $RMS_{[B]}$ depending on the mesh density and the shape of the fundamental solution. When we increase μ,

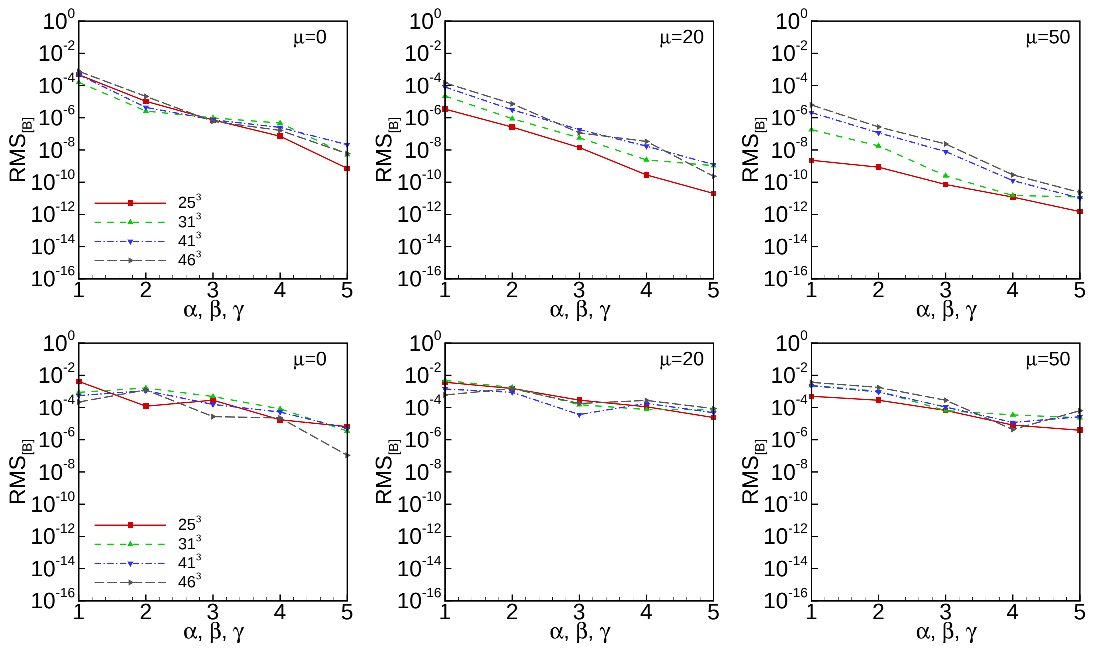

Figure 8: The norm $RMS_{[B]}$ versus interpolation order for different meshes and values of parameter μ. No recompression is applied. In the top panels the admissibility condition (14) is used and in the bottom panels (15) both with $\eta = 1$.

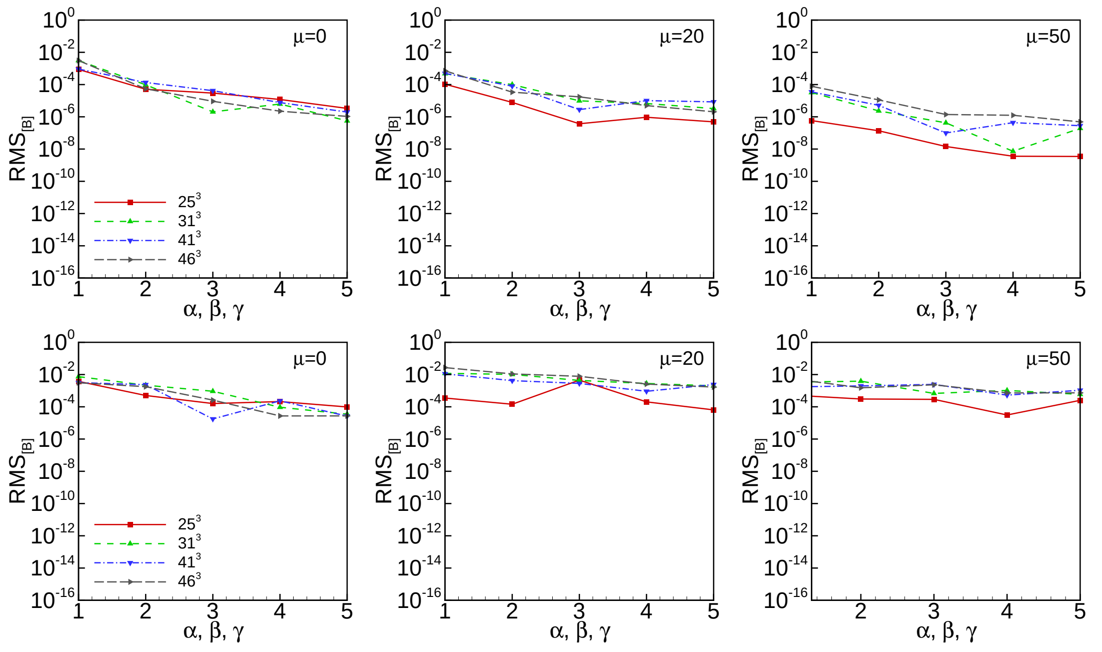

Figure 9: The norm $RMS_{[B]}$ versus interpolation order for different meshes and values of parameter μ. No recompression is applied. In the top panels the admissibility condition (14) is used and in the bottom panels (15) both with $\eta = 5$.

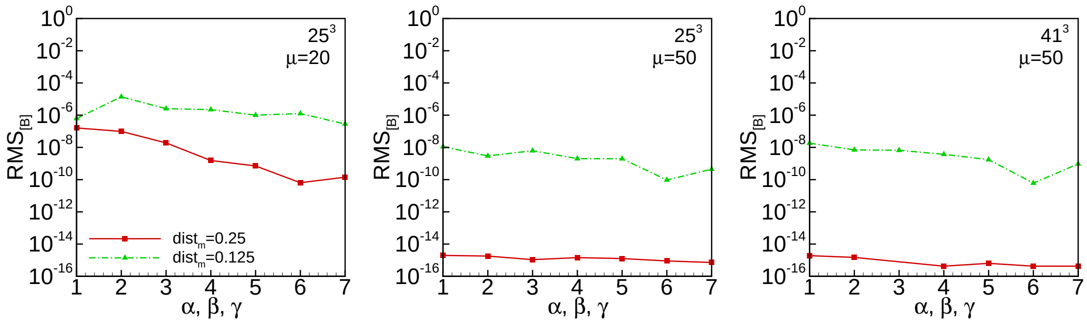

Figure 10: The $RMS_{[B]}$ for $dist_m = 0.125$ and $dist_m = 0.25$. The value of parameter μ was 20 and 50 and results for the meshes with 25^3 and 41^3 nodes are displayed. Admissibility condition (15) is used with $\eta = 5$. No recompression is applied.

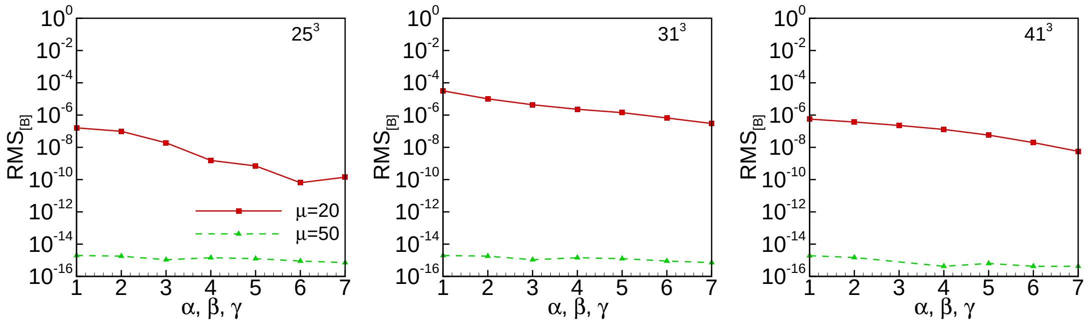

Figure 11: The $RMS_{[B]}$ for two values of parameter μ. The mesh densities are 25^3 (left), 31^3 (middle) and 41^3 (right). No recompression is used. Admissibility condition (15) is used with $\eta = 5$.

the global character of the Yukawa fundamental solution changes to local, i.e., the values of the fundamental solution decrease away from the collocation point to a low order polynomial and, hence, can be better approximated by the chosen Lagrange interpolation. This effect is similar for all presented meshes. Increasing the interpolation order decreases the error. However, for high values of μ the influence of the interpolation order is nearly not visible because the error is too small to see any influence.

A similar study but differently displayed can be found in Fig.12. There, again, the $RMS_{[B]}$ is shown for different meshes and different values of μ but now the error for all meshes is shown in one graph. The number of interpolation points $\alpha = \beta = \gamma$ were increased from 1 to 7. The condition (16) was enabled and $dist_m$ was set to 0.25. Four different mesh densities were employed. We observe that the accuracy depends slightly on the mesh density, whereas an increase of the interpolation order decreases the error as expected. The error level is different for different values of μ because an increase of μ allows to discard matrix blocks following (17). On the left and middle panel in Fig.12 the results for interpolation order $\alpha = \beta = \gamma = 7$ are missing. Computing $RMS_{[B]}$ requires to compute the dense matrix as well and, hence, the meshes 41^3 and 46^3 last too long.

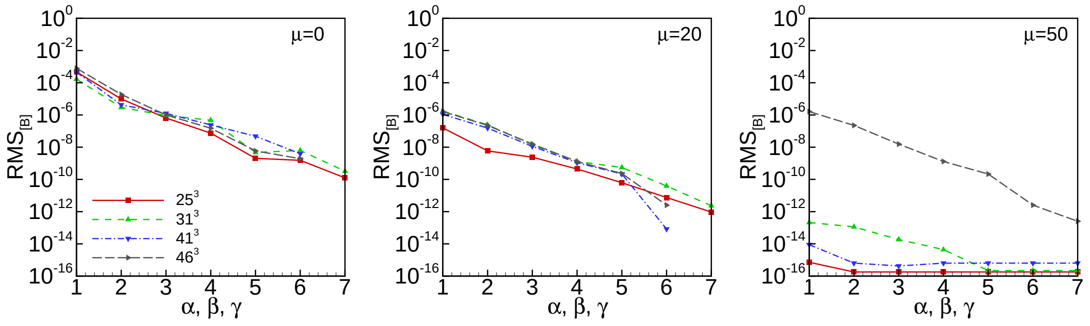

Figure 12: The norm $RMS_{[B]}$ versus interpolation order for different meshes and parameters μ. No recompression is applied and the admissibility condition was (15) with $\eta = 1$. The condition (16) was enabled with $dist_m = 0.25$.

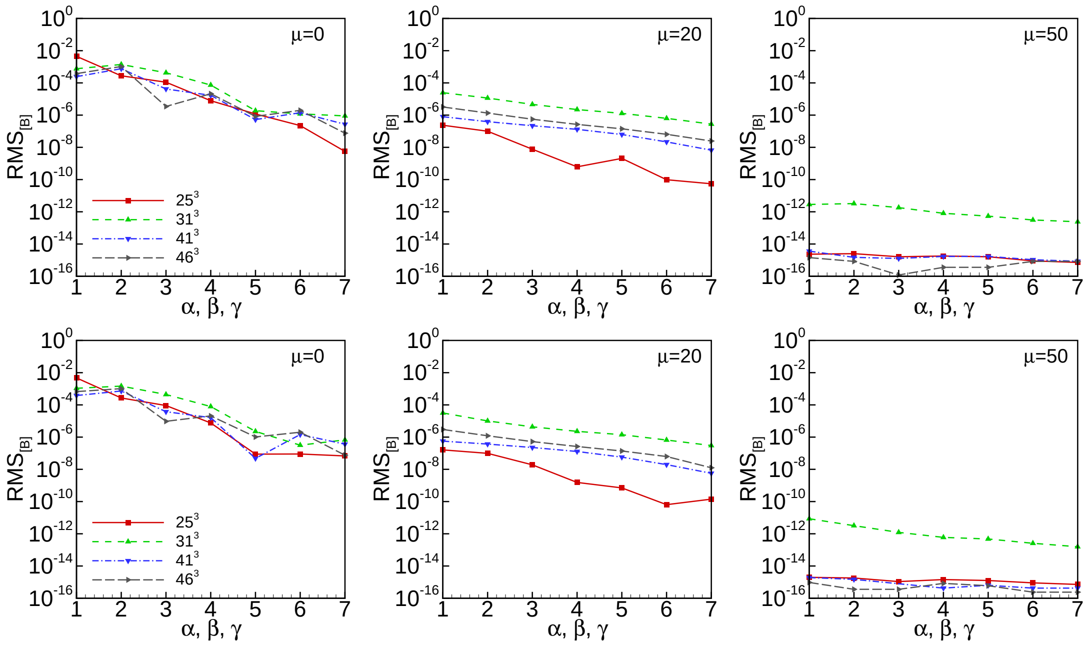

Figure 13: The norm $RMS_{[B]}$ versus interpolation order for different meshes and parameters μ without recompression. In the top panels the admissibility condition (14) is used and in the bottom panels (15) with $\eta = 5$. The condition (16) was enabled with $dist_m = 0.25$.

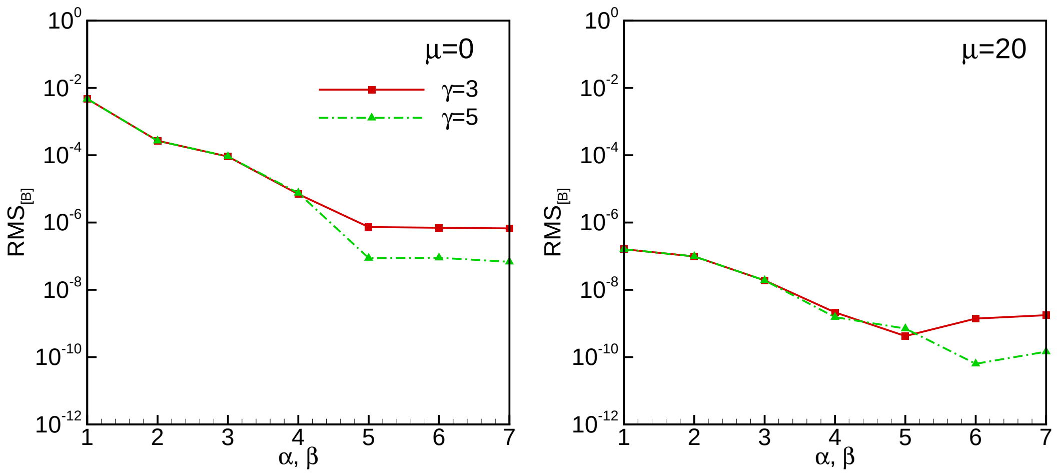

Figure 14: The norm $RMS_{[B]}$ for $\gamma = 3$ or $\gamma = 5$ and different μ without recompression. The used mesh has 25^3 nodes and $\eta = 5$ is chosen.

To show the influence of the two admissibility conditions (14) and (15), in Fig.13 we compare the $RMS_{[B]}$ again for both conditions but now, different to the study above, including the condition (16) with $dist_m = 0.25$. We observe no dependence of the approximation accuracy on the admissibility condition. The $RMS_{[B]}$ is for all cases similar. Compared to the study in Fig.9 without condition (16) the accuracy is similar or even better for $\mu = 50$. Hence, this additional condition (16) pays off.

In the previous subsection the influence of the interpolation order γ was studied on the artificial polynomial function (see Fig.7). The same effect should be visible for the fundamental solution. In Fig.14, we present the $RMS_{[B]}$ versus the interpolation order γ. As expected, the error decreases with increasing interpolation order $\alpha = \beta$ up to the point $\alpha = \beta = \gamma$. An further increase of the interpolation order does not decrease the error because it is dominated by the error due to the nested cluster basis (23), which is only introduced for $\gamma < \alpha = \beta$. The effect is the same for different values of μ as this parameter does not change this dominance of the interpolation error. However, overall the error is smaller for larger μ due to the above discussed local behavior.

4.3 Results with recompression

For the last test, the ACA algorithm was employed to compress the matrix $[\hat{S}]$ from (22) in admissible block clusters, which is denoted recompression. In Fig.15, we present the $RMS_{[B]}$ to measure the influence of this recompression. Please note that the admissibility condition (15) and condition (16) with $dist_m = 0.25$ are used. The interpolation of the kernel is done in all three coordinate directions, hence, the size of $[\hat{S}]$ is $\alpha^3 \times \beta^3$ with the number of interpolation points α and β. The user-defined parameter ε determines the compression of the matrix with the ACA and is varied on the horizontal axis in Fig.15 from 10^{-4} to 10^{-10}. Obviously and expected, a lower ε results in a better approximation of $[B]$. In the left panel it can be further observed that a point can be reached, where the interpolation error dominates and the ACA does not increase

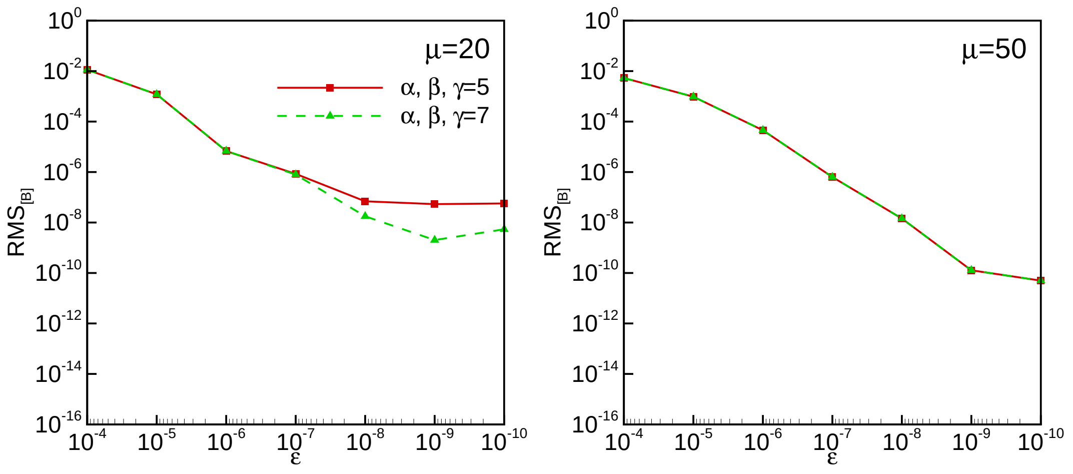

Figure 15: The $RMS_{[B]}$ versus ε of the ACA stopping condition for $\mu = 20$ (left) and $\mu = 50$ (right). The mesh with 46^3 nodes and $\eta = 5$ was chosen.

the overall error. For $\mu = 50$ in all tests the ACA based error dominates because the kernel can easily be approximated by a low order interpolation.

The recompression is introduced to decrease the storage of the $\mathscr{H}^2$-matrix. To see the influence of the ACA the storage of $[B]$ is plotted versus ε in Fig.16. For higher values of ε the storage is substantially smaller. The effect is larger for larger matrices comparing the left and right plot in Fig.16. This is not astonishing because larger matrices have more admissible blocks, where the recompression with ACA can be applied. As discussed above, the price for less storage is less accuracy.

After these tests for different approximations, next, the overall complexity is studied numerically. It can be expected that a linear complexity is obtained. In Fig.17, we present the memory usage for an increasing number of unknowns m. The memory is presented for computations with $\mu = 20$ and $\mu = 50$ and for both admissibility conditions. Additionally, complexity curves are displayed for $\mathscr{O}(nm)$, $\mathscr{O}(m \log m)$, and $\mathscr{O}(m)$. It seems that the proposed method reduces the complexity from $\mathscr{O}(nm)$ for a dense version to $\mathscr{O}(m)$, i.e. the expected linear behavior. When μ is growing the memory decreases, as more matrix blocks are neglected following condition (17). The admissibility condition (15) and (14) have a minor influence on the storage, which is for both conditions $\mathscr{O}(m)$.

Beside storage the CPU-time usage is an important criterium. In Fig. 5 it was shown that the min-based condition produces less block clusters, which should improve the speed in the matrix-vector product. This is studied in Fig 18 where the CPU-time for 100 matrix-vector products $[B]\{b\}$ is plotted versus the number of unknowns. It can be observed that the matrix-vector product is faster, when condition (15) is employed. The parameter μ has no influence on the CPU-time. This faster matrix-vector product might justify the slightly worse approximation presented in Fig.8. However it must be remarked that condition (15) has no mathematical basis.

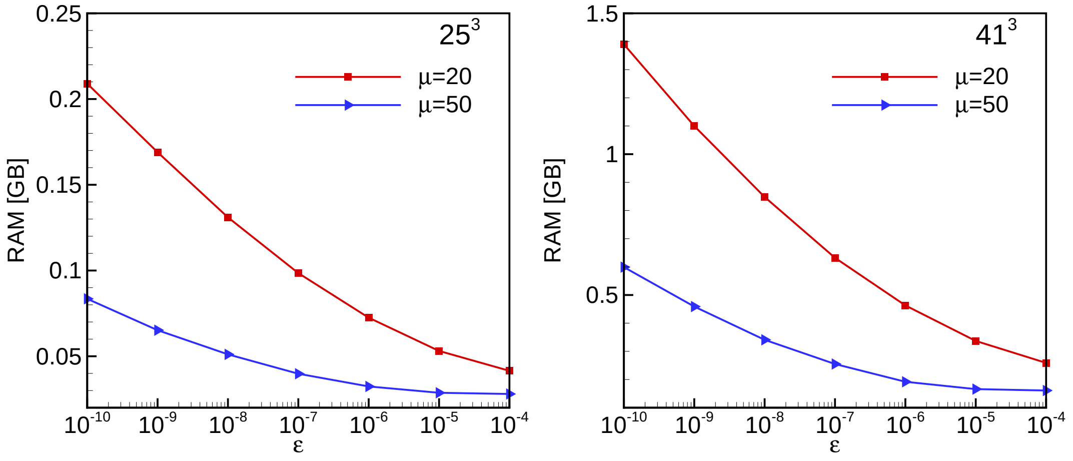

Figure 16: The influence of ACA on the storage of matrix $[B]$ for two meshes and values of μ. The interpolation order was $\alpha = \beta = \gamma = 3$. The admissibility conditions (14) and condition (16) with $dist_m = 0.25$ are used.

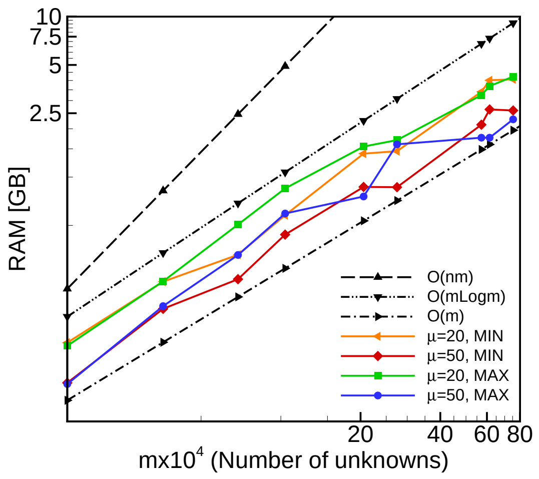

Figure 17: The amount of memory needed to store the matrix [B]. The interpolation order is $\alpha = \beta = \gamma = 3$ and the stopping condition was set to $\varepsilon = 10^{-8}$. MIN indicates the admissibility condition (15) and MAX is (14).

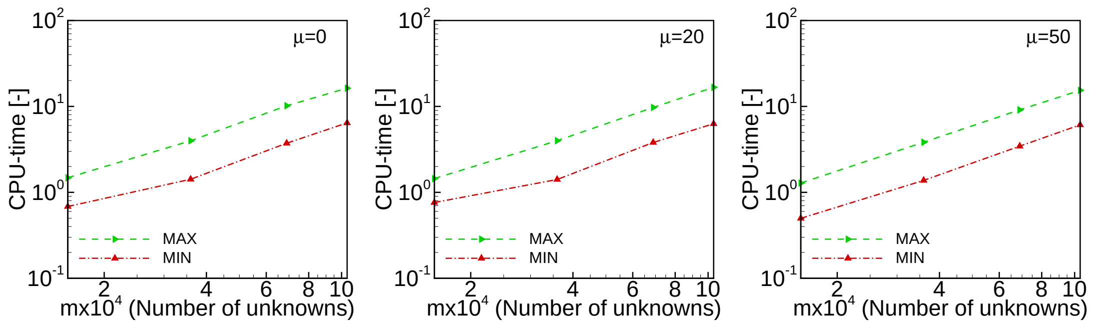

Figure 18: CPU-time of 100 matrix-vector products $[B]\{b\}$ for four meshes and three values of μ. MIN indicates the admissibility condition (15) and MAX is (14). The interpolation order was $\alpha = \beta = \gamma = 3$, the ACA stopping condition $\varepsilon = 10^{-8}$ and $\eta = 5$.

5 Conclusions

The Yukawa equation with an inhomogeneous right hand side has been considered. Beside other applications, an exemplarily physical application might be the incompressible, laminar flow of a Newtonian fluid handled with the false transient approach. The Poisson equation is a special case and can be handled as well with the proposed approach. The corresponding boundary domain integral equation (BDIM) is classically discretised but the $\mathcal{H}^2$-methodology is applied to reduce the complexity. The main focus was on the application of $\mathcal{H}^2$-matrices to represent the matrix from the domain integral. Two cluster trees are necessary where one is living on the boundary and the other in the domain. Further, the block clusters have been combined not necessarily within one level. Recompression was done with a fully pivoted adaptive cross approximation (ACA).

Tests revealed that the developed algorithm reduces the computational cost of the BDIM from $\mathcal{O}(nm)$ to $\mathcal{O}(m)$, where m is the number of unknowns in the domain and n on the boundary. The accuracy of the approximation depends on the interpolation order of the kernel expansion and on the recompression with ACA. A naive implementation may cause for medium sized problems an higher effort compared to a classical BDIM without fast methods. Here, additionally to the admissibility criterion of the $\mathcal{H}^2$-matrices, block clusters with very small entries compared to other blocks, i.e., with a very small Frobenius norm, have been discarded. Such situations are caused by the strong decrease of the fundamental solution of the Yukawa operator. As usual in all fast BE methods, one should take care that the error introduced by adding additional approximation techniques within the $\mathcal{H}^2$-method is of the same order of magnitude as the discretization error and the error due to the solver of the system of linear equations.

Acknowledgments

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0196). The authors would like to acknowledge the use of HPC resources provided by the ZID of Graz Univresity of Technology.

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] Verhnjak, O. and Hriberšek, M. and Steinmann, P. and Ravnik, J.,A novel two-way coupling model for Euler-Lagrange simulations of multiphase flow, Engineering Analysis with Boundary Elements 119 (2020) 119–132.

[4] 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.

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

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

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

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

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

[22] 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.

[23] The Fast Solution of Boundary Integral Equations, Springer-Verlag 2007.

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

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

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

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

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

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

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

[35] Ravnik, Jure and Tibaut, Jan,Fast boundary-domain integral method for unsteady convection-diffusion equation with variable diffusivity using the modified Helmholtz fundamental solution, Numerical Algorithms, 2019.

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

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

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

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

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

[41] Börm, Steffen and Lopez-Fernandez, Maria and Sauter, Stefan A,Variable order, directional $\mathscr{H}^2$-matrices for Helmholtz problems with complex frequency, IMA Journal of Numerical Analysis 12 (2020) 0272-4979.

[42] , Steffen Börm,Efficient Numerical Methods for Non-local Operators, EMS Tracts in Mathematics 14(2) (2013).