Monographic Series TU Graz

Computation in Engineering and Science

Matthias Messner

Fast Boundary Element Methods in Acoustics

Graz, 5. Dezember 2011

Abstract

Wave propagation problems play an important role, not only in the scientific community, but also in industry. Think of acoustic scattering, sound radiation and other related problems. A common feature of these problems is the large or even infinite extension of the acoustic domain compared to the rather limited extension of the surface of the embedded scatterer. The appropriateness of solving such problems is inherent to the boundary element method: The solution in the domain (large or even infinite extension) is available, once the solution on the boundary (limited extension) is computed. But, standard boundary element formulations lead to fully populated system matrices and the computation of their entries is very expensive. This issue has been remedied by the introduction of fast boundary element formulations, the topic of the thesis at hand. Our objective is twofold. In the first place, we construct two efficient numerical schemes, the $\mathcal{H}$-matrix and the directional fast multipole scheme. Both enable an efficient treatment of fully populated matrices of oscillatory nature. In the second place, we apply these schemes to acoustic boundary element formulations and reduce their quadratic complexity to an almost linear one. We validate our approaches by means of numerical examples. We solve a time-domain problem by means of $\mathcal{H}$-matrices and several time-harmonic problems by means of the directional fast multipole method. We emphasize on the fact that both presented approaches are kernel independent, up to a certain extend.

Zusammenfassung

Die Simulation von Wellenausbreitungsphänomenen ist nicht nur für die Wissenschaft, sondern auch für die Industrie von großem Interesse: Man denke zum Beispiel an akustische Streuprobleme oder die Schallabstrahlung von bewegten Körpern. Bei solchen Problemen hat der akustische Außenraum per Definition eine unendliche und darin vorkommende Körper eine endliche Ausdehnung. Im Vergleich zu anderen numerischen Methoden, hat die Randelementemethode den Vorteil, dass die Lösung im akustische Aussenraum nicht diskretisiert werden muss, sondern durch die Lösung auf der Oberfläche dieser Körper exakt dargestellt werden kann. Ein Nachteil jedoch sind ihre vollbesetzten Systemmatrizen. Erst die Konstruktion schneller Randelementeformulierungen, womit sich die Arbeit beschäftigt, macht die Methode wieder attraktiv. Es werden zwei unterschiedliche Herangehensweisen zur effizienten Behandlung vollbesetzter Matrizen mit oszillierenden Eigenschaften präsentiert: $\mathcal{H}$-Matrizen und eine richtungsabhängige Fast Multipol Methode. Beide Ansätze ermöglichen es, die quadratische Komplexität der Randelementemethode auf eine fast lineare Komplexität zu senken, was anhand numerischer Beispiele verifiziert werden konnte. In einem Anwendungsbeispiel wurde die Schallabstrahlung eines Motors simuliert. Weitere Beispiele zeigen die allgemeine Gültigkeit der Formulierungen auf.

CONTENTS

1 INTRODUCTION

The demand of simulating large real-world problems arising in science and engineering has come along with the booming development of computing hardware. The growing interest in conducting simulations on computers has multiple reasons. What are they? Computers represent a highly flexible and efficient testing environment. Their fundamental merit is the destruction-free nature when doing simulations. For example, only a limited number of actual test cases have to be setup in a laboratory in order to calibrate and verify a simulation software. Then, all further tests can be performed on computers which leads to a significant cost reduction. These facts motivate the research in the field of computer aided simulations.

The development of numerical schemes and their implementation on computers require mathematical models which describe the physical problem we have to solve. Such models often result in summations like

$$f_i = \sum_{j=1}^{N} K(x_i, y_j)\, w_j \quad \text{for } i = 1, \ldots, N.$$

The complexity of computing this summation directly grows quadratically with the problem size N. Let us assume, the computing capabilities grow linearly. Then, the limiting factor of computer aided simulations will always be the quadratic growth of its computational complexity, unless, we develop efficient algorithms which reduce it.

In the thesis at hand, we focus on such algorithms, more specifically, on those which are well suited for problems which are of oscillatory nature. We think of physical problems where the kernel function in the above summation can be modeled like

$$K(x,y) = \frac{1}{4\pi} \frac{\mathrm{e}^{\imath k|x-y|}}{|x-y|},$$

where k is the measure for the oscillatory behavior and is often denoted as wavenumber. For example, we consider acoustic problems here. They often consist of a solid body being a scatterer or an exciter and the acoustic pressure in the surrounding acoustic medium is of interest. A powerful numerical scheme which is often adopted for the simulation of such problems is the boundary element method (BEM). It requires a boundary description only. In our specific case, the boundary of the acoustic domain is the surface of the solid body. Once, the boundary solution is available, the solution within the acoustic medium is available, too. The bottleneck of the BEM is the presence of dense system matrices, which yields the quadratic complexity we addressed above.

To close the circle, the main objective of this thesis are efficient algorithms which can be applied to speedup the simulation of acoustic problems by using boundary element methods.

1.1 State of the art

Let us give an overview on the current state of research. We arrange the considered topics in two paragraphs. The first one addresses the research in the field of efficient summation schemes and the second one in the field of boundary element methods and their efficient realization.

Efficient summation schemes Numerous approaches have been developed in the last decades. All have the common goal to reduce the quadratic complexity of direct summations which arise when pair-wise interactions need to be evaluated. To our knowledge, the first publications in this field are Appel [3] and Barnes and Hut [8]. The paper by Rokhlin [84] is probably the most known one. They presented for the first time algorithms which scale like $\mathcal{O}(N \log N)$. Based on the latter work, the most prominent representative, the fast multipole method (FMM) has been developed in [50, 31, 27, 78]. The method has been improved significantly by Greengard and Rokhlin [51]. They proposed an approach which scales like $\mathcal{O}(N)$ by exploiting the multilevel concept.

Another prominent representative are hierarchical matrices, also known as $\mathcal{H}$-matrices (see Hackbusch [56]). They are mostly used together with the adaptive cross approximation (ACA, see Bebendorf [9, 12]) and lead to $\mathcal{O}(N \log N)$ methods. $\mathcal{H}$-matrices can be understood as matrices in the usual sense with the difference that they have a more efficient structure. The entire framework of matrix arithmetics has been developed in [21, 58, 55] and parallelized in [14]. A second hierarchy level has been introduced in [57, 19], it leads to the so called $\mathcal{H}^2$-matrices. This approach represents the algebraic counterpart to multilevel FMMs and lead to $\mathcal{O}(N)$ methods, as well.

Further approaches are the panel clustering method (see [59]) and wavelet techniques (see [1]). The former approach is considered as the analytic counterpart of $\mathcal{H}$-matrices and has strong similarities to a single-level FMM. The latter approach yields very good compression rates for the approximation of integral operators. It produces sparse matrices based on orthogonal systems of wavelet like functions. However, their construction depends on the geometry of the problem and that decisive limits the approach. If source and target particles are ordered ideally on a regular grid the resulting influence matrix exhibits a Toeplitz structure. Using fast Fourier transforms [68] the complexity of the resulting matrix-vector product can be reduced to $\mathcal{O}(N \log N)$. However, the drawback of such approaches is that

physically interesting effects of randomly distributed particles can only be modeled with further effort.

All mentioned approaches have a fundamental concept in common. They expect kernel functions to be separable. Proper admissibility criteria provide a priori knowledge when this applies. The most common one is based on the distance of the particle pair $|x-y|$ and typically applies to kernels which are asymptotically smooth with respect to that distance (see, e.g., [56, 12]). Another admissibility criterion, which applies for kernels of retarded boundary integral operators, has been introduced in [64].

In the thesis at hand, we focus on oscillatory kernel functions. Additionally to the distance $|x-y|$, they are affected by the so called wavenumber k. The rank of the oscillatory part of the kernel $\mathrm{e}^{\imath k|x-y|}$ grows like $\mathcal{O}(kW)$, where W is the size of the domain of the particles x and y, respectively. This is explained in [88] and also shown in the thesis at hand. Several solutions have been proposed to address this issue. In [85], a high-frequency FMM based on diagonal translation operators has been developed for the Helmholtz kernel. Diagonal operators have also been worked out in [52] for both the low- and high-frequency regime. A stable plane wave expansion for the whole frequency regime has been proposed in [36]. A combined wide-band scheme that switches between different representations in order to cover both the high- and low-frequency regime has been presented in [32]. Brandt has taken a different approach in [23]. He exploited the fact that the oscillatory part of the kernel $\mathrm{e}^{\imath k(|x-y|-u\cdot(x-y))}$ is low-rank in the direction of the unit vector u and independent of the wave number k. Along the same line, Engquist and Ying have proposed the directional admissibility criterion for oscillatory kernels in [40, 41]. Relying on that idea they construct a fast directional multilevel algorithm for oscillatory kernels. It is also the approach we follow, here.

Fast boundary element methods The boundary element method (BEM) is a well established numerical scheme. As the name already proclaims, only the boundary, i.e., the surface of the computational domain is considered. This seems to reduce the complexity of a, say, 3D problem to a 2D problem. This is true in terms of number of unknowns. However, the arising system matrices are fully populated and the entries are costly to compute as opposed to the finite element method (FEM). We refer to the textbooks [91, 83, 43] as our main references. The first one gives a good engineering introduction to the BEM. The second one is a practical guide to $\mathcal{H}$-matrices, ACA and their application to the BEM. The third textbook provides a rigorous mathematical framework of the method.

What type of problems are well-suited to be solved with the BEM? Characteristic for this method is that the solution in the domain is fully given by the solution on the boundary. This is in favor of exterior problems where the domain of interest is large or even unlimited compared to the boundary. They often arise in wave propagation simulations. A detailed review on time-domain BEM can be found in the articles [17, 18, 38] and [34].

We consider the convolution quadrature method (CQM) proposed by Lubich [69, 70] to solve time-domain problems. It provides a time-stepping scheme and uses the Laplace domain fundamental solution which is essential in certain cases (see [87, 86]). A reformulated CQM has been published by Banjai and Sauter [6], they rewrite the time-domain problem as a system of decoupled Laplace domain problems. It allows the application of well known efficient summation schemes. We adopt this approach for acoustic problems in time-domain. An improved version using Runge-Kutta methods can be found in [4] and is studied in [7] and in this thesis.

In order to pave the way for the simulation of large-scale problems fast boundary element methods have been introduced. There exist several textbooks addressing them [12, 19, 56]. The first and the last one treat $\mathcal{H}$ and the second one $\mathcal{H}^2$-matrices. Two important textbooks about FMM BE-formulations are [65, 53]. In the last two decades, numerous application of fast boundary element methods have been published. In the following, we list the most important representatives. The first applications of ACA to speedup BE-formulations have been published in [9] and [15]. In [10], the computation of good preconditioners based on $\mathcal{H}$-inversion or $\mathcal{H}$-LU decomposition is introduced. An engineering approach is presented in [16]. Besides using ACA with $\mathcal{H}$-matrices, also an interpolation based approximation can be used [20]. A combined $\mathcal{H}/\mathcal{H}^2$-matrix approach for low- and high-frequency scattering is given in [5]. A wavelet based fast BEM approach is presented in [24]. In the work of Of et al. [81], the FMM is applied to elastostatic problems based on a Galerkin BEM discretization. Engineering applications are presented in [73, 67] . The extension to elastodynamics has been published in [29] based on a collocation approach. In time domain, the FMM with a plane wave expansion applied to the BEM is presented in [92]. In [26], a comparison of fast approaches is given.

1.2 Model example

Generally speaking, any matrix $\mathsf{K} \in \mathbb{C}^{M\times N}$ can be associated to a generating function $K(x,y)$, also called kernel function. The entries of that matrix are computed as

$$(\mathsf{K})_{ij} = K(x_i, y_j) \quad \text{with } i = 1,\dots,M \text{ and } j = 1,\dots,N.$$

There exist various types of kernel functions. In the work at hand, we treat only so called *non-local* ones which generate *fully populated matrices*. Non-local functions describe the direct influence of the values at the set of source points $\{y_j\}_{j=1}^N$ on the values at the set of field points $\{x_i\}_{i=1}^M$ which might be separated in space.

Model problem We use the kernel function

$$K(x,y) = \frac{1}{\alpha + |x-y|}, \quad x,y \in \mathbb{R}^d \text{ and } \alpha \in \mathbb{R}^+, \tag{1.1}$$

where the spacial dimension is denoted by d. Without the parameter α the kernel function would become singular as y approaches x. We use this parameter in order to avoid this issue, it removes the "singularity". The goal is to evaluate the influence of the values $\{w_j\}_{j=1}^N$ at the source points on the values $\{f_i\}_{i=1}^M$ at the target points. This can be done by means of the direct summation

$$f_i = \sum_{j=1}^{N} K(x_i, y_j) w_j \quad i = 1, \ldots, M, \tag{1.2}$$

which requires $\mathcal{O}(MN)$ operations. This becomes inacceptable as N and M grow. Our goal is to construct a fast summation scheme which requires less operations. Usually, such fast methods are approximations of a direct method. Hence, the cost of the speedup is the accuracy of the result.

Low-rank matrices Let us write the direct summation from (1.2) in matrix notation as $\mathsf{f} = \mathsf{K}\mathsf{w}$. In order to reduce its cost, we need to approximate K by a low-rank matrix K_r of rank $r \leq \min\{M,N\}$. Crucial is that we must be able to control the accuracy of the result. In order to achive that, we introduce the following bound

$$\|\mathsf{K} - \mathsf{K}_r\| \leq \varepsilon \|\mathsf{K}\|, \tag{1.3}$$

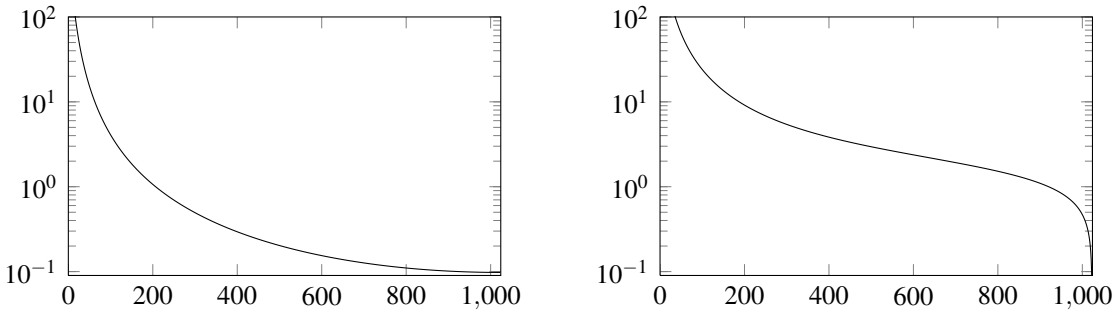
where $\|\cdot\|$ denotes some matrix norm. The matrix K can be approximated by another matrix K_r of lower rank $r \leq N$ for example by means of a truncated singular value decomposition

$$\mathsf{K} \sim \mathsf{K}_r = \mathsf{U}\Sigma\mathsf{V}^* \quad \mathsf{U},\mathsf{V} \in \mathbb{C}^{N\times r},\ \Sigma \in \mathbb{R}^{r\times r}, \tag{1.4}$$

where Σ stores the singluar values on its diagonal and U and V store the left and right eigenvectors, respectively. $()^*$ denotes the conjugate transpose and $\sim$ means that the term on its right is the approximation of the term on its left. At this point of the work we do not go deeper into details of this approximation strategy (for more details we refer to sec. 2), we just point out that the quality of the approximation depends on how fast the singular values of the matrix decay. In fact, the number of non-zero singular values represents the rank of a matrix.

Let us set $\alpha = 10^{-1}$ in the kernel function (1.1) and distribute the source and target points equally in the intervall $[-1,1]$ as

$$x_i = y_i = -1 + 2\frac{i-1}{N-1}, \quad i = 1, \ldots, N,$$



(a) Decay of singular values

(b) Error decay ε_r for growing rank r

Figure 1.1: Behavior of entire matrix $[-1,1] \times [-1,1]$ with $N = 1000$

then fig. 1.1a shows the decay of the singular values of the matrix K. There are no vanishing singular values, i.e., all are non-zero, hence K as a whole has full rank. With the error bound (1.3) we can define the accuracy of the approximation in terms of the rank r

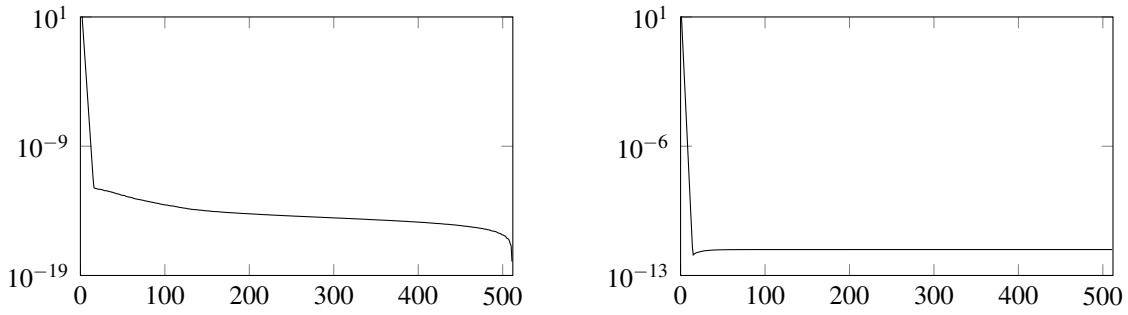
$$\varepsilon_r = \frac{\|\mathsf{K} - \mathsf{K}_r\|_F}{\|\mathsf{K}\|_F}.$$

In fig. 1.1b, we plot this accuracy ε_r in terms of the rank $r = 1, \ldots, N$. In practice, accuracies of 10^{-4} and higher are required. We can conclude from this plot that it is not possible to obtain a low-rank approximation K_r for the kernel function (1.1) in the domain $[-1,1] \times [-1,1]$ which satisfies an accuracy $\varepsilon \leq 10^{-4}$. To rephrase this, we are not able to compute a *low-rank* approximation of the matrix K in its entirity. However, we can compute a *data-sparse* approximation of K, we will see that in the following.

Data-sparse approximation The problem is that matrices with kernels of the type $1/|x-y|$ have full rank. The reason is that their generating kernel becomes singular as $|x-y|$ approaches 0. As we have seen above, the goal can not be to find a low-rank approximation of the entire matrix K. Instead we are looking for a data-sparse approximation. The idea is to find submatrices of K which are effectively low-rank. To do so, we partition K in four submatrices: We get two diagonal blocks $[-1,0] \times [-1,0]$ and $[0,1] \times [0,1]$ and two off-diagonal blocks $[-1,0] \times [0,1]$ and $[0,1] \times [-1,0]$.

Let us first treat the latter two blocks, there target and source points are separated. In fig. 1.2 we show the decay of the singular values and the accuracy ε_r of these off-diagonal blocks. Figure 1.2b shows that already a very small rank r approximates the matrix block accurately. Indeed a rank $r = 7$ leads to an accuracy of about 10^{-4}.

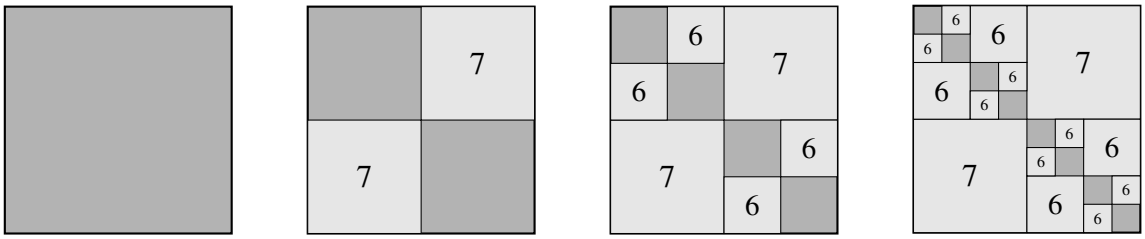
Next, we look at the diagonal blocks. The target and source points are not separated, they have a "singularity" located on their diagonal, i.e., they have the same structure as K as



(a) Decay of singular values (b) Error decay ε_r

Figure 1.2: Behavior of off diagonal blocks, eg. $[-1,0] \times [0,1]$

a whole. Thus, it seems natural to partition these diagonal blocks in the same way as we partitioned the K beforehand. This procedure can be repeated recursively. We add the off-diagonal blocks having a low-rank approximation to the *far-field* and subdivide the diagonal blocks until we reach a given recursion depth. There, the remaining diagonal blocks form the *near-field*. The result is shown if fig. 1.3, a partitioned matrix with separated near- and far-field. In our model example we stop at the third level. The numbers in the far-field blocks represent the low-rank which leads to an accuracy of 10^{-4}. The near-field blocks have full-rank. A general note to the the recursion depth of the partitioning: The aim is to somehow balance the cost of evaluating near-field and far-field.



(a) Entire matrix (b) 1 level (c) 2 levels (d) 3 levels

Figure 1.3: Separation of near- and far-field; the numbers denote the low-rank r which guarantees the accuracy $\varepsilon = 10^{-4}$

Fast matrix-vector product We can construct a fast matrix-vector product by exploiting the low-rank approximations of far-field blocks. The main difference to direct matrix-vector products is that we need to sum up the contribution of all blocks forming K. The multiplication of near-field blocks is performed in the usual way, the multiplication of far-

field blocks is performed efficiently: We use the low-rank representation (1.4) and get

$$\mathsf{f} \sim \mathsf{U}\Sigma(\mathsf{V}^*\mathsf{w}) = \mathsf{U}(\Sigma\bar{\mathsf{w}}) = \mathsf{U}\bar{\mathsf{f}}, \tag{1.5}$$

where $\bar{\mathsf{w}} = \mathsf{V}^*\mathsf{w}$ and $\bar{\mathsf{f}} = \Sigma\bar{\mathsf{w}}$ are intermediate vectors. By exploiting (1.5) the cost can be reduced from $\mathcal{O}(MN)$ to only $\mathcal{O}(r(M+N))$ operations for any low-rank block.

The cost of computing and storing matrix entries and the cost of performing a matrix-vector multiplication is equivalent. Hence, for the sake of simplicity we just talk about cost here. Table 1.1 shows the overall cost and the amound of cost reduction of the current example in terms of different levels of matrix partitioning.

# levels	cost	reduction
0	$1000N$	1.00
1	$514N$	0.51
2	$276N$	0.27
3	$163N$	0.16

Table 1.1: Cost reduction depending on the number of levels as shown in fig. 1.3.

Recall, the entire matrix K has full-rank, i.e., no accurate low-rank approximation exists. Although by separating near- and far-field a *data-sparse* approximation can be found. Note, this is not the same as a low-rank approximation, it is a more general concept. All fast schemes to perform matrix-vector products efficiently have the following three points in common

- Separation of near- and far-field blocks
- Direct evaluation of near-field blocks
- Approximation of far-field blocks

Finally, by exploiting the resulting data-sparse approximation various fast schemes can be constructed which reduce the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\log N)$ or even $\mathcal{O}(N)$.

1.3 Outline

Our objective can be seen as twofold. On one hand, we investigate on tools to efficiently treat pairwise interactions of oscillatory nature. On the other hand, these tools can be utilized to increase the efficiency of various numerical methods, e.g., boundary element formulations. The structure of the thesis at hand reflects this splitting.

From the model example in the previous section we have learnded about some of the major concepts of efficiently treating dense matrices. We will deepen these concepts in the first part of the thesis. In chap. 2, we introduce the concept of oscillatory kernels which arise in the acoustic problems we treat. Moreover, we investigate on their properties and present several approaches to efficiently represent associated matrices. In chap. 3, we construct two schemes which allow for an efficient treatment of dense matrices arising from oscillatory kernel functions. Thereby, we always focus on the kernel independent nature of both methods. We conclude this chapter with numerical benchmarks.

In the second part of the thesis, we solve large acoustic problems with the aid of the tools we constructed in the first part of the thesis. In chap. 4, we briefly present the governing equations for the acoustic fluid. Then, the boundary integral representations are given, which provide the basis for the numerical treatment of acoustic problems. Finally, in chap. 5 we present various numerical simulations and show the efficiency of the presented methods.

We conclude the thesis in chap. 6 with a summary and an outlook of future work in this field and potential uses of the presented methods.

2 LOW-RANK MATRICES

In this chapter, we introduce a basic tool of fast methods: the approximation by low-rank matrices. Such low-rank approximants can be exploited to reduce storage requirement and to speed up arithmetic operations, such as multiplications and even factorizations.

Matrix kernels

Let the matrix $\mathsf{K} \in \mathbb{C}^{M\times N}$ be associated to the generating matrix kernel function $K(x,y)$ with $x \in X \subset \mathbb{R}^d$ and $y \in Y \subset \mathbb{R}^d$. The entries of K are then computed as

$$(\mathsf{K})_{ij} = K(x_i, y_j)$$

for all $\{x_i : i = 1,\ldots,M\}$ and $\{y_j : j = 1,\ldots,N\}$. In the following, we treat only matrix kernel functions which lead to dense matrices having MN entries. We introduce here two kernel types, *asymptotically smooth* and *oscillatory* ones. Both are important throughout the work at hand.

Asymptotically smooth kernels Most kernel functions arising in physical problems are of the type $1/r$ with $r = |x-y|$. An example is the fundamental solution of the Laplace equation

$$K_L(x,y) = \frac{1}{4\pi}\frac{1}{|x-y|} \quad \text{for } x,y \in \mathbb{R}^3. \tag{2.1}$$

Such functions have unbounded derivatives, both in x and y, as y approaches x. On the other hand their smoothness increases indefinitely as the distance between x and y grows to infinity. Generally speaking, such kernels $K(x,y)$ are considered *asymptotically smooth* as a function of x and y if

$$\left|\frac{\partial^p K(x,y)}{\partial x^p}\right| \le C_p|x-y|^{g-p}, \quad \text{respectively} \quad \left|\frac{\partial^p K(x,y)}{\partial y^p}\right| \le \bar{C}_p|x-y|^{\bar{g}-p},$$

hold for all $x,y \in \mathbb{R}^3$ and $x \neq y$. Above $g,\bar{g} \in \mathbb{R}$ are independent on the order of the derivative p, whereas the constants C_p and $\bar{C}_p$ depend only on p. For more details we refer the reader to [23, 12].

Oscillatory kernels In acoustics and electromagnetic problems and in various other wave theories the arising matrices are associated to oscillatory kernel functions which incorporate the term $\mathrm{e}^{\imath kr}$. Here, $\imath = \sqrt{-1}$ is the immaginary unit, k is the wavenumber and $r = |x-y|$ the distance between x and y. An example is the fundamental solution of the Helmholtz equation

$$K_H(x,y) = \frac{1}{4\pi}\frac{\mathrm{e}^{\imath k|x-y|}}{|x-y|} \quad \text{for } x,y \in \mathbb{R}^3.$$

Note, the above kernel can be written as $K_H(x,y) = K_L(x,y)\,\mathrm{e}^{\imath k|x-y|}$. Hence, it exibits the same properties as $K_L(x,y)$ as the distance between x and y goes to infinity, i.e., it becomes indefinitely smooth. However, as we will see in sec. 2.3, the oscillatory nature due to $\mathrm{e}^{\imath k|x-y|}$ brings along complications considering their approximation. Whenever, we speak of oscillatory kernels in this thesis we think of the Helmholtz kernel K_H or kernels having the same properties.

Low-rank approximation

Any matrix K can be factorized if there are matrices $\mathsf{U} \in \mathbb{C}^{M\times\bar{r}}$ and $\mathsf{V} \in \mathbb{C}^{N\times\bar{r}}$ such that

$$\mathsf{K} = \mathsf{U}\mathsf{V}^*,$$

with $\bar{r} \leq \min\{M,N\}$. Note, this factorized representation is no approximation yet. Hence, it does not neccessarily reduce the cost of computing and storing the entries of K which is $\mathcal{O}(MN)$. In fact, the representation $\mathsf{U}\mathsf{V}^*$ can be less efficient than the original representation, it is $\mathcal{O}(\bar{r}(M+N))$. Note also that no error is introduced yet. Let us introduce a low-rank representation of K:

Definition 1. The matrix $\mathsf{K}_r \in \mathbb{C}^{M\times N}$ is a low-rank matrix if $r(M+N) < MN$ holds. It approximates $\mathsf{K} \in \mathbb{C}^{M\times N}$ satisfying the error bound

$$\|\mathsf{K} - \mathsf{K}_r\| \leq \varepsilon \,\|\mathsf{K}\|$$

for a given accuracy $\varepsilon > 0$ and some matrix norm $\|\cdot\|$.

Low-rank matrices K_r provide an efficient representation of dense matrices K. Beside reducing the cost of computing and storing matrix entries, low-rank matrices allow also to perform matrix-vector products efficiently.

There exist basically two sets of approximation methods to construct low-rank matrices. On one hand, *analytic methods* are based on a truncated kernel function expansion, i.e., an analytic approximation of the kernel function. On the other hand, *algebraic methods* stick with evaluations of the original kernel function. Their idea bases on the exact computation

of only some of the original matrix entries. Both, analytic and algebraic methods have in common that they are designed to reduce memory requirement and computational cost.

In the following, we present two algebraic approximation methods, the truncated singular value decomposition (SVD) in sec. 2.1 and the adaptive cross approximation (ACA) in sec. 2.2. Moreover, we point out the merits of either approach and also show how they can be combined to become even more efficient.

In contrast to the algebraic approaches, we present in sec. 2.3 an analytic approximation of the kernel function via a Chebyshev interpolation scheme. Finally, we will demonstrate the merits of this approach and show how it can be combined with algebraic approximation methods to further increase efficiency.

2.1 Singular value decomposition

Let us use the Frobenius norm

$$\|\mathsf{K}\|_F = \left(\sum_{i=1}^{M} \sum_{j=1}^{N} |(\mathsf{K})_{ij}|^2 \right)^{1/2}$$

to define the accuracy of the low-rank approximation K_r of the original matrix K as

$$\|\mathsf{K} - \mathsf{K}_r\|_F \leq \varepsilon \, \|\mathsf{K}\|_F. \tag{2.2}$$

Apparently, a tight connection exists between the accuracy ε and the rank r. The more accurate the approximation is supposed to be, the higher the rank r becomes. We can find the rank r by solving the minimization problem

$$r = \min\{\operatorname{rank}(\mathsf{K}_r) : \|\mathsf{K} - \mathsf{K}_r\|_F \leq \varepsilon \|\mathsf{K}\|_F\}.$$

The *truncated* singular value decomposition (SVD) gives the best solution to it. It provides a method to find the best-possible K_r of the original matrix K for a given accuracy ε (see [39]). First, the SVD of a matrix $\mathsf{K} \in \mathbb{C}^{M \times N}$ reads as $\mathsf{K} = \mathsf{U}\Sigma\mathsf{V}^*$ where $\mathsf{U} \in \mathbb{C}^{M \times \bar{r}}$ and $\mathsf{V} \in \mathbb{C}^{N \times \bar{r}}$ are unitary matrices with $\bar{r} = \min\{M, N\}$. $\Sigma \in \mathbb{R}_+^{\bar{r} \times \bar{r}}$ is a diagonal matrix with the singular values $\sigma_i = (\Sigma)_{ii}$ ordered such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\bar{r}}$ holds. According to the Frobenius norm defined in terms of the singular values

$$\|\mathsf{K}\|_F = \left(\sum_{i=1}^{\bar{r}} \sigma_i^2 \right)^{1/2}$$

and the error bound from (2.2) the rank r of the best approximation is determined by the condition

$$\|\mathsf{K}-\mathsf{K}_r\|_F^2 = \sum_{i=r+1}^{\bar{r}} \sigma_i^2 \leq \varepsilon^2 \sum_{i=1}^{\bar{r}} \sigma_i^2 = \varepsilon^2 \, \|\mathsf{K}\|_F^2.$$

Once we know r, we set $\{\sigma_i : r < i \leq \bar{r}\} = 0$, i.e., we truncate the singular values based on the rank r. Hence, the best-possible approximation of K reads as

$$\mathsf{K}_r = \mathsf{U}_r \Sigma_r \mathsf{V}_r^*,$$

where U_r and V_r contain only the first r columns of U and V, respectively, and Σ_r contains only the first r rows and columns of Σ.

The use of the SVD is usually not reasonable because it requires the computation and storage of all matrix entries beforehand. Beside that, the computational cost is very high, it sums up to $14MN^2 + 8N^3$ operations for a matrix $\mathsf{K} \in \mathbb{C}^{M\times N}$ with $M \geq N$ (see [12]). As a consequence, approaches based on the decomposition of the entire matrix K using the SVD will never lead to fast methods. The SVD can, however, be used in combination with other methods such as the adaptive cross approximation presented in sec. 2.2.

2.2 Adaptive Cross Approximation

The adaptive cross approximation (ACA), which was first presented in [9], is based on the idea of pseudo-skeleton approximations [48]. It has probably become the most widespread and easy-to-use tool to approximate matrices arising in boundary element formulations [15, 13]. As we will see in the following, it provides similar approximation quality as the SVD, however, it is considerably more efficient. There exist two variants of the algorithm, on one hand the *fully pivoted* ACA and on the other hand the *partially pivoted* ACA [83]. We use the former variant first, to present the basic idea of the algorithm, then, we switch to the latter variant, which is the widely used one.

Fully pivoted ACA Let us split the matrix $\mathsf{K} \in \mathbb{C}^{M\times N}$ in

$$\mathsf{K} = \mathsf{K}_r + \mathsf{R}_r$$

with the approximant K_r and the residual R_r. Then, the idea of the adaptive cross approximation (ACA) is the following: After initializing the residual $\mathsf{R}_0 = \mathsf{K}$ and the approximant $\mathsf{K}_0 = 0$, information is shifted iteratively from the residual to the approximant until the error bound $\|\mathsf{R}_r\|_F \leq \varepsilon \|\mathsf{K}\|_F$ holds. The procedure might become clearer in the following example.

Example 1. Let the matrix K be generated by the kernel function $K(x,y) = 1/|x-y|$ for equidistantly distributed points $\{x_i\}_{i=1}^5$ and $\{y_j\}_{j=1}^5$ in $[-1,-0.5] \times [0.5,1]$. The resulting matrix reads as

$$\mathsf{K} = \begin{bmatrix} 0.666 & 0.615 & 0.571 & 0.533 & 0.500 \\ 0.727 & 0.666 & 0.615 & 0.571 & 0.533 \\ 0.800 & 0.727 & 0.666 & 0.615 & 0.571 \\ 0.888 & 0.800 & 0.727 & 0.666 & 0.615 \\ 1.000 & 0.888 & 0.800 & 0.727 & 0.666 \end{bmatrix}, \quad \|\mathsf{K}\|_F = 3.486.$$

We initialize $\mathsf{R}_0 = \mathsf{K}$ and $\mathsf{K}_0 = \mathsf{0}$. Then comes the key point of the algorithm: First, we have to locate the maximal entry in modulus of the current residual, i.e., $\{i_0, j_0\} = \mathrm{ArgMax}\,|(\mathsf{R}_0)_{ij}|$, in our example they are $i_0 = 5$ and $j_0 = 1$. Then we subtract from R_0 the normalized outer product of its i_0^{th} row and j_0^{th} column vector and add it to K_0 which becomes K_1. The new residual becomes R_1 and reads as

$$\mathsf{R}_1 = \begin{bmatrix} 0 & 0.022 & 0.038 & 0.048 & 0.055 \\ 0 & 0.020 & 0.033 & 0.042 & 0.048 \\ 0 & 0.016 & 0.026 & 0.033 & 0.038 \\ 0 & 0.009 & 0.016 & 0.020 & 0.022 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \|\mathsf{R}_1\|_F = 0.134.$$

The rank of the approximant K_1 is 1. Let the prescribed accuracy be $\varepsilon = 10^{-2}$. We check whether $\|\mathsf{R}_1\|_F \leq \varepsilon \|\mathsf{K}\|_F$ already holds. Apparently with $0.134/3.486 > \varepsilon$ this is not yet the case, hence we need to repeat the procedure, now with $i_1 = 1$ and $j_1 = 5$. The residual becomes

$$\mathsf{R}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 3.10 \cdot 10^{-4} & 3.19 \cdot 10^{-4} & 1.88 \cdot 10^{-4} & 0 \\ 0 & 5.32 \cdot 10^{-4} & 5.44 \cdot 10^{-4} & 3.19 \cdot 10^{-4} & 0 \\ 0 & 5.25 \cdot 10^{-4} & 5.32 \cdot 10^{-4} & 3.10 \cdot 10^{-4} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \|\mathsf{R}_2\|_F = 1.25 \cdot 10^{-3}.$$

Finally $1.25 \cdot 10^{-3} \leq \varepsilon\ 3.486$ holds and the obtained approximant K_2 of rank 2 is sufficiently accurate.

This variant, the fully pivoted ACA, requires the computation of all entries beforehand, because the residual gets initialized with the original matrix. We will neither use nor further explain this variant. We exclusively deal with the partially pivoted ACA hereafter. In contrast to the fully pivoted variant the partially pivoted one requires the computation of only few of the original matrix entries. That makes the partially pivoted ACA so attractive.

Partially pivoted ACA A remark to the notation we use here: As usual $(\mathsf{K})_{ij}$ denotes the ij^{th} entry of the matrix K, $(\mathsf{K})_{i:}$ and $(\mathsf{K})_{:j}$ denote the i^{th} row and j^{th} column of K, respectively, and $(\mathsf{u}_n)_i$ denotes the i^{th} entry of the vector u_n, which is the n^{th} column vector of the matrix U, analogously for $(\mathsf{v}_n)_j$.

A graphic illustration of the algorithm is sketched in fig. 2.1. Apparently, in each step, which we denote by n, an outer product uv^* is added to the current approximant $\mathsf{K}_n = \mathsf{UV}^*$ which then becomes K_{n+1}, i.e., its rank increments. The key part of the algorithm is to find these vectors u and v. At step n they are computed as

$$\begin{aligned}
\hat{\mathsf{u}}_n &= (\mathsf{K})_{:j_n} - \sum_{n=1}^{n-1} (\overline{\mathsf{v}}_n)_{j_n}\, \mathsf{u}_n \quad \text{with } j_n = \operatorname{ArgMax} |(\mathsf{v}_{n-1})_j| \\
\mathsf{u}_n &= \gamma_n\, \hat{\mathsf{u}}_n \quad \text{with the pivot } \gamma_n = (\hat{\mathsf{u}}_n)_{i_n}^{-1} \text{ and } i_n = \operatorname{ArgMax} |(\hat{\mathsf{u}}_n)_i| \\
\mathsf{v}_n &= [(\mathsf{K})_{i_n:}]^* - \sum_{n=1}^{n-1} (\mathsf{u}_n)_{i_n}\, \overline{\mathsf{v}}_n .
\end{aligned}$$

Note, neither the approximant nor the residual are computed and stored explicitely at any time. The vectors $\hat{\mathsf{u}}$ and v are solely obained from rows and columns of the matrix K and a subsequential subtraction of the yet obtained rows and columns of the yet existing approximant UV^* with $\mathsf{U} \in \mathbb{C}^{M\times n}$ and $\mathsf{V} \in \mathbb{C}^{N\times n}$. They would exactly result in the respective rows

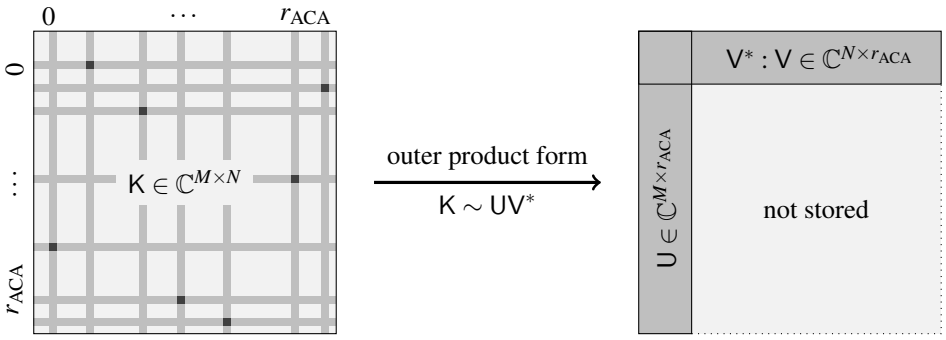


Figure 2.1: Low-rank representation

and columns of an "immaginary" residual R_n. The pivot γ_n (represented by a dark square in fig. 2.1) is chosen to be the inverse of the largest entry in modulus of $\hat{\mathsf{u}}_n$. It determines the row and column indices i_n and j_n of the n^{th} approximation step, respectively. Further hints for the right choice of the pivots γ_n and the initial index j_0 can be found in [12].

We cannot use $\|\mathsf{R}_n\|_F \leq \varepsilon\, \|\mathsf{K}\|_F$ as stopping criterion here, because in contrast to the fully pivoted ACA the matrix K is not known. We use its approximant instead and define the stopping criterion for the partially pivoted ACA as

$$\|\mathsf{u}_n\|_F\, \|\mathsf{v}_n\|_F < \varepsilon\, \|\mathsf{K}_n\|_F .$$

The Frobenius norm of the approximant which is stored in its factorized form $\mathsf{K}_n = \mathsf{U}\mathsf{V}^*$ can be computed recursively as

$$\|\mathsf{K}_n\|_F^2 = \|\mathsf{K}_{n-1}\|_F^2 + \sum_{n=1}^{n-1} [(\mathsf{u}_n\overline{\mathsf{u}}_n)(\mathsf{v}_n\overline{\mathsf{v}}_n) + (\overline{\mathsf{u}}_n\mathsf{u}_n)(\overline{\mathsf{v}}_n\mathsf{v}_n)] + \|\mathsf{u}_n\|_F^2\|\mathsf{v}_n\|_F^2$$

by using the definition of the conjugate complex value $\overline{z}$ of $z \in \mathbb{C}$.

Why choosing ACA over SVD? Basically both methods, SVD and ACA, will find a rank r representation of a matrix K: SVD will find the *best possible*, ACA the *near-best possible* one, thus $r_{\mathrm{SVD}} \leq r_{\mathrm{ACA}}$. SVD requires, however, the computation of all entries beforehand, hence, only a memory reduction can be achieved: The computational cost is of $\mathcal{O}(MN)$ to compute the entries plus $\mathcal{O}(N^2(M+N))$ for $M \geq N$ to perform the SVD for finally reducing the memory cost to $\mathcal{O}(r_{\mathrm{SVD}}(M+N))$. The decisive advantage of the ACA over the SVD is that the computational cost scales like $\mathcal{O}(r_{\mathrm{ACA}}^2(M+N))$ and the memory cost like $\mathcal{O}(r_{\mathrm{ACA}}(M+N))$ (see [12] and for the required memory also fig. 2.1).

Remark. If the cost of generating matrix entries outweighs the algebraic transformations of the algorithm the computational cost may scale like $\mathcal{O}(r_{\mathrm{ACA}}(M+N))$. If a kernel function has certain "non-nice" properties, one might need to augment the ACA algorithm in order to ensure its convergence [12]. In that case the complexity is slightly higher, however, we do not deal with such kernel functions.

2.2.1 Recompression via SVD

We know that ACA leads to the near-best possible approximant. Is there a way to exploit the efficiency of ACA and the fact that SVD allows to find the best possible approximant?

Let us write the approximant in its factorized form as $\mathsf{K}_{r_{\mathrm{ACA}}} = \mathsf{U}\mathsf{V}^*$ with $\mathsf{U} \in \mathbb{C}^{M \times r_{\mathrm{ACA}}}$ and $\mathsf{V} \in \mathbb{C}^{N \times r_{\mathrm{ACA}}}$. Next, we compute their QR decomposition

$$\mathsf{U} = \mathsf{Q}_\mathsf{U}\mathsf{R}_\mathsf{U} \quad \text{and} \quad \mathsf{V} = \mathsf{Q}_\mathsf{V}\mathsf{R}_\mathsf{V},$$

with the unitary matrices Q_U, Q_V and the upper triangular matrices R_U and R_V. Then the approximant can be rewritten as $\mathsf{K}_{r_{\mathrm{ACA}}} = \mathsf{Q}_\mathsf{U}(\mathsf{R}_\mathsf{U}\mathsf{R}_\mathsf{V}{}^*)\mathsf{Q}_\mathsf{V}^*$ and we decompose the product $\mathsf{R}_\mathsf{U}\mathsf{R}_\mathsf{V}{}^* \in \mathbb{C}^{r_{\mathrm{ACA}} \times r_{\mathrm{ACA}}}$ using the SVD

$$\mathsf{R}_\mathsf{U}\mathsf{R}_\mathsf{V}{}^* = \hat{\mathsf{U}}\Sigma\hat{\mathsf{V}}^*$$

with the unitary matrices $\hat{\mathsf{U}}$ and $\hat{\mathsf{V}}$ and the diagonal matrix Σ containing the singular values in descending order. The product of $\mathsf{Q}_\mathsf{U}\hat{\mathsf{U}}$ and $\mathsf{Q}_\mathsf{V}\hat{\mathsf{V}}$ are again unitary matrices. Hence,

$$\mathsf{K}_{r_{\mathrm{ACA}}} = \mathsf{U}\mathsf{V}^* = (\mathsf{Q}_\mathsf{U}\hat{\mathsf{U}})\,\Sigma\,(\mathsf{Q}_\mathsf{V}\hat{\mathsf{V}})^*$$

is an SVD of the near-best approximant $\mathsf{K}_{r_{\mathrm{ACA}}}$ which can hereby be truncated to the best possible approximant $\mathsf{K}_{r_{\mathrm{SVD}}}$ with $r_{\mathrm{SVD}} \leq r_{\mathrm{ACA}}$.

For the complexity estimate of this approach we refer the reader to [12]: The SVD of the matrix $\mathsf{K} \in \mathbb{C}^{M\times N}$ for $M \geq N$ sums up to $\mathcal{O}(N^2(M+N))$ operations, whereas the here presented approach requires $\mathcal{O}(r^2(M+N))$ operations for the ACA and $\sim 6r^2(M+N) + 20r^3$ operations which sums up to a total of only $\mathcal{O}(r^2(M+N))$ operations, as long as $r \ll \min\{M,N\}$ is true.

2.3 Directional interpolation of oscillatory kernels

The aim of this chapter is to develop a polynomial interpolation scheme to efficiently and accurately approximate both, *asymptotically smooth* and *oscillatory* kernel functions. A very formal description of such schemes can be found in [20]. They represent a simple and powerful approach to approximate a broad range of kernel functions (see [56]). The directional interpolation of oscillatory kernels has already been published in [76]. We will present it here in more detail.

The previsously presented truncated SVD and ACA compute low-rank approximants K_r iteratively, by explicitely checking the approximation error. They neccessarily converge for both kernel types to the best and near-best low-rank approximations, respectively (see [39, 12]). This is different when using polynomial kernel interpolations. Their basic concept are separable kernel expansion. In [56], we found the following definition for that:

Definition 2. Let us assume that the kernel function $K(x,y)$ with $x \in X \subset \mathbb{R}^d$ and $y \in Y \subset \mathbb{R}^d$ can be split up as $K(x,y) = K_\ell(x,y) + R_\ell(x,y)$. Then, any function

$$K_\ell(x,y) = \sum_{n=1}^{\ell} u_n(x)\, v_n(y)$$

is a separable kernel expansion in $X \times Y$ and approximates $K(x,y)$. The functions $u_n(x)$ and $v_n(y)$ may be any function depending only on x and y, respectively. ℓ denotes the truncation rank and the expression

$$R_\ell(x,y) = \sum_{n=\ell+1}^{\infty} u_n(x)\, v_n(y)$$

is the remainder.

Recall, the entries of the matrix $\mathsf{K} \in \mathbb{C}^{M\times N}$ are computed by evaluating the associated kernel function $K(x,y)$. Thus, when using a truncated kernel expansion $K_\ell(x,y)$ the resulting low-rank approximant $\mathsf{K}_\ell \in \mathbb{C}^{M\times N}$ is defined as

$$\mathsf{K}_\ell = \mathsf{U}\mathsf{V}^* \quad \text{with } U \in \mathbb{C}^{M\times \ell},\ V \in \mathbb{C}^{N\times \ell}$$

and the entries are computed as $(\mathsf{U})_{in} = u_n(x_i)$ and $(\mathsf{V}^*)_{nj} = v_n(y_j)$ for all $x_i \in X$ and $y_i \in Y$. The cost for computing and storing K_ℓ is $\mathcal{O}(\ell(M+N))$. Note, the different notation for the low-rank: Here we let ℓ denote the low-rank, when we use the SVD or ACA we let r denote the low-rank of the approximant. The reason is that ℓ needs to be set apriori, whereas r is determined during the approximation procedure. This is a crucial difference between these approximation approaches.

A central aim is to derive so called admissibility criteria for X and Y to determine the truncation rank ℓ such that the required accuracy of the resulting approximant $\|\mathsf{K}-\mathsf{K}_\ell\| \leq \varepsilon\|\mathsf{K}\|$ is satisfied. More importantly, we require the remainder $|R_\ell(x,y)|$ to decay exponentially as the truncation rank grows. The analysis is based on the theory of Chebyshev expansions and is presented in sec. 2.3.1. In sec. 2.3.2, we finally construct a Chebyshev interpolation scheme for oscillatory kernels.

Another important fact is that an a priori determinded low-rank is indepent on the actual size of the matrix. This is normally not the case, though, and such approximation approaches might lead to sub-optimal approximants. In section 2.3.3, we present an approach to overcome this issue.

2.3.1 Truncated Chebyshev expansion

A natural way to derive a polynomial approximation of $K(x,y)$ is the expansion in a truncated Taylor series in either x or y. This requires the analytical computation of derivatives up to the expansion order. Hence, such approaches become less attractive the more accurate the kernel needs to be approximated and also the more complicated it is. Other ideas are to approximate the kernel by using multipole expansions based on spherical harmonics (see [51, 84] for more details).

We use a truncated Chebyshev expansion whose definition we found in [74].

Definition 3. Let a continuous function $f(x)$ with $x \in [-1,1]$ be approximated via a truncated Chebyshev expansion of rank ℓ

$$p_\ell(x) = \frac{1}{2}c_0 + \sum_{n=1}^{\ell} c_n T_n(x)$$

with the Chebyshev polynomials of the first kind $T_n(x) = \cos(n \arccos x)$ of order n and the coefficients c_n are given by

$$c_n = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x)\, T_n(x)}{\sqrt{1-x^2}} dx \quad \text{for } x \in [-1,1].$$

There exist refined error estimates for truncated Chebyshev expansions [74]. The following theorem gives a tool to control their accuracy.

Theorem 1. Assuming the function $f(x)$ can be extended to a function $f(z)$ then the error bound

$$|f(x) - p_\ell(x)| \leq \frac{M}{\rho^\ell(\rho-1)} \quad \text{for } x \in [-1,1] \text{ and } \rho > 1$$

with the function $f(z)$ being analytic on the ellipse E_ρ and M is such that

$$|f(z)| \leq M \quad \text{for } z \in E_\rho \text{ with } E_\rho = \left\{ z \in \mathbb{C} | z = \frac{\rho \mathrm{e}^{\imath\theta} + \rho^{-1} \mathrm{e}^{-\imath\theta}}{2} \right\}$$

holds. As long as M is bounded for z in E_ρ, the error decays exponentially like $\mathcal{O}(1/\rho^\ell)$.

Directional expansion of oscillatory kernels

With these tools at hand we are ready to prepare the basis for the construction of a truncated Chebyshev expansion $K_\ell(x,y)$ of the oscillatory kernel $K(x,y)$ where the remainder $|K(x,y) - K_\ell(x,y)|$ decays exponentially as ℓ grows. We start with an introductory example:

Example 2. Let us analyse the error of a truncated Chebyshev expansion of the oscillatory function

$$f(x) = \mathrm{e}^{\imath k x} \quad \text{for } x \in \mathbb{R}$$

depending on the truncation rank ℓ and the wave number $k > 0$. After extending $f(x)$ to $f(z)$ with $z \in \mathbb{C}$ and choosing $\theta = -\pi/2$ in the definition of the ellipse E_ρ we obtain $z = -\imath(\rho + 1/\rho)/2$. We plug this into $f(z)$ and end up with $\mathrm{e}^{k/2(\rho+1/\rho)}$, which apparently grows exponentially fast in k for any admissible ρ and no constant M exists which is independent of the wave number k. Hence, there exists no truncated Chebyshev expansion approximating f for a fixed accuracy while varying k. As k increases, the oscillations must be resolved by increasing correspondingly ℓ.

Even though the reasoning is more complicated for the oscillatory kernel, we can conclude due to the similarity to example 2 that we cannot achieve a k independend approximation by directly deriving a truncated Chebyshev expansion. The idea is to introduce a modified

representation of the oscillatory kernel, similarly as in [23]. Based on that, we show for some X and Y its boundedness. Although, we derive the procedure for the Helmholtz kernel, it applies to all oscillatory kernels having a multiplicative term $\mathrm{e}^{\imath k|x-y|}$.

Let us introduce some notations. Recall the clusters X and the cluster Y containing the points x and y, respectively. Moreover, X is centered at c_x and Y at c_y. By introducing the vectors $r_x = x - c_x$, $r_y = y - c_y$, $r = r_x - r_y$ and $c = c_x - c_y$ we can rewrite the vector between the two points x and y as

$$x - y = c + r,$$

as shown in shown in fig. 2.2. All these newly introduced vectors are in $\mathbb{R}^3$.

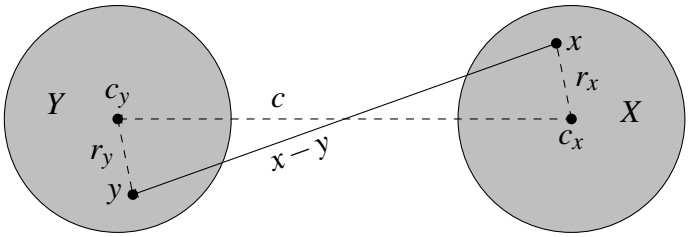


Figure 2.2: Two clusters X and Y and the vectors c, r_x and r_y

We split the oscillatory kernel in two multiplicative parts $K = K^s K^o$ which read as

$$K^s(x,y) = \frac{1}{4\pi}\frac{1}{|x-y|} \quad \text{and} \quad K^o(x,y) = \mathrm{e}^{\imath k|x-y|} \quad \text{for } x,y \in \mathbb{R}^3,$$

K^s represents the asymptotically smooth and K^o the oscillatory kernel part.

First, we look at the *asymptotically smooth* kernel part K^s. We use the relation $x - y = c + r$ and see that it is analytic unless $|c + r| = 0$. To find a stronger admissible relation between c and r we expand the denominator $|c + r|$ into a Taylor series in the vicinity of r. By rewriting

$$|c + r| = |c|\sqrt{1 + \frac{|r|^2}{|c|^2} + 2\frac{c \cdot r}{|c|^2}},$$

and by substituting $\xi = (|r|^2 + 2\, c \cdot r)/|c|^2$ we can use the Taylor expansion for $\sqrt{1+\xi} = 1 + \frac{1}{2}\xi - \frac{1}{8}\xi^2 + \mathcal{O}(\xi^3)$ for $|\xi| \leq 1$ and get

$$|c + r| = |c| + \frac{1}{2}\frac{|r|^2}{|c|} + \frac{c \cdot r}{|c|} - \frac{1}{2}\frac{(c \cdot r)^2}{|c|^3} + \mathcal{O}(|r|^3), \tag{2.3}$$

as long as the relation $|r|/|c| \leq \sqrt{2} - 1$ holds. Now, we look for the situation where $|c + r|$ most likely approaches 0. Certainly this is not the case if $r \perp c$, unless $|c| = 0$ the kernel

K^s cannot become singular in that case. However, if $r \parallel c$ with r and c pointing in opposite direction, such that $r \cdot c = -|r||c|$, the expansion (2.3) becomes $|c| - |r|$. Hence, we obtain a stronger condition $|r| < |c|$. It implies that the clusters X and Y are not allowed to touch themselves, i.e., $X \cap Y = \emptyset$. This is an often used condition later on, if it holds, we say the clusters X and Y are *well separated*. If we analyse the kernel in the complex plane and choose $\rho > 1$ such that $K^s(z)$ is analytic within the ellipse E_ρ, then there exists an upper bound $|K^s(z)| \leq M$. Hence, the non-oscillatory kernel part K^s can be approximated by a truncated Chebyshev expansion unless cluster X and Y overlap.

Next, we analyse the *oscillatory* part K^o of the kernel. From Example 2 we know that such functions grow exponentially fast with k in the complex plain. Hence, no truncated Chebyshev expansion of truncation rank ℓ which is independent on k exists. We put up the following theorem:

Theorem 2. Let the oscillatory kernel have the decomposition $K^o(x,y) = K^u(x,y)\, \mathrm{e}^{\imath k u \cdot (x-y)}$ with

$$K^u(x,y) = \mathrm{e}^{\imath k(|x-y| - u\cdot(x-y)} \quad \text{for } x \in X,\ y \in Y,$$

the unit vector $u \in \mathbb{R}^3$ and $|u| = 1$. Both clusters $X, Y \subset \mathbb{R}^3$ have the diameter w. If there exists a constant γ such that

$$kw \left| \frac{c}{|c|} - u \right| \leq \gamma \qquad \text{and} \qquad \frac{kw^2}{|c|} \leq \gamma \tag{2.4}$$

hold, then K^u has a truncated Chebyshev expansion of fixed accuracy and truncation rank ℓ while varying k.

Proof. We need to show that the kernel K^u is bounded independently of k in the ellipse E_ρ for some $\rho > 1$. First, we set $x - y = c + r$ and expand the exponent of K^u into a Taylor series in the vicinity of r. By using (2.3) we get

$$|c+r| - u \cdot (c+r) = |c| + \frac{1}{2}\frac{|r|^2}{|c|} + \frac{c\cdot r}{|c|} - \frac{1}{2}\frac{(c\cdot r)^2}{|c|^3} - u\cdot(c+r) + \ldots, \tag{2.5}$$

which we plug into K^u. Next, we extend $r \in \mathbb{R}^3$ to $z \in \mathbb{C}^3$ and obtain

$$K^u(z) = \exp\left(\imath k \left(|c| - u\cdot c + \left(\frac{c}{|c|} - u\right)\cdot z + \frac{1}{2}\frac{|z|^2}{|c|} - \frac{1}{2}\frac{(c\cdot z)^2}{|c|^3} + \ldots \right)\right).$$

If there exists a constant γ such that the bounds (2.4) hold, then there exists an upper bound for $|K^u(z)| \leq M$ in the Ellipse E_ρ which is independent of k. □

From the bounds presented in Theorem 2 we can derive the following two criteria:

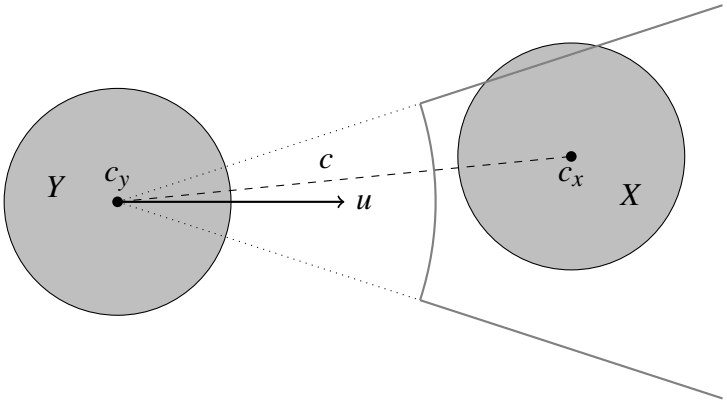


Figure 2.3: Directional admissibility criteria

Cone aperture criterion We introduce a cone with the origin at the center of Y, the direction given by the unit vector u and an aperture of $|c/|c| - u|$. If this aperture is at most $\mathcal{O}(1/kw)$ the first bound in Theorem 2 holds.

Separation criterion The distance of the centers of cluster X and Y is denoted by $|c|$. If this separation is at least $\mathcal{O}(kw^2)$ the second bound in Theorem 2 holds.

These conditions are also depicted in fig. 2.3. If both conditions are satisfied a truncated Chebyshev expansion of $K^u(x,y)$ exists and, consequently, also of $K(x,y)$ whose remainder decays exponentially for growing truncation rank ℓ. The approximation is independent on the wave number k.

2.3.2 Directional Chebyshev interpolation

The Chebyshev interpolation is a very good substitute to the truncated Chebyshev expansion we presented above [74].

The simplest way to construct a polynomial approximation of degree ℓ to a given continuous function $f(x)$ with $x \in [-1,1]$ is to interpolate $f(x)$ at $\ell+1$ equidistant points $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_\ell$ in the intervall $[-1,1]$ by the polynomial

$$f_\ell(x) = \sum_{i=0}^{\ell} c_i x^i.$$

The only requirement is to know the argument $f(x_i)$ at all interpolation points $\bar{x}_i$. With that we get a set of $\ell+1$ linear equations which we can solve for all unknown coefficients c_i.

However, the construction of such a polynomial approximation besides being time consuming becomes also numerically unstable if ℓ grows. There exist other, more efficient

and reliable interpolation schemes. In order to get a measure of the quality of an approximation we need to define the best-possible approximation. Let us introduce the maximum error of an approximation $p(x)$ of $f(x)$ in the interval $x \in [a,b]$

$$\max_{a \le x \le b} |f(x) - p(x)|.$$

The polynomial $p(x)$ of order at most ℓ which minimizes the maximum error

$$\min_{\text{order } p(x) \le \ell} \left\{ \max_{a \le x \le b} |f(x) - p(x)| \right\}$$

provides the sought after best-possible approximation, also called *minimax approximation* [74]. Apparently the construction of such approximations is rather cumbersome because a minimization problem needs to be solved.

The *Lebesgue constant* gives an idea on how good a given interpolant of a function is, compared to its best-possible approximation of the same degree ℓ. Interpolants at equidistant points have a Lebesgue constant which grows exponentially in ℓ. On the other hand, the Lebesgue constant grows only logarithmically when using Chebyshev nodes as interpolation points

$$\bar{x}_m = \cos\left(\frac{\pi}{2}\frac{2m-1}{\ell}\right) \quad \text{for } m = 1, \ldots, \ell, \tag{2.6}$$

which are the zeros of the Chebyshev polynomials $T_\ell(x)$ of order ℓ

$$T_\ell(x) = \cos\left(\ell \arccos x\right) \quad \text{for } x \in [-1,1].$$

Indeed, interpolating at Chebyshev nodes gives us a *near-minimax approximation*, which is the optimal compromise for efficiency and accuracy we are looking for. For more details we refer the reader to [82], [25] and [89].

Moreover, the discrete othogonality (see [74]) of the Chebyshev polynomials

$$\sum_{m=1}^{\ell+1} T_i(\bar{x}_m)\, T_j(\bar{x}_m) = \begin{cases} 0 & i \neq j \ (\le \ell) \\ \ell + 1 & i = j = 0 \\ (\ell+1)/2 & 0 < i = j \le \ell \end{cases}$$

with the set of zeros $\{\bar{x}_m\}$ of $T_{\ell+1}(x)$, leads to a very efficient interpolation formula: The polynomial $f_\ell(x)$ interpolates the function $f(x)$ in the Chebyshev nodes (2.6) as a sum of Chebyshev polynomials in the form

$$f_\ell(x) = \frac{1}{2}c_0 + \sum_{i=1}^{\ell} c_i\, T_i(x) \tag{2.7}$$

for $x \in [-1,1]$ with the coefficients

$$c_i = \frac{2}{\ell+1} \sum_{m=1}^{\ell+1} f(\bar{x}_m)\, T_i(\bar{x}_m). \tag{2.8}$$

After inserting (2.8) into (2.7) and swapping the sums we get the final interpolation formula

$$f_\ell(x) = \sum_{m=1}^{\ell+1} f(\bar{x}_m)\, S_\ell(x, \bar{x}_m) \tag{2.9}$$

for $x \in [-1,1]$ with the interpolating polynomial (see also [42])

$$S_\ell(x, \bar{x}_m) = \frac{1}{\ell+1} + \frac{2}{\ell+1} \sum_{i=1}^{\ell} T_i(x)\, T_i(\bar{x}_m).$$

Arbitrary interval Most real problems, however, require the function to be defined in an arbitrary interval $x \in [a,b]$. Hence, we need to map the arbitrary interval $[a,b]$ to the reference interval $[-1,1]$ and vice versa. We introduce the affine mapping functiong $\Phi : [-1,1] \to [a,b]$, given by

$$\Phi(x) = \frac{a+b}{2} + \frac{b-a}{2} x.$$

It maps from the reference interval $[-1,1]$ to any arbitrary interval $[a,b]$ and the inverse mapping $\Phi^{-1} : [a,b] \to [-1,1]$, given by

$$\Phi^{-1}(x) = \frac{2x-b-a}{b-a}$$

maps from any interval $[a,b]$ to the reference interval $[-1,1]$. If we now interpolate a function $f(x)$ in the arbitrary interval $[a,b] \subset \mathbb{R}$ we need to rewrite (2.9) using the mapping function $\Phi(x)$ as

$$f_\ell(x) = \sum_{m=1}^{\ell+1} f\left(\Phi(\bar{x}_m)\right)\; S_\ell\left(\Phi^{-1}(x), \bar{x}_m\right) \quad \text{for } x \in [a,b].$$

Obviously, the Chebyshev points $\{\bar{x}_m\}$ which are defined in the reference interval $[-1,1]$ has to be mapped to the interval $[a,b]$ when interpolating $f(x)$ there. Moreover, when evaluating the interpolating polynomial $S_\ell(x, \bar{x}_m)$ the point $x \in [a,b]$ has to be mapped to the reference intervall $[-1,1]$.

Multidimensional interpolation We let the d-dimensional reference interval $[-1,1]^d$ be defined via the tensor product

$$[-1,1]^d = [-1,1]_1 \times \cdots \times [-1,1]_d.$$

Then, we construct the interpolation nodes $\bar{x}_\alpha \in [-1,1]^d$ in the same way

$$\bar{x}_\alpha = \bar{x}_{\alpha_1} \times \cdots \times \bar{x}_{\alpha_d}, \tag{2.10}$$

where $\bar{x}_{\alpha_i}$ are the zeros of the Chebyshev polynomials $T_{\ell+1}(x_i)$ with $x_i \in [-1,1]_i$ and $\alpha = (\alpha_1, \cdots, \alpha_d)$ is a d-dimensional multi-index with $\alpha_i = 1, \ldots, \ell+1$ for all $i = 1, \ldots, d$, hence $|\alpha| \leq (\ell+1)^d$. The d-dimensional interpolation polynomial is constructed analogously

$$S_\ell(x, \bar{x}_\alpha) = S_\ell(x_1, \bar{x}_{\alpha_1}) \times \cdots \times S_\ell(x_d, \bar{x}_{\alpha_d}) \quad \text{for } x \in [-1,1]^d. \tag{2.11}$$

Hence, the interpolation of the function $f(x)$ read as

$$f_\ell(x) = \sum_{m \in \alpha} f(\bar{x}_m)\, S_\ell(x, \bar{x}_m) \quad \text{for } x \in [-1,1]_d.$$

Interpolating derivatives We recall the derivative of the Chebyshev polynomials

$$\frac{dT_\ell(x)}{dx} = \ell U_{\ell-1}(x) \quad \text{with} \quad U_{\ell-1}(x) = \frac{\sin(\ell \arccos x)}{\sqrt{1-x^2}} \tag{2.12}$$

for $x \in [-1,1]$ and $U_\ell(x)$ are the Chebyshev polynomials of second kind. Based on them we can derive the interpolation polynomial of the derivative of $f(x)$

$$\frac{df_\ell(x)}{dx} = \sum_{m=1}^{\ell+1} f(\bar{x}_m)\, P_\ell(x, \bar{x}_m) \quad \text{with} \quad P_\ell(x, \bar{x}_m) = \frac{dS_\ell(x, \bar{x}_m)}{dx}.$$

We observe that nothing happens with the function $f(x)$. The derivative d/dx gets shifted to the polynomial $S_\ell(x, \bar{x})$ and by using the relation (2.12) the polynomial $P_\ell(x, \bar{x})$ reads as

$$P_\ell(x, \bar{x}_m) = \frac{2}{\ell+1} \sum_{i=1}^{\ell} i U_{i-1}(x)\; T_i(\bar{x}_k) \quad \text{for } x \in [-1,1].$$

In the multidimensional case the interpolation of the gradient of $\nabla f(x) \sim \nabla f_\ell(x)$ reads as

$$\nabla f_\ell(x) = \sum_{m \in \alpha} f(\bar{x}_m)\, P_\ell(x, \bar{x}_m) \quad \text{for } x \in [-1,1]_d.$$

The gradient of the interpolating polynomial $P_\ell(x, \bar{x}) = \nabla S_\ell(x, \bar{x})$ is computed as

$$P_\ell(x, \bar{x}_m) = \begin{pmatrix} P_\ell(x_1, \bar{x}_{\alpha_1}) \times \cdots \times S_\ell(x_i, \bar{x}_{\alpha_i}) \times \cdots \times S_\ell(x_d, \bar{x}_{\alpha_d}) \\ \vdots \\ S_\ell(x_1, \bar{x}_{\alpha_1}) \times \cdots \times P_\ell(x_i, \bar{x}_{\alpha_i}) \times \cdots \times S_\ell(x_d, \bar{x}_{\alpha_d}) \\ \vdots \\ S_\ell(x_1, \bar{x}_{\alpha_1}) \times \cdots \times S_\ell(x_i, \bar{x}_{\alpha_i}) \times \cdots \times P_\ell(x_d, \bar{x}_{\alpha_d}) \end{pmatrix} \tag{2.13}$$

for $x \in [-1,1]^d$. Note, $P_\ell(x,\bar{x})$ in (2.13) is a vector function here, in contrast to $S_\ell(x,\bar{x})$ in (2.11) which is a scalar function. The extension from the reference interval $[-1,1]_d$ to an arbitrary interval $[a,b]_d$ is straightforward by simply plugging the mapping function $\Phi(x)$ into the equations. For the sake of readability, however, we limit our explanations of the ongoing tasks to the reference interval $[-1,1]$. Such can be made true by simply scaling the the geometry, the simplification makes the notation much simpler, though.

Interpolating the oscillatory kernel

In the following, we use the Chebyshev interpolation to construct a polynomial approximation of the oscillatory kernel $K(x,y)$. Only evaluations of $K(x,y)$ at some given interpolation points $\{\bar{x}\}$ and $\{\bar{y}\}$ are required. This makes polynomial interpolation very attractive because no modification of the kernel is required.

Recall, the directional representation of the oscillatory kernel reads as

$$K(x,y) = K^s(x,y)\, K^u(x,y)\, \mathrm{e}^{\imath k u\cdot(x-y)} \tag{2.14}$$

for $x \in X$ and $y \in Y$ with $X, Y \subset \mathbb{R}^3$ and is bounded in the complex plane. There exists a Chebyshev expansion of K^u whose truncation rank ℓ does not dependent on the wave number k.

Frequency regimes The number of waves which have to be approximated does not only depend on the wave number k but also on the size of the computational domain, which is represented by the clusters X and Y. Think of the function $\mathrm{e}^{\imath kx}$ with $x \in [a,b]$ where the wavelenght is $2\pi/k$. The more we enlarge the interval $[a,b]$ the more waves have to be approximated. If we use a polynomial interpolation of fixed order ℓ the quality of the approximation suffers more and more. Obviously, it depends on the number of waves that have to be reproduced. On the other hand, if $|a-b|$ is at most $\mathcal{O}(1/k)$ the function $\mathrm{e}^{\imath kx}$ itself can be considered smooth in the interval $[a,b]$. Apparently, the relation of cluster size $w = \min\{\operatorname{diam} X, \operatorname{diam} Y\}$ and wave number k is a convenient tool to define two frequency regimes:

Definition 4. There exists a constant B such that the cluster pair X and Y is in the low frequency regime for $w \leq B/k$ and in the high frequency regime for $w > B/k$.

Low frequency regime For the constant $B \leq \min\{\gamma/\eta, \gamma/2\}$ equation (2.4) is always true. Moreover, it is sufficient to satisfy the separation criterion $w \leq \eta|c|$ for X and Y and the interpolant of the oscillatory kernel reads as

$$K(x,y) \sim \sum_{m\in\alpha} S_\ell(x,\bar{x}_m) \sum_{n\in\alpha} K(\bar{x}_m,\bar{y}_n)\, S_\ell(y,\bar{y}_n). \tag{2.15}$$

The arbitrary parameter $\eta > 0$ comes from the well known separation criterion for asymptotically smooth kernel functions [12, 56].

High frequency regime Both criteria of (2.4) need to be satisfied: If X and Y fulfill the minimal separation of $\mathcal{O}(kw^2)$ and the maximal cone aperture of $\mathcal{O}(1/kw)$ an interpolant of $K^u(x,y)$, which reads as

$$K^u(x,y) \sim \sum_{m\in\alpha} S_\ell(x,\bar{x}_m) \sum_{n\in\alpha} K^u(\bar{x}_m,\bar{y}_n)\, S_\ell(y,\bar{y}_n),$$

exists. After inserting it into (2.14) the interpolated oscillatory kernel reads as

$$K(x,y) \sim \mathrm{e}^{\imath k u\cdot x} \sum_{m\in\alpha} S_\ell(x,\bar{x}_m)\, \mathrm{e}^{-\imath k u\cdot \bar{x}_m} \sum_{n\in\alpha} K(\bar{x}_m,\bar{y}_n)\, \mathrm{e}^{\imath k u\cdot \bar{y}_n} S_\ell(y,\bar{y}_n)\, \mathrm{e}^{-\imath k u\cdot y}. \quad (2.16)$$

Recall, the multi-index α with $|\alpha| \le (\ell+1)^d$. For $d=3$ and with ℓ denoting the number of interpolation points there are about $(\ell+1)^3$ interpolation points in each cluster.

Interpolation error

To investigate the Chebyshev interpolation error we use an estimate from [74]. Conversely to the error bound presented in Theorem 1 which holds for truncated Chebyshev expansions, here we analyze the error of the Chebyshev interpolation.

Theorem 3. If the function $f(x)$ extends to a function $f(z)$ of the complex variable z, which is analytic on the ellipse E_ρ then the estimate

$$\left| f(x) - \sum_{m=1}^{\ell+1} S_\ell(x,\bar{x}_m) f(\bar{x}_m) \right| \le \frac{(\rho+\rho^{-1})M}{(\rho^{\ell+1}-\rho^{-\ell-1})(\rho+\rho^{-1}-2)} \quad (2.17)$$

holds for all $-1 \le x \le 1$ and with M and E_ρ defined in Theorem 1.

For a given interpolation order $\ell+1$ we obtain the most strict error bound if we find the optimal compromise between minimizing M and maximizing ρ.

How do we estimate the interpolation error of a kernel function $K(x,y)$ which depends on two variables? Let us take two clusters $X,Y \subset \mathbb{R}^3$ that are centered at c_x and c_y, respectively. Recall, $c = c_x - c_y$ and $r = x - y - c$. We can rewrite $x - y = c + r$ where c is constant. Thus, $K(x,y)$ becomes $K(r)$ and can be extended to $K(z)$ with $z \in \mathbb{C}^3$.

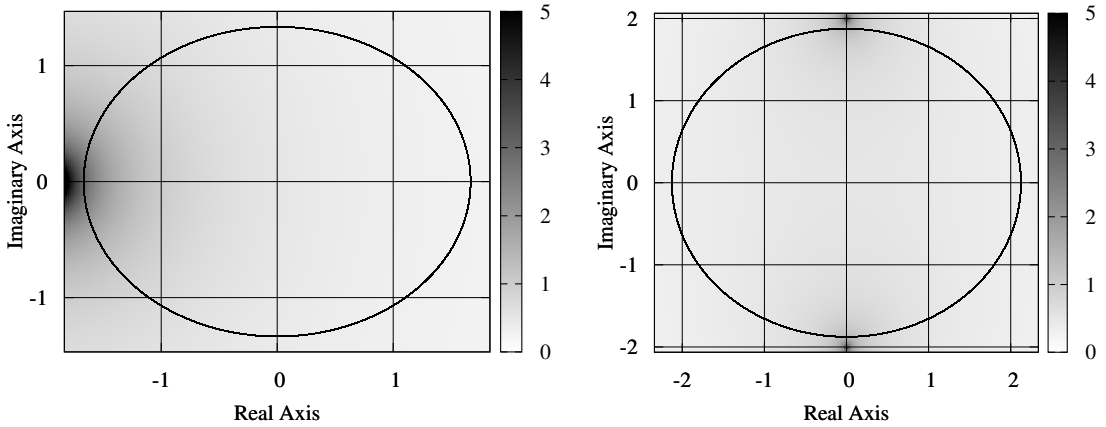
Based on the following two examples we analyse the interpolation error of the kernels K^s and K^u. In each example, we let the clusters $X,Y \subset \mathbb{R}^2$ be of diameter $w=1$ and the cluster X be centered at the origin $c_x = [0,0]$. Only the center c_y of cluster Y changes.

Example 3. Let us analyse the interpolation error of the *asymptotically smooth* kernel

$$K^s(r) = \frac{1}{|c+r|}.$$

We let cluster Y be centered at $c_y = [2,0]$ and vector $c = c_y - c_x$ becomes $c = [2,0]$. The interpolation error for $K^s(r)$ is analyzed for two cases

$$r^{\parallel} = [\zeta, 0] \quad \text{varies parallel to } c, \text{ and}$$
$$r^{\perp} = [0, \zeta] \quad \text{varies perpendicular to } c, \quad \text{for all } -1 \leq \zeta \leq 1.$$



(a) $r^{\parallel} = [\zeta, 0]$ and E_ρ for $\rho = 3$ (b) $r^{\perp} = [0, \zeta]$ and E_ρ for $\rho = 4$

Figure 2.4: Modulus of $K^s(r) = \frac{1}{|c+r|}$ with $r \in \mathbb{C}^2$

figure 2.4 shows the modulus of $K^s(z)$ in the complex plane for these two cases. The ellipse E_ρ is chosen such that $K^s(z)$ remains analytic on it.

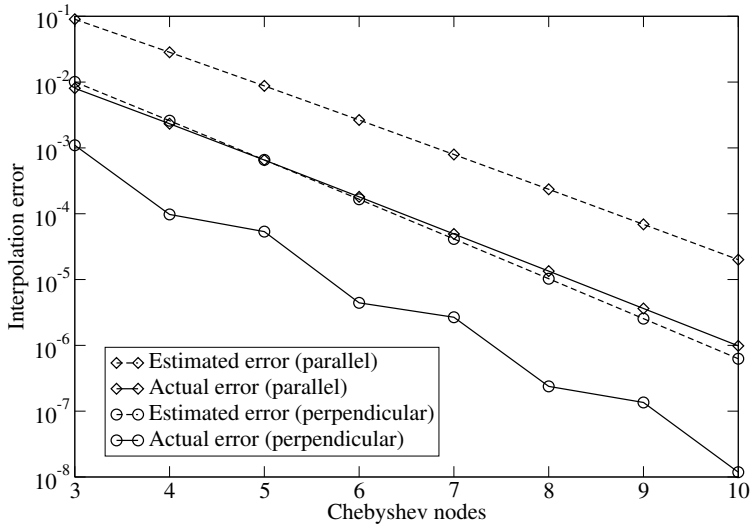


Figure 2.5: Interpolation error for $K^s(r) = \frac{1}{|c+r|}$

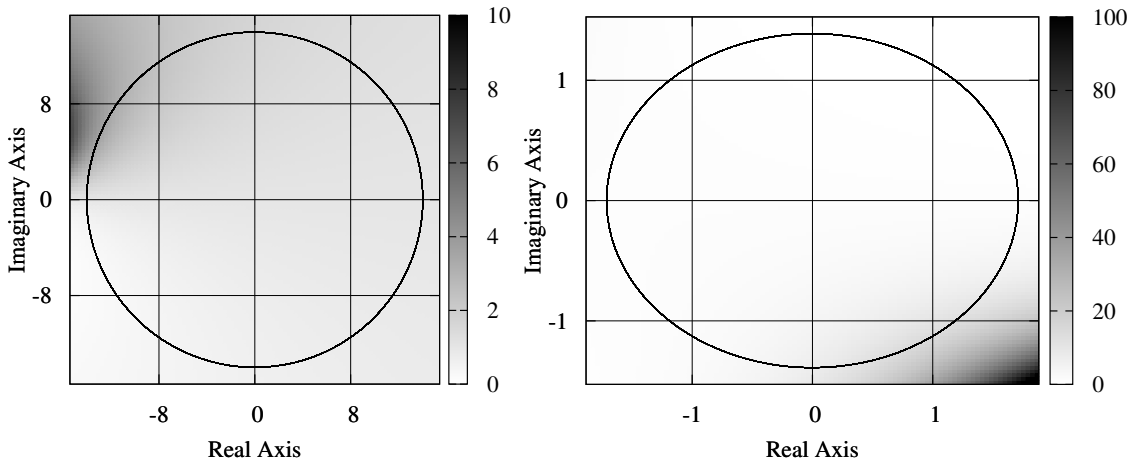
The estimated and actual interpolation error for $r^{\parallel}$ and $r^{\perp}$, with respective poles at $[-2,0]$ and $[0,\pm 2\imath]$, are compared in fig. 2.5. The error decay is of order $\mathcal{O}(1/\rho^{\ell})$ as expected.

Example 4. Let us analyze the interpolation error of the kernel

$$K^u(r) = \mathrm{e}^{\imath k(|c+r|-u\cdot(c+r))}.$$

The center of cluster Y depends on the wave number k. We choose it such that the separation criterion $\mathcal{O}(k)$ and cone aperture criterion $\mathcal{O}(1/k)$ for $u = [1,0]$ are satisfied, i.e., $c = c_y = [k,1]$. Again, we analyze two cases:

$$\begin{aligned} r^{\parallel} &= [\zeta,0] \quad \text{varies parallel to } u, \text{ and} \\ r^{\perp} &= [0,\zeta] \quad \text{varies perpendicular to } u, \quad \text{for all } -1 \le \zeta \le 1. \end{aligned}$$



(a) $r^{\parallel} = [\zeta,0]$ and E_ρ for $\rho = 30$ (b) $r^{\perp} = [0,\zeta]$ and E_ρ for $\rho = 3$

Figure 2.6: Modulus of $K^u(r) = \mathrm{e}^{\imath k(|c+r|-u\cdot(c+r))}$ with $r \in \mathbb{C}^2$ and for $k = 20$

Figure 2.6 shows the modulus of the oscillatory kernel $K^u(z)$ in the complex plane for these two cases. We notice that for $r^{\parallel}$ the kernel increases at a much slower rate than for $r^{\perp}$. This implies that we can choose larger ρ and, due to estimate (2.17), the interpolation error must be smaller. We can observe that in fig. 2.7. There, the actual and estimated interpolation error for $k = 2$, 20, 200, 2000, and $r^{\parallel}$ and $r^{\perp}$ are compared. We can understand this behavior if we look at the Taylor expansion of the exponent of $K^u(r)$ in (2.5): when k increases, the angle between c and u becomes smaller, and all the terms shown in (2.5) become negligible (they cancel out when c and u are perfectly aligned). The reason why the actual interpolation error for growing k and ℓ plateaus at some point is due to rounding errors in the floating-point operations.

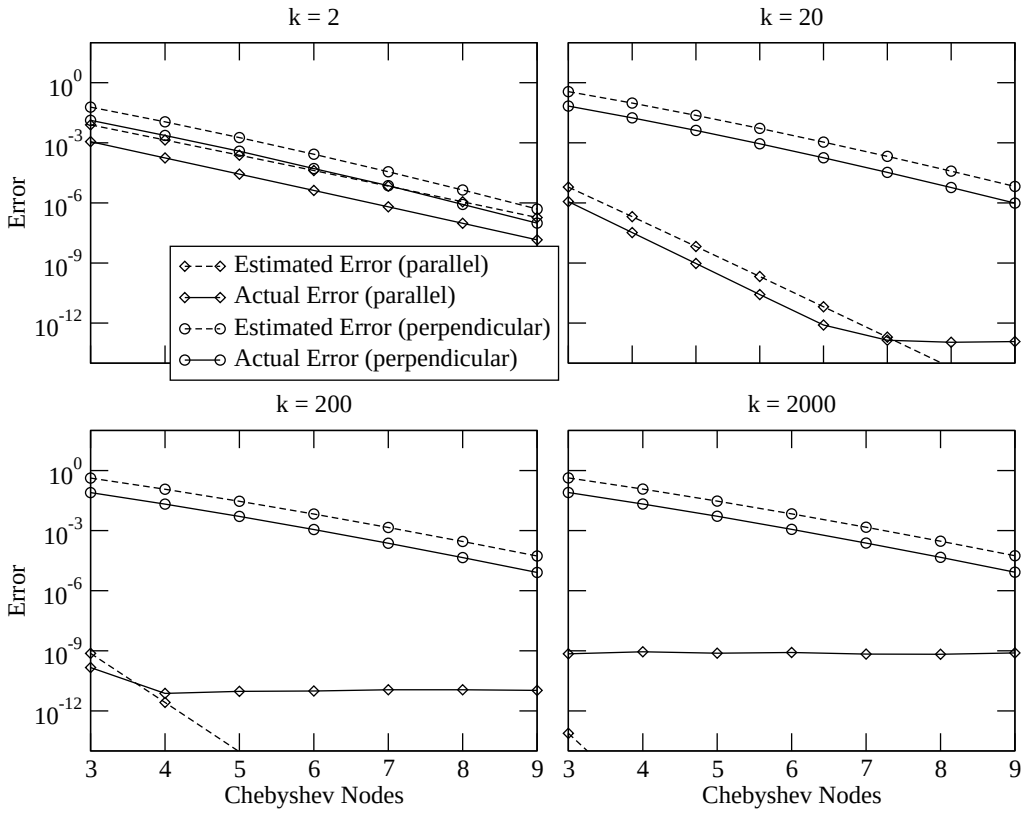


Figure 2.7: Interpolation error for $K^u(r) = \mathrm{e}^{ik(|c+r|-u\cdot(c+r))}$

2.3.3 Efficient treatment of interpolants

Before we come to the efficient treatment we introduce the matrix representation of interpolants. First, we need to clarify an important but somewhat confusing point: Eventhough in (2.15) and (2.16) both polynomials S_ℓ interpolating $K(x,y)$ in x and in y, respectively, are equivalent, in matrix notation they are indeed transposed operations. It will become clear in a moment.

The low and high frequency regime have to be treated separately also in matrix notation. However, we will find out that there exist some commonalities. Recall, the kernel $K(x,y)$ is defined for $x \in X$ and $y \in Y$. In the low frequency regime, the low-rank representation of $\mathsf{K} \in \mathbb{C}^{M\times N}$ for admissible clusters X and Y reads as

$$\mathsf{K}_\ell = \mathsf{S}_X \, \bar{\mathsf{K}} \, \mathsf{S}_Y^*, \tag{2.18}$$

with $\mathsf{S}_X \in \mathbb{C}^{M\times L}$, $\mathsf{S}_Y \in \mathbb{C}^{N\times L}$ and $\bar{\mathsf{K}} \in \mathbb{C}^{L\times L}$ with the multi-index $|\alpha| \leq L = (\ell+1)^d$. The entries are computed as $(\bar{\mathsf{K}})_{mn} = K(\bar{x}_m, \bar{y}_n)$, as $(\mathsf{S}_X)_{im} = S_\ell(x_i, \bar{x}_m)$ and as $(\mathsf{S}_Y^*)_{nj} =$

$S_\ell(y_j,\bar{y}_n)$ for $i=1,\dots,M$, $j=1,\dots,N$ and $m,n=1,\dots,L$.

If we write the low-rank representation of K in the high frequency regime in the same way as in the low frequency regime, the entries of the interpolation matrices are computed differently, i.e., as $(\mathsf{S}_X)_{im} = S_\ell(x_i,\bar{x}_m)\,\mathsf{e}^{\imath ku\cdot(x_i-\bar{x}_m)}$ and $(\mathsf{S}_Y^*)_{nj} = S_\ell(y_j,\bar{y}_n)\,\mathsf{e}^{-\imath ku\cdot(y_j-\bar{y}_n)}$. Apparently, the additionally required plane wave terms $\mathsf{e}^{\imath ku\cdot(x_i-\bar{x}_m)}$ and $\mathsf{e}^{-\imath ku\cdot(y_j-\bar{y}_n)}$ are separable: They define the outer products $\mathsf{P}_X = \mathsf{p}_X\,\bar{\mathsf{p}}_X^*$ and $\mathsf{P}_Y = \mathsf{p}_Y\,\bar{\mathsf{p}}_Y^*$ computed as $(\mathsf{p}_X)_i = \mathsf{e}^{\imath ku\cdot x_i}$, $(\bar{\mathsf{p}}_X^*)_m = \mathsf{e}^{-\imath ku\cdot\bar{x}_m}$ and $(\mathsf{p}_Y)_j = \mathsf{e}^{-\imath ku\cdot y_j}$, $(\bar{\mathsf{p}}_Y^*)_n = \mathsf{e}^{\imath ku\cdot\bar{y}_n}$. Thus, we write the low-rank approximant K_ℓ in the high frequency regime as

$$\mathsf{K}_\ell = (\mathsf{S}_X \circ \mathsf{P}_X)\,\bar{\mathsf{K}}\,(\mathsf{S}_Y \circ \mathsf{P}_Y)^*, \tag{2.19}$$

with the Hadamard product $\mathsf{S}_X \circ \mathsf{P}_X$ and $\mathsf{S}_Y \circ \mathsf{P}_Y$ defined as the entrywise product $(\mathsf{S}_X \circ \mathsf{P}_X)_{im} = (\mathsf{S}_X)_{im}\,(\mathsf{P}_X)_{im}$ of the matrices $\mathsf{S}_X,\mathsf{P}_X \in \mathbb{C}^{M\times L}$, analogously for $\mathsf{S}_Y \circ \mathsf{P}_Y$. Recall, the matrices P_X and P_Y do only exist in the outer product form.

The matrices $\bar{\mathsf{K}}$, S_X and S_Y are required for the representation of the low-rank matrix K_ℓ in the low and the high frequency regime, see (2.18) and (2.19), respectively. Noteworthy are two facts: On one hand they are computed identically in both frequency regimes and on the other hand the generating function for $\bar{\mathsf{K}}$ is the original, unmodified oscillatory kernel $K(x,y)$. In the high frequency regime, the plane wave vectors $\mathsf{p}_X \in \mathbb{C}^M$, $\mathsf{p}_Y \in \mathbb{C}^N$ and $\bar{\mathsf{p}}_X, \bar{\mathsf{p}}_Y \in \mathbb{C}^L$ are required additionally.

Recompression via ACA and SVD Recall, the rank of the approximant K_ℓ obtained via interpolation is denoted by L. It denotes the number of interpolation points in the clusters X and Y and is determined before computing the interpolant K_ℓ. Conversely, the low-rank of approximants computed via the SVD or ACA is denoted by r. Usually, $r \leq L$ holds for any prescribed approximation accuracy ε because r is the rank of the best, respectively near best apprixmation K_r. In the following, we present how the ACA and SVD can be used to recompress interpolants K_ℓ.

Let us concentrate on $\bar{\mathsf{K}}$ in the representations of K_ℓ in the low frequency regime (2.18) and in the high frequency regime (2.19). It is a dense matrix of the size $L\times L$ which might have a low-rank approximant $\bar{\mathsf{K}} = \bar{\mathsf{K}}_r + \bar{\mathsf{R}}_r$ satisfying $\|\bar{\mathsf{R}}_r\|_F \leq \varepsilon\,\|\bar{\mathsf{K}}\|_F$. As long as $r < L/2$ holds, the approximant $\bar{\mathsf{K}}_r$ constitutes a more efficient representation then $\bar{\mathsf{K}}$. ACA can be adopted, it leads to a low-rank approximant $\bar{\mathsf{K}}_r = \mathsf{U}\mathsf{V}^*$ of $\bar{\mathsf{K}}$ and the interpolant K_ℓ in the low frequency regime becomes

$$\mathsf{K}_\ell \sim \mathsf{K}_r = \mathsf{S}_X\,\mathsf{U}\mathsf{V}^*\,\mathsf{S}_Y^*,$$

and the interpolant in the high frequency regime becomes

$$\mathsf{K}_\ell \sim \mathsf{K}_r = (\mathsf{S}_X \circ \mathsf{P}_X)\,\mathsf{U}\mathsf{V}^*\,(\mathsf{S}_Y \circ \mathsf{P}_Y)^*.$$

Moreover, the approach we presented in section 2.2.1, which combines ACA with a subsequent application of the SVD, would lead to the best-possible low-rank approximant. In any case the accuracy $\|\mathsf{K}_\ell - \mathsf{K}_r\|_F \leq \varepsilon \, \|\mathsf{K}_\ell\|_F$ is guarateed.

Efficient interpolation on regular grids The application of the approach which combines the ACA and SVD (see sec. 2.2.1) leads to an approximant K_r of best-possible rank r, which is optimal when considering one cluster X being admissible with one cluster Y.

In the following, we consider one cluster X being admissible with a set of clusters $\{Y_t\}_{t=1}^T$. Moreover, we need to introduce an essential cluster propery: Since, we are anticipating something we will introduce in Chapter 3 we stick with the basics here. All cluster must be identical up to translations, this must hold also for orientation and distribution of interpolation points, i.e., we require a regular grid. Each admissible cluster pair X and Y_t leads to a matrix $\mathsf{K}_t \in \mathbb{C}^{M \times N_t}$ which we approximate in the low frequency regime as presented in (2.18) and in the high frequency regime as presented in (2.19). After omitting S_X, S_{Y_t} and P_X, P_{Y_t} we end up with the set of matrices $\{\mathsf{K}_t\}_{t=1}^T$, each of size $L \times L$. We can write them in a twofold way, either by organizing them as a row vector or as a column vector

$$\begin{aligned} \mathsf{K}^{(\mathrm{row})} &= [\bar{\mathsf{K}}_1, \bar{\mathsf{K}}_2, \ldots, \bar{\mathsf{K}}_t, \ldots, \bar{\mathsf{K}}_{T-1}, \bar{\mathsf{K}}_T] \\ \text{or} \quad \mathsf{K}^{(\mathrm{col})} &= [\bar{\mathsf{K}}_1; \bar{\mathsf{K}}_2; \ldots; \bar{\mathsf{K}}_t; \ldots; \bar{\mathsf{K}}_{T-1}; \bar{\mathsf{K}}_T]. \end{aligned}$$

We use the , and ; notations to distinguish column and row ordering. We chose for the set based on cluster X the row representation $\mathsf{K}^{(row)}$ (see the left setting in fig. 2.8). Any

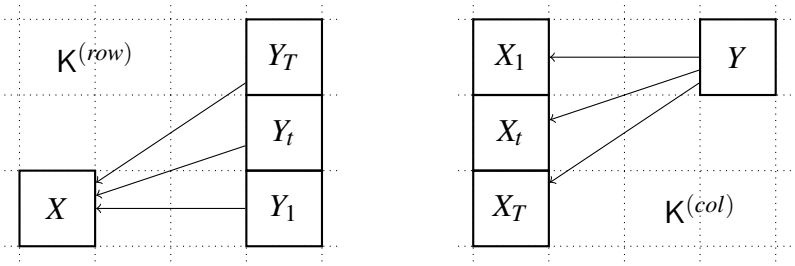


Figure 2.8: Row and column representation $\{\mathsf{K}_t\}_{t=1}^T$

matrix $\bar{\mathsf{K}}_t$ having a translation invariant kernel $K(x-y)$ and the same translation vector $c_t = c_X - c_Y$ is identical due to the regular grid. We choose the column representation $\mathsf{K}^{(col)}$ for the matrix set $\{\bar{\mathsf{K}}_t\}_{t=1}^T$ based on cluster Y (see the right setting in fig. 2.8).

Next we approximate both, $\mathsf{K}^{(row)}$ and $\mathsf{K}^{(col)}$, by using the ACA for a given accuracy ε. The resulting low-rank representations read as

$$\begin{aligned} \mathsf{K}^{(\mathrm{row})} &\sim \mathsf{U}\,[\mathsf{V}_1^*, \mathsf{V}_2^*, \ldots, \mathsf{V}_t^*, \ldots, \mathsf{V}_{T-1}^*, \mathsf{V}_T^*] \\ \text{or} \quad \mathsf{K}^{(\mathrm{col})} &\sim [\mathsf{A}_1; \mathsf{A}_2; \ldots, \mathsf{A}_t; \ldots, \mathsf{A}_{T-1}; \mathsf{A}_T]\,\mathsf{B}^*. \end{aligned}$$

Hence, for an individual matrix $\bar{\mathsf{K}}_t \sim \mathsf{U}\mathsf{V}_t^* \sim \mathsf{A}_t\mathsf{B}^*$ is true and by means of a QR decompositions we can rewrite it as

$$\bar{\mathsf{K}}_t \sim \mathsf{Q}_U\mathsf{R}_U\,(\mathsf{Q}_{V_t}\mathsf{R}_V)^* \sim \mathsf{Q}_{A_t}\mathsf{R}_A\,(\mathsf{Q}_B\mathsf{R}_B)^*. \tag{2.20}$$

Here $\mathsf{Q}_U, \mathsf{Q}_{V_t}, \mathsf{Q}_{A_t}$ and Q_B are unitary matrices. Next, we introduce $\Phi = \mathsf{R}_U\mathsf{R}_V^*$ and $\Psi = \mathsf{R}_A\mathsf{R}_B^*$ then (2.20) reads as

$$\bar{\mathsf{K}}_t \sim \mathsf{Q}_U\,\Phi\,\mathsf{Q}_{V_t}^* \sim \mathsf{Q}_{A_t}\,\Psi\,\mathsf{Q}_B^*. \tag{2.21}$$

Now, by means of (2.21) we can rearrange any matrix $\bar{\mathsf{K}}_t$ as follows

$$\begin{aligned}
\bar{\mathsf{K}}_t &\sim \mathsf{Q}_{A_t}\,\Psi\,\mathsf{Q}_B^* \\
&\sim \mathsf{Q}_{A_t}\,\Psi\,(\mathsf{Q}_B^*\mathsf{Q}_B)\,\mathsf{Q}_B^* \\
&\sim \mathsf{K}_t\,(\mathsf{Q}_B\mathsf{Q}_B^*) \\
&\sim \mathsf{Q}_U\,\Phi\,\mathsf{Q}_{V_t}^*\,(\mathsf{Q}_B\mathsf{Q}_B^*) \\
&\sim \mathsf{Q}_U\,(\mathsf{Q}_U^*\mathsf{Q}_U)\,\Phi\,\mathsf{Q}_{V_t}^*\,(\mathsf{Q}_B\mathsf{Q}_B^*) \\
&\sim \mathsf{Q}_U\,(\mathsf{Q}_U^*\mathsf{K}_t\mathsf{Q}_B)\,\mathsf{Q}_B^* = \mathsf{Q}_U\,\mathsf{C}_t\,\mathsf{Q}_B^*.
\end{aligned}$$

With $\mathsf{C}_t \in \mathbb{C}^{r\times r}$ being computed as $\mathsf{C}_t \sim \Phi\,\mathsf{Q}_{V_t}^*\,Q_B$ or $\mathsf{C}_t \sim \mathsf{Q}_U^*\,\mathsf{Q}_{A_t}\Psi$ (both expressions are equal neglecting the approximation errors) we can rewrite the set $\{\bar{\mathsf{K}}_t\}_{t=1}^T$ as

$$\mathsf{Q}_U\,[\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_t, \ldots, \mathsf{C}_{T-1}, \mathsf{C}_T]\,\mathsf{Q}_B^*$$

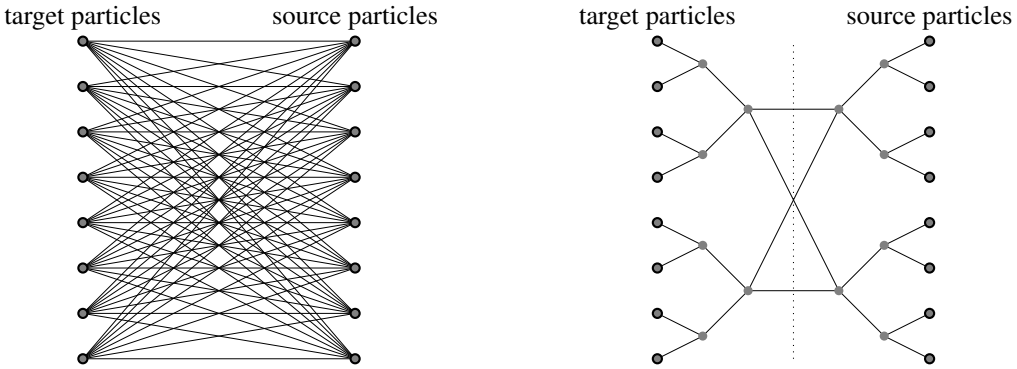
The cost of the pre-computation increases slightly due to this representation of $\bar{\mathsf{K}}_t$. However, the overall memory requirement and computational cost decreases substantially. Instead of storing $\mathsf{U}\mathsf{V}^*$, which is of order $\mathcal{O}((L+TL)r)$, only one Q_U and one Q_B per set and $\{\mathsf{C}_t\}_{t=1}^T$ need to be stored. Hence, the memory requirement reduces to $\mathcal{O}(2Lr+Tr^2)$. The computational cost must be analysed in a slightly different way. Due to the fact that we can write $\bar{\mathsf{K}}_t \sim \mathsf{U}\mathsf{V}_t^* \sim \mathsf{Q}_U\,\mathsf{C}_t\,\mathsf{Q}_B^*$ we can shift the application of Q_B and Q_U to the application of S_X and S_Y, respectively. Hence, the cost of applying $\bar{\mathsf{K}}_t$ gets reduced from $\mathcal{O}(2Lr)$ to only $\mathcal{O}(r^2)$ operations (one multiplication by C_t). Remember that this operation has to be performed for each cluster X with each in the admissible set of clusters $\{Y_t\}_{t=1}^T$. Hence, the savings are significant. A similar method is described in [42].

3 EFFICIENT EVALUATION OF PARTICLE INTERACTIONS

The simulation of physical phenomena arising from applications in science and engineering often imply the evaluation of many pairwise particle interactions as presented in fig. 3.1. Such problems can be modeled as

$$f_i = \sum_{j=1}^{N} K(x_i, y_j)\, w_j$$

for $i = 1, \ldots, M$. The source value w_j at each source particle y_j contributes to the field value f_i at each target point x_i. The contribution is described by the kernel function $K(x,y)$. As depicted in fig. 3.1a, if the contributions evaluated directly, the complexity grows quadratically. In matrix notation we can write it as matrix-vector product $\mathsf{f} = \mathsf{K}\mathsf{w}$, with the matrix $\mathsf{K} \in \mathbb{C}^{M \times N}$, the vector of source values $\mathsf{w} \in \mathbb{C}^N$ and the vector of field values $\mathsf{f} \in \mathbb{C}^M$. Plenty of research has been done to reduce the complexity of such tasks. The common approach is to exploit properties of the kernel functions. Under certain circumstances they permit to gather information of source and target particles within so called clusters what clears the way for the construction of methods to evaluate particle interactions efficiently. This idea is presented figuratively in fig. 3.1b.



(a) $\mathcal{O}(N^2)$ (b) $\mathcal{O}(N \log N)$

Figure 3.1: Particle interactions

We focus on a class of methods which can be wrapped up as *tree methods*. As the name says, these methods base on a tree representation of the domain Ω which contains source and target particles. The fundamental idea is to exploit the separability of the kernel function $K(x,y) \sim \sum u_n(x)\, v_n(y)$ and as a consequence the low-rank of the resulting matrices.

Let us explain this idea. Generally, kernel functions are asymptotically smooth, i.e., they are of the form $1/|x-y|$, where $|x-y|$ denotes the distance between source and target particle. Resulting matrices are dense and have full-rank in their entirety. In other words, they cannot be approximated by low-rank matrices (see sec. 1.2). That is why the tree representation of the domain comes into play. It permits a partitioning of the original matrix such that the near and far-field can be separated. Matrix blocks in the near-field contain singularities, hence, they effectively have full-rank and must be evaluated directly. Blocks in the far-field possess low-rank approximations and can be evaluated efficiently. Thinking of efficient matrix vector products, partitioned matrices admit the construction of an efficient matrix-vector multiplication. Instead of one large dense matrix, we have a matrix consisting of many smaller matrices, with the major part being low-rank.

We focus on two approaches, hereafter. The fast multipole method (FMM, see [50]) has become the best known representative of fast summation schemes. It is used for the simulation of many-particle interactions in computational physics, chemistry, engineering and applied mathematics, see, e.g., [51, 31, 27, 78]. The FMM leads to $\mathcal{O}(N \log N)$ and in some cases to $\mathcal{O}(N)$ algorithms by exploiting the multilevel concept. Another prominent representative are hierarchical matrices ($\mathcal{H}$-matrices, see [55, 58]). They are mostly used together with the adaptive cross approximation (ACA, see [9] and the brief introduction in sec. 2.2). They lead to $\mathcal{O}(N \log N)$ methods.

This chapter is organized as follows: First, we present construction principles which are the same for all tree methods (see sec. 3.1). In the second half, we present two representatives of the class of tree methods. We do not give a very formal description, for that we refer to the referenced literature given above. We rather concentrate on pointing out similarities and merits of either method. Both are kernel independent up to a certain stage. $\mathcal{H}$-matrices together with ACA (presented in sec. 3.2) are entirely kernel independent, hence no special treatment is needed when using them for the Helmholtz kernel. The FMM based on a Chebyshev interpolation scheme (presented in sec. 3.3) is slightly less kernel independent.

3.1 Construction principles

The construction of all tree methods is based on the same concept: The separation of near- and far-field. This objective can be split as:

1. The first task is to generate an optimal *cluster tree* representation of the domain. In sec. 3.1.1, we present the uniform and the balanced cluster tree.

2. The second task is to define suitable *admissibility criteria* which allow us to identify near- and far-field blocks. In sec. 3.1.2, we present such criteria for asymptotically smooth and for oscillatory kernel functions.

The evaluation of the near-field is trivial, all fast methods use direct evaluation. The efficient approximation of the far-field is what differs. Approaches for that are based on Taylor expansions [59, 93] or interpolation schemes [22, 45]. Other possibilities are special approximations like multipole expansions [50, 51] or their counterparts for the Helmholtz kernel [37, 5]. We focus on the kernel independent approaches we presented in sec. 2.

3.1.1 Cluster tree

We already know that the entries of the original matrix $\mathsf{K} \in \mathbb{C}^{M\times N}$ are computed as $(\mathsf{K})_{ij} = K(x_i, y_j)$ for $x_i \in X$ and $y_j \in Y$ and the root clusters X and Y cover the entire domain $\Omega \subset \mathbb{R}^d$. Consequently, the setup of a tree representation of the domain goes along with the partitioning of these root clusters. In the following, we present two tree types and use as example the root cluster X. The first one is based on an uniform subdivision of the bounding box of X. The second one is based on a balanced bisection of X. Both are hierarchical procedures, i.e., the subdivision is applied recursively to the previously obtained partitions. In each case, the result is a cluster tree of the domain being associated to the root cluster X.

In order to point out the differences of both presented types we use the set of particles, e.g., shown in fig. 3.2a.

Uniform cluster tree

In the general case, any axis parallel box which we associate to the root cluster X as

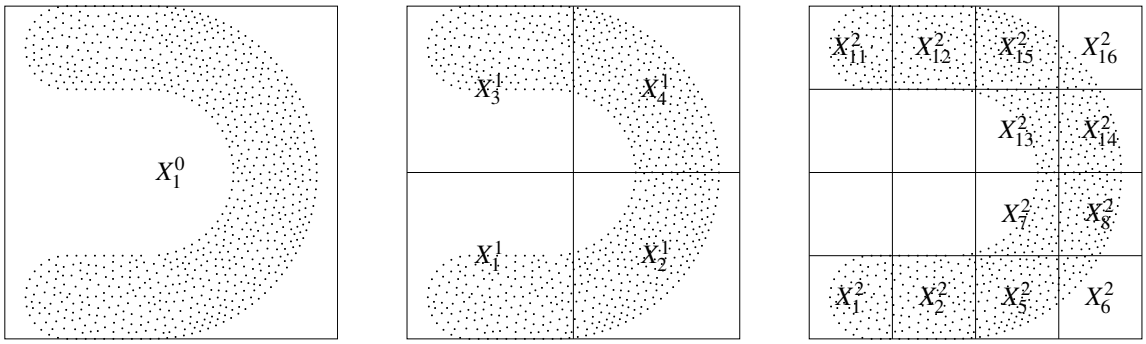
$$X = [a_1, b_1] \times \cdots \times [a_d, b_d],$$

with $\{[a_i, b_i] : i = 1, \ldots, d\}$ being closed intervals and d denoting the spacial dimension can be used as bounding box. Usually X is chosen to be the minimum bounding box containing the domain $\Omega \subset \mathbb{R}^d$.

However, here we choose a special type of bounding box for X: We define it as $[a_i, b_i] = [a, b]$ with

$$a = \min\{(x_i)_k\}_{i=1}^{M} \quad \text{and} \quad b = \max\{(x_i)_k\}_{i=1}^{N} \quad \text{for all } k = 1, \ldots, d.$$

Hence, all edges have the same length $b - a$ which corresponds to the maximum axis parallel extension of the domain. The geometric center c_X of X is not neccessarily unambiguous. This is not a problem. We fix it such that X contains the entire domain as shown in fig. 3.2a. For the sake of completeness we denote the root cluster X as X_1^0. The subscript indicates the unique index of the cluster at a given tree level which is denoted by the superscript. In this case it is about tree level 0.



(a) Level 0 (b) Level 1 (c) Level 2

Figure 3.2: Uniform partitioning

Once, the root cluster X is found its uniform oct-tree is defined. There are d separation planes along which a cluster is subdivided into 2^d equal child clusters as shown in fig. 3.2b. The separation planes are defined by their normal vectors $\{n_k \in \mathbb{R}^d : (n_k - c_X, e_k) = 0\}_{k=1}^d$ with the unit vectors $\{e_k\}_{k=1}^d$ defining the coordinate axis.

This procedure is recursively applied to all child clusters until the number of contained particles deceeds a given minimum number. If we look at fig. 3.2c, which corresponds to tree level 2, we notice that the clusters X_3^2, X_4^2, X_9^2 and X_{10}^2 do not exist. Indeed, they do not contain any particle and can be neglected in the computations.

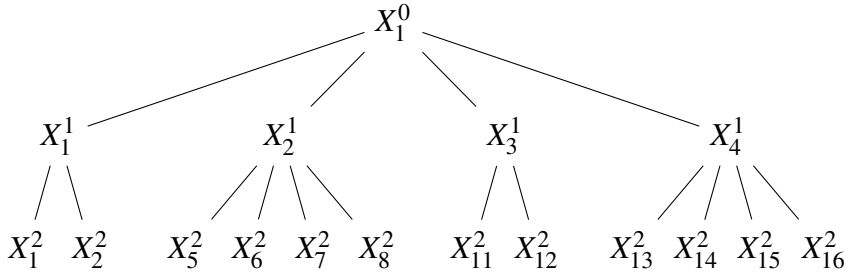


Figure 3.3: Uniform cluster tree based on fig. 3.2

A partitioning of this type leads to *quad-trees* for particle distributions in $\mathbb{R}^2$ and to *oct-trees* for particle distributions in $\mathbb{R}^3$. The tree from fig. 3.3 shows a quad-tree obtained from the exemplary particle distribution in fig. 3.2. Notice the empty, hence, not existing clusters at tree level 2.

Balanced cluster tree

Contrary to the previously introduced uniform partitioning the balanced partitioning subdivides a cluster X in two child clusters. The algorithm we use is mainly adopted from [83] and works as follows: The centroid $\hat{c}_X$ of a cluster X given by a set of particles $\{x_i\}_{i=1}^M$ is obtained by

$$\hat{c}_X = \frac{1}{M}\sum_{i=1}^{M} x_i.$$

Note, the geometric center c_X of the cluster and its centroid $\hat{c}_X$, as computed above, are normally not identical. Our goal is to bisect the cluster X in two equally balanced child clusters X_1 and X_2 (both should contain about $M/2$ particles). Such can be achived if the principal direction w of the particle cloud is known, which ends up in the solution of the following maximization problem

$$\sum_{i=1}^{M} |x_i - \hat{c}_X, w|^2 = \max_{v\in\mathbb{R}^d, |v|=1} \sum_{i=1}^{M} |x_i - \hat{c}_X, v|^2,$$

and w can be identified as the normalized eigenvector of the maximum eigenvalue of the covariance matrix $\mathsf{C} \in \mathbb{R}^{d\times d}$ defined as

$$(\mathsf{C})_{k\ell} = \sum_{i=1}^{M} (x_i - \hat{c}_X)_k (x_i - \hat{c}_X)_\ell \quad \text{for } k,\ell = 1,\ldots,d.$$

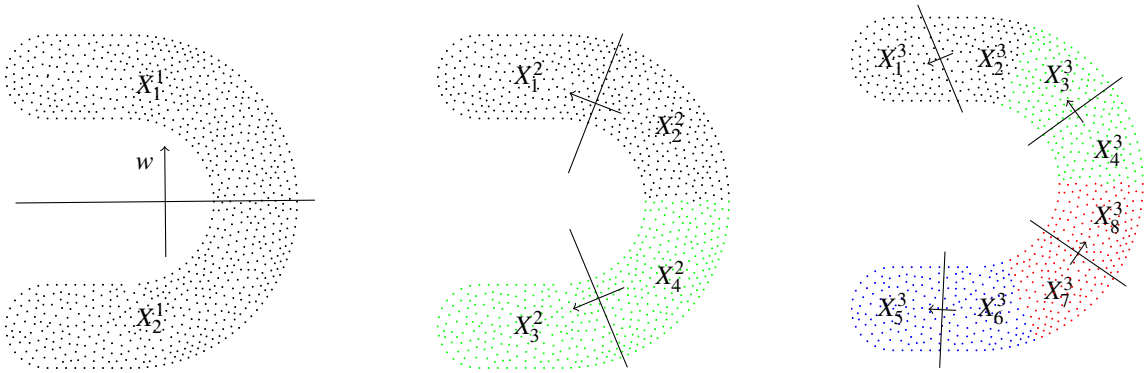
Once the centroid $\hat{c}_X$ and the principal direction w of a particle cloud is computed the child cluster affiliation is determinded by

$$(x_i - \hat{x}, w) \begin{cases} \geq 0 & x_i \in X_1, \\ < 0 & x_i \in X_2, \end{cases} \quad \text{for } i = 1,\ldots,M.$$

The hyperplane, given by $\{x \in \mathbb{R}^d : (x - \hat{c}_X, w) = 0\}$ defines the separation plane. All particles on the positive side belong to the first child cluster X_1 and all other particles belong to the second child cluster X_2.

In fig. 3.4a, the separation plane of the particle cloud is given by its normal w. In fig. 3.4b and fig. 3.4c, we show the separation at level 2 and 3 respectively. In fig. 3.4c, it can clearly be observed that the clusters X_1^3, X_2^3 and X_5^3, X_6^3 are no more symmetric. This comes due to the fact that we are looking for equally balanced child clusters and not for uniformly looking ones.

Figure 3.5 shows the cluster tree obtained via the principal direction based bisection. Contrary to the uniform partitioning, we obtain equally balanced *binary-trees*. The recursive bisection procedure ends once the number of contained particles in a cluster deceeds a given minimum number.



(a) Level 1 (b) Level 2 (c) Level 3

Figure 3.4: Balanced partitioning

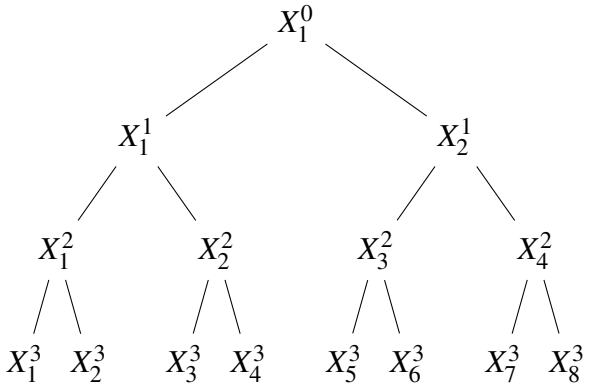


Figure 3.5: Balanced cluster tree based on fig. 3.4

Comparing uniform and balanced cluster trees

There are two decisive differences. We illustrate them in the following and explain why uniform trees are the better choice for interpolation based approximations and why balanced trees is the way to go if ACA is used.

Balance Let us define the balance at a given level as the ratio between the cluster containing the fewest and the cluster containing the most particles. Hence, if this ratio approaches 1 the tree is perfectly balanced. In Tab. 3.1, we compare the balance of both appoaches. If

Level	uniform tree	balanced tree
1	0.683	0.984
2	0.192	0.928
3	-	0.920

Table 3.1: Comparison of cluster tree balance based on the figures 3.2 and 3.4

we look at fig. 3.2 or 3.3, we notice that certain clusters contain no particles, i.e., they are empty and can be neglected in any computation. That is why we also neglected them in the evaluation of the cluster tree balance of the uniform cluster tree in Tab. 3.1.

Uniformity The uniform subdivision leads to an entirely regular clusters grid. Each cluster corresponds to an axis parallel box whose edges have the same length, which is twice the the length of its child box edge. All clusters at a given level are identical up to translations. Such cluster trees allow us to apply the efficient treatment of interpolants as presented in sec. 2.3.3.

3.1.2 Admissibility criteria

Once, we have tree representations of the root clusters the further tasks are twofold: First, we need to find submatrices of the original matrix and, second, we need to identify them as being admissible or not. Therefore, we introduce the concept of cluster pairs:

Definition 5. If the entries of the matrix $\mathsf{K} \in \mathbb{C}^{M\times N}$ are computed as

$$(\mathsf{K})_{ij} = K(x_i, y_j) \quad \text{with } x_i \in X,\ y_j \in Y,$$

then K is associated to the cluster pair $X \times Y$, with the clusters $X, Y \subset \mathbb{R}^d$ containing the particles $\{x_i\}_{i=1}^{M}$ and $\{y_j\}_{j=1}^{N}$, respectively.

Now, we have the tool to approach the first goal, i.e., we can define submatrices of the original matrix. Next, we need to identify admissible cluster pairs which are associated to potential low-rank submatrices of the original matrix. Therefore, we introduce the concept of admissibility criteria as a tool to a priori identify submatrices having low-rank representations:

Definition 6. If the cluster pair $X \times Y$ satisfies given admissibility criteria then it is admissible and the associated matrix K has a low-rank representation K_r of rank r. The admissibility of cluster pairs guarantees that the remainder $\|\mathsf{K} - \mathsf{K}_r\|_F$ decays exponentially as r increases.

Finally, with the admissibility criteria we have the last tool to separate near- and far-field. All matrices which are associated to admissible cluster pairs are assigned to the far-field $\mathcal{F}$ and all other matrices are assigned to the near field $\mathcal{N}$. Recall, all matrices in $\mathcal{N}$ must be evaluated directly and all matrices in $\mathcal{F}$ can be approximated efficiently. In the following, we present such admissibility criteria for asymptotically smooth and oscillatory kernel functions.

Asymptotically smooth kernels One admissibility criterion is needed for asymptotically smooth kernels: the *well separation criterion* which reads as

$$w \leq \eta\, |c|, \tag{3.1}$$

where $\eta > 0$ is the well known admissiblity parameter for asymptotically smooth kernels [21, 9, 12, 56]. The cluster size $w = \min\{\mathrm{diam}(X), \mathrm{diam}(Y)\}$ and distance $|c|$ are computed depending on the type of cluster tree we use (see sec. 3.1.1). If we use uniform cluster trees, the diameter is simply the length of a box edge, it is identical for each cluster at a given level in the cluster tree. The distance is computed as $c = c_X - c_Y$. If we use balanced cluster trees, we compute the diameters as $\mathrm{diam}(X) = 2\max\{|\hat{c}_X - x_i|\}_{i=1}^{M}$ and $\mathrm{diam}(Y) = 2\max\{|\hat{c}_Y - y_j|\}_{j=1}^{N}$, and the distance as $c = \hat{c}_X - \hat{c}_Y$.

Figure 3.6 shows admissible cluster pairs based on a uniform cluster tree: The cluster pairs $X \times Y_1$, $X \times Y_2$ and $X \times Y_3$ are admissible. Touching clusters pairs are not admissible.

Oscillatory kernels Oscillatory kernels require two bounds, see (2.4): The *minimum separation criterion* and the *maximum cone aperture criterion*. Remember, a cone is defined by the unit vector u, k is the wave number. Both criteria read as

$$kw^2 \leq \gamma |c| \qquad \text{and} \qquad \alpha_u^c \leq \frac{\gamma}{kw} \tag{3.2}$$

where the cluster size w and the distance c are computed as previously introduced. The cone aperture is computed as $\alpha_u^c = 2\,|{}^c/_{|c|} - u|$. The constant γ has been introduced in (2.4).

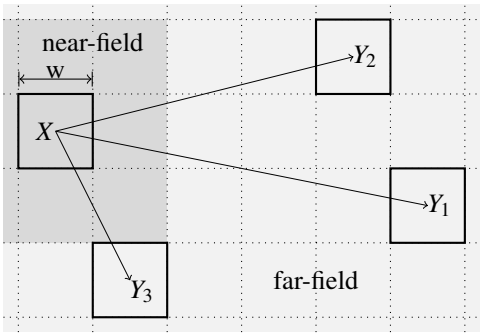


Figure 3.6: Admissibile cluster pairs for asymptotically smooth kernels

Figure 3.7 shows a uniform cluster grid. The dashed circle represents the minimum separation of $\mathcal{O}(kw^2)$ around X. The vector u points from the center of X in the direction of the cone of maximum aperture $\mathcal{O}(1/kw)$. All clusters in this cone satisfy both admissibily criteria in (3.2) with respect to cluster X and direction u. Obviously, the clusters Y_2 and Y_3 do not satisfy the cone aperture criterion for the cone of direction u. They are admissible to X for cones having other directions.

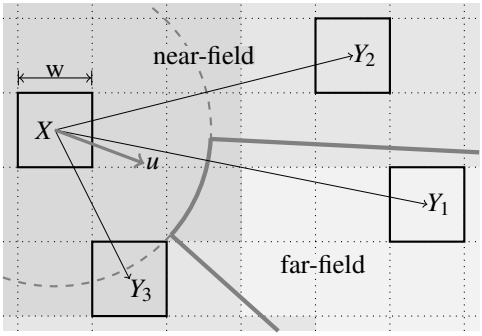


Figure 3.7: Admissible cluster pairs for oscillatory kernels

3.2 Hierarchical Matrices

A formal description of hierarchical matrices ($\mathcal{H}$-matrices) was first given in [55]. Important textbooks are [56] and [12]. Here, we present a rather intuitive view on $\mathcal{H}$-matrices. Our goal is to point out the simplicity of using them and not to thoroughly and gaplessly describe them. Hence, our notation will be simplified for the sake of clarity and, hopefully, the context does not allow any misunderstandings.

At the beginning of this chapter we claimed that fast methods share the same idea, which is the separation of near- and far-field. So do $\mathcal{H}$-matrices. First, we explain which strate-

gies we deploy to approximate the far-field. Then, based thereon, we explain how the partitioning of the original matrix takes place.

3.2.1 Construction principles

Fundamental for the construction of $\mathcal{H}$-matrices are the cluster tree type and the admissibility criteria. Once they are determined, the near-field can be separated from the far-field which, eventually, can be approximated. Even though, the construction follows this order, we first determine the approximation method. Based thereon, we specify the cluster tree type and the admissibility criteria and construct the $\mathcal{H}$-matrix.

Adaptive cross approximation We deploy ACA to approximate the far-field. In order to exploit its merits we make use of balanced cluster trees (see sec. 3.1.1) and the admissibility criterion for asymtotically smooth kernel functions (see sec. 3.1.2). In the following, we motivate our choice and support it by means of the examples 5 and 6.

Balanced cluster trees Equally balanced cluster trees imply that the ratio of the smallest and the largest cluster at a given level approaches 1 (see tab. 3.1). Hence, balanced cluster trees have smaller depths than uniform cluster trees. This, obviously, leads to fewer and larger submatrices in the far-field. Since the efficiency of ACA increases with the size of the matrix and it does not require any uniformity, balanced cluster trees are the way to go when using ACA.

Admissibility criteria for asymptotically smooth kernels The ACA algorithm can be augmented [12] such that every computed approximant converges to the prescribed accuracy. Let us explain this with other words. If a matrix has full-rank the approximant computed with ACA has full-rank, too. No matter whether we treat smooth or oscillatory kernels or even kernels featuring singularities, ACA computes the almost optimal approximant (see sec. 2.2). Nevertheless, we use the admissibility criterion from (3.1) to exclude true full-rank blocks from the far-field, in order not to uselessly try to approximate them.

3.2.2 Separation of near- and far-field

The block-cluster-tree is the commonly used concept to describe the structure of $\mathcal{H}$-matrices. It stores information about near-field and far-field in a hierarchical manner. We could also use the term tree of cluster-pairs $X \times Y$ with the cluster-trees X and Y.

To make the idea clearer we present in Algorithm 1 the setup of an $\mathcal{H}$-matrix. It takes as argument a cluster-pair $X \times Y$ and checks whether it is admissible or not. If it is, the associated matrix is added to the far-field. Otherwise, the admissibility of all child cluster-pairs $X_{\text{child}} \times Y_{\text{child}}$ is checked recursively, unless X and Y are already leafs.

Initially, we pass the root cluster-pair $X^0 \times Y^0$ to the algorithm which, normally, does not satisfy admissibility. This is, when the hierarchical construction of the block-cluster-tree starts. The union of all admissible cluster-pairs forms the far-field and the union of all non-admissible cluster-pairs forms the near-field.

Algorithm 1 Construction of block-cluster-tree from cluster a cluster-pair $X \times Y$

```
1:  procedure BLOCK CLUSTER TREE(X × Y)
2:      if clusters X and Y are admissible then
3:          add cluster-pair X × Y to far-field
4:      else if X and Y are leaf clusters then
5:          add cluster-pair X × Y to near-field
6:      else
7:          for all X_child ⊂ X and Y_child ⊂ Y do
8:              BLOCK CLUSTER TREE(X_child × Y_child)
9:          end for
10:     end if
11: end procedure
```

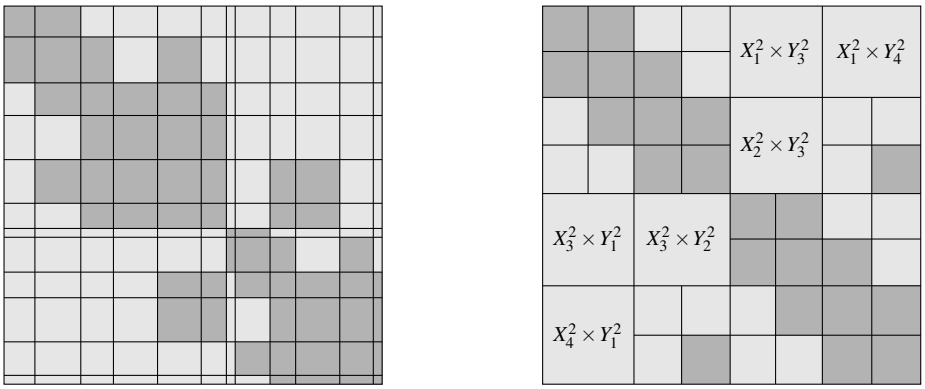
A formal and complete description of the setup of block-cluster-trees can be found in [55, 58, 56].

Different block-cluster-trees Recall, we have chosen balanced cluster-trees here. The following example shows that other types of cluster-trees lead to differently looking block-cluster-trees.

Example 5. We use Algorithm 1 and let the root clusters $X^0 = Y^0$ be identical. Then, we compare the resulting block-cluster-trees for a uniform and a balanced cluster-tree in fig. 3.8. As minimum separation criterion, we require that admissible clusters are not allowed to touch themselves. Anytime a cluster-pair does not satisfy this criterion its child cluster-pairs are checked recursively for admissibility until the criterion is fulfilled or the leaf-level of the cluster-trees is reached.

- We use the uniform cluster-tree from fig. 3.3. Apparently, no admissible cluster-pairs exist a level 0 and 1, only at level 2 such pairs can be found: For example cluster X_{13}^2 is admissible to the clusters Y_1^2, Y_2^2, Y_5^2, Y_6^2 and Y_{11}^2. It is not admissible to the clusters Y_7^2, Y_8^2, Y_{12}^2, Y_{13}^2, Y_{14}^2 and Y_{16}^2, because they touch themselves. The resulting block-cluster-tree is displayed in fig. 3.8a.

- We use the balanced cluster-tree from fig. 3.5. No admissible cluster pairs exist at level 1. There exist some at level 2, e.g., cluster X_1^2 is admissible to the clusters Y_3^2 and Y_4^2 but it is not admissible to the cluster Y_2^2. For each not admissible cluster-pair its child cluster-pairs are checked recursively. We end up with the block-cluster-tree in fig. 3.8b.



(a) Uniform cluster-trees (b) Balanced cluster-trees

Figure 3.8: Block-cluster-trees for different types of cluster-trees

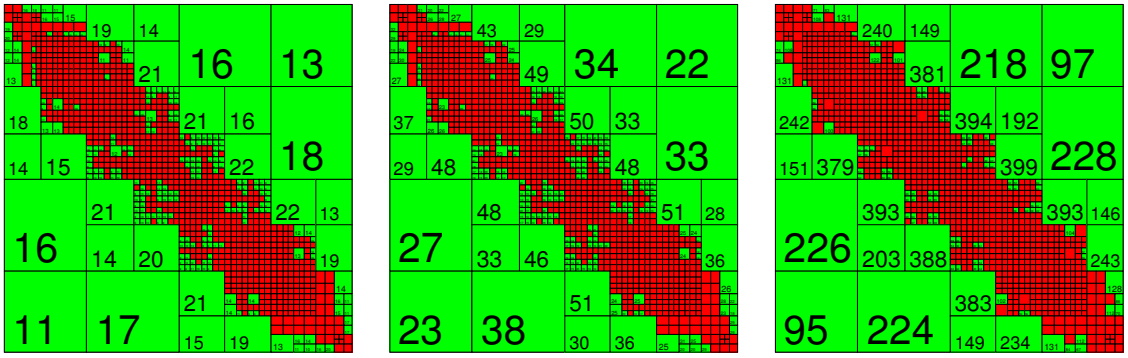
Let us compare the block-cluster-trees from fig. 3.8. The ratio of far- and near-field is nearly the same. However, the right block-cluster-tree, which is based on the balanced cluster-tree, contains fewer, but larger blocks compared to the left one. Considering the efficiency of the far-field approximation this fact is not irrelevant. If we assume that each far-field block has a low-rank representation of approximately the same rank, a balanced cluster-tree leads to a more efficient $\mathcal{H}$-matrix structure in terms of computational cost and memory requirement.

Remark. The matrix partitions shown in fig. 3.8 cannot be seen as a general conclusion to any problem geometry and size. It should give an idea about the stucture of an $\mathcal{H}$-matrix depending on the cluster-tree type.

Influence of the wavenumber Since we use the admissiblity criterion from (3.1), the structure of the resulting $\mathcal{H}$-matrix does not depend on the wavenumber k. In this sense, the admissibility criteria for oscillatory kernels from (3.2), which dependends on k, are not satisfied. This comes along with an increasing near-field and an increasing low-rank of far-field blocks as k increases. We explain the behavior in the following.

Example 6. We randomly distribute points on the surface of a prolate spheroid of extension d in x_1 direction and $d/10$ in x_2 and x_3 direction. We choose $d = 16$ and get 7667 points being associated to the root clusters X and Y. Then, we setup balanced cluster-trees and the $\mathcal{H}$-matrix structure based on the admissibility criterion from (3.1). At last, we evaluate all blocks based on the oscillatory kernel (2.1) for the wavenumbers $k = 1, 10, 100$. Far-field

blocks are approximated using ACA of accuracy $\varepsilon_{\text{ACA}} = 10^{-4}$. The resulting $\mathcal{H}$-matrices are displayed in fig. 3.9.



(a) $k = 1$ (b) $k = 10$ (c) $k = 100$

Figure 3.9: Same $\mathcal{H}$-matrix structure but different wavenumber k

Since the structure of the $\mathcal{H}$-matrix does not depend on the wave number, we expect the near- and far-field to be identical in all cases shown in fig. 3.9. However, this is no longer the case. High wave numbers imply highly oscillatory kernels which require larger ranks to satisfy prescribed accuracies. We know that ACA is able to handle this. However, a low-rank representation of the form UV^* is only more efficient then a full-rank representation $\mathsf{K} \in \mathbb{C}^{M \times N}$ as long as the condition $r(M+N) < MN$ holds (see sec. 2). This is the reason why the near-field increases with increasing wave numbers (think of the near-field as the union of full-rank blocks).

3.2.3 Applications

The whole $\mathcal{H}$-matrix concept is not restricted to fast matrix-vector products only. Actually, the only difference between $\mathcal{H}$-matrices and ordinary matrices is the hierarchical block partitioning. Each block stores its entries, either in full- or low-rank representation. In this sense, $\mathcal{H}$-matrices can be seen as ordinary matrices. All matrix operations, such as the computation of matrix norms, matrix-vector and matrix-matrix multiplication, addition and subtraction of matrices, matrix inversion, *LU* and *QR* decompositions, and so on, can be performed efficiently, and are based on a prescribed accuracy. For the derivation and complexity analysis af the entire suite of $\mathcal{H}$-matrix arithmetics we refer to [55, 58, 56, 12]. For example, a preconditioning strategy is presented in [10]. The idea is that hierarchical inversion or *LU* decomposition of rather low accuracy leads to effcient preconditioners. Parallelization strategies for $\mathcal{H}$-matrix arithmetics are presented in [14].

3.3 Directional Fast Multipole Method

The fast multipole method (FMM), invented by Greengard and Rokhlin [50], was one of the first algorithm which allowed to overcome one of the biggest headaches of N-body problems as claimed in [33]: "The fact that accurate calculation of the motions of N particles interacting via gravitational or electrostatic forces would seem to require $\mathcal{O}(N^2)$ operations. The fast multipole algorithm get by with $\mathcal{O}(N)$ operations." We refer the reader who is unfamiliar with the FMM to the introductory tutorial given in [66].

Our approach is based on the ideas published in [42, 88]. We extended their scheme to oscillatory kernels by exploiting the idea presented in [40]. Many concepts we present in this section have already been published in [76], we will elaborate them in more detail here. We follow the same structure as in the previous section. First, we present the construction principles and motivate them. Then, we derive the directional fast multipole method for oscillatory kernels.

3.3.1 Construction principles

Fundamental for any FMM are the specification of cluster-tree type and admissibility criteria, then, the method can be constructed. The choice of these principles depends on the approximation scheme we utilize for the far-field.

Chebyshev interpolation For the approximation of far-field blocks we adopt the Chebyshev interpolation scheme as presented in sec. 2.3. In order to exploit the merits of this scheme we use uniform cluster-trees we elaborated in sec. 3.1.1. All approaches which base on a separable kernel approximation require tight admissibility criteria in order to guarantee prescribed accuracies. So does the Chebyshev interpolation: We have to judiciously choose between the admissibility criteria from (3.1) and from (3.2). In the following we illustrate how we do that.

Uniform cluster trees Whenever we use an interpolation based approximation, uniform cluster-trees are the more beneficial choice. Let us explain three decisive advantages.

1. We construct the interpolation nodes $\bar{x}$ and $\bar{y}$ in each cluster via a tensor product rule as presented in (2.10). If we use uniform cluster-trees the interpolation nodes of all clusters at a given tree level remain identical up to translations.
2. The matrix $\bar{\mathsf{K}}$ in (2.18) results from evaluations of a translation invariant kernel $K(\bar{x}-\bar{y})$ at interpolation nodes $\{\bar{x}\} \subset X$ and $\{\bar{y}\} \subset Y$. It stays constant whenever the translation vector $c = c_X - c_Y$ is the same. Hence, on a uniform cluster grid, the number of different matrices $\bar{\mathsf{K}}$ depends only on the maximal possible number of

different translation vectors. This number can be bounded by the cluster size w and the wave number k. On a not uniform grid the number of kernel evaluations would grow with the size of the problem.

3. This point follows from the previous one. The recompression of $\bar{\mathsf{K}}$, as presented in sec. 2.3.3, can only be used on uniform cluster grids. The reason becomes clear in fig. 2.8. $\bar{\mathsf{K}}_t$ is only identical for both cluster pairs $X \times Y_t$ and $X_t \times Y$, if they live on a regular grid.

Single-level scheme

Given is a uniform cluster tree of the interval $[a,b] \in \mathbb{R}$ having 4 levels (in a single-level scheme only the leaf level is considered). The objective is the construction of a fast single-level summation scheme for the sum in (3.3). Considering a single-level scheme, the leaf-level consists of 16 clusters which are associated to $\{X_I\}_{I=1}^{16}$ and $\{Y_J\}_{J=1}^{16}$. Let us evaluate the sum for an arbitrary X_I with $I \in [1,16]$. In our example we chose $I = 7$ as shown in fig. 3.10. First, we identify all cluster $\{Y_{J^F}\}$ which are well separated, hence,

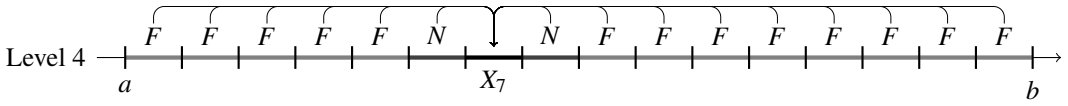


Figure 3.10: Fast single-level summation in the interval $[a,b]$

admissible to X_I. In fig. 3.10 the far-field is denoted by F and the near-field by N. The clusters $\{Y_{J^N} : J^N \in [I-1, I+1]\}$ are neighbors, all others are admissible to X_I. In the FMM notation non admissible clusters form the *neighbor list* and all admissible clusters form the *interaction list*. The union of all interaction lists forms the far-field and the union of all neighbor lists the near-field.

Using a single-level summation scheme the contribution from non-admissible clusters $\{Y_{J^N}\}$ is computed directly

$$f_i^N = \sum_{j \in J^N} K(x_i, y_j)\, w_j$$

and from admissible clusters $\{Y_{J_F}\}$ approximately by interpolating the kernel function

$$f_i^F \sim \sum_{m \in \alpha} S_\ell(\bar{x}_m, x_i) \sum_{n \in \alpha} K(\bar{x}_m, \bar{y}_n) \sum_{j \in J^F} S_\ell(\bar{y}_n, y_j)\, w_j$$

with the multi-index $|\alpha| \leq (\ell+1)^d$. The above equation can be computed efficiently by splitting it up in the following three steps

1. Moment to moment operator (M2M): Equivalent source values are computed at the interpolation points $\bar{y}_n \in Y_{J^F}$ by $W_n = \sum_{j \in J^F} S(\bar{y}_n, y_j)\, w_j$.

2. Moment to local operator (M2L): Field values are computed at the interpolation points $\bar{x}_m \in X_I$ by $F_M = \sum_{n \in \alpha} K(\bar{x}_m, \bar{y}_n) W_n$.
3. Local to local operator (L2L): Field values are computed at final points $x_i \in X_I$ by $f_i^F \sim \sum_{m \in \alpha} S(\bar{x}_m, x_i) F_m$.

Finally, near-field and far-field contributions are summed up as $f_i \sim f_i^N + f_i^F$.

Multilevel scheme

Conversely to single-level schemes, far-field contributions are gathered at multiple levels, when using multilevel schemes. Let us explain this. Given is a uniform cluster tree of the interval $[a,b]$ as shown in fig. 3.11. The near-field of X_7 is the same for both, single- and

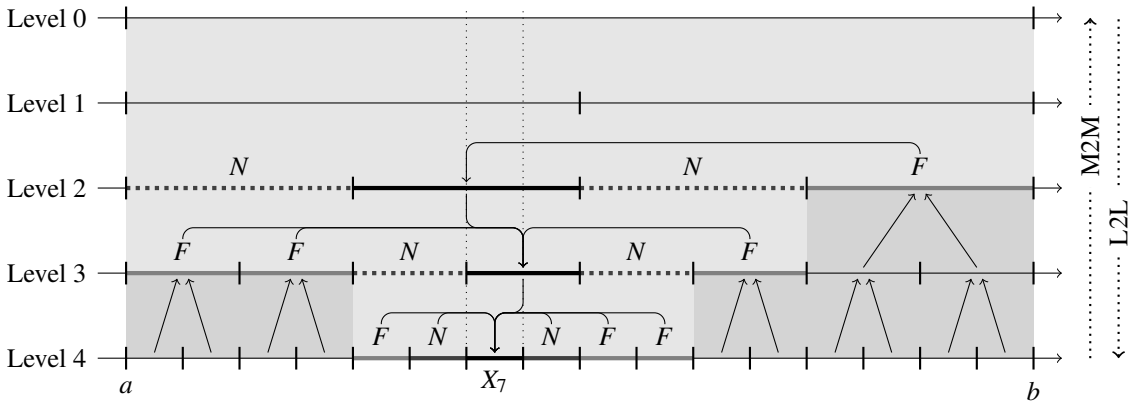
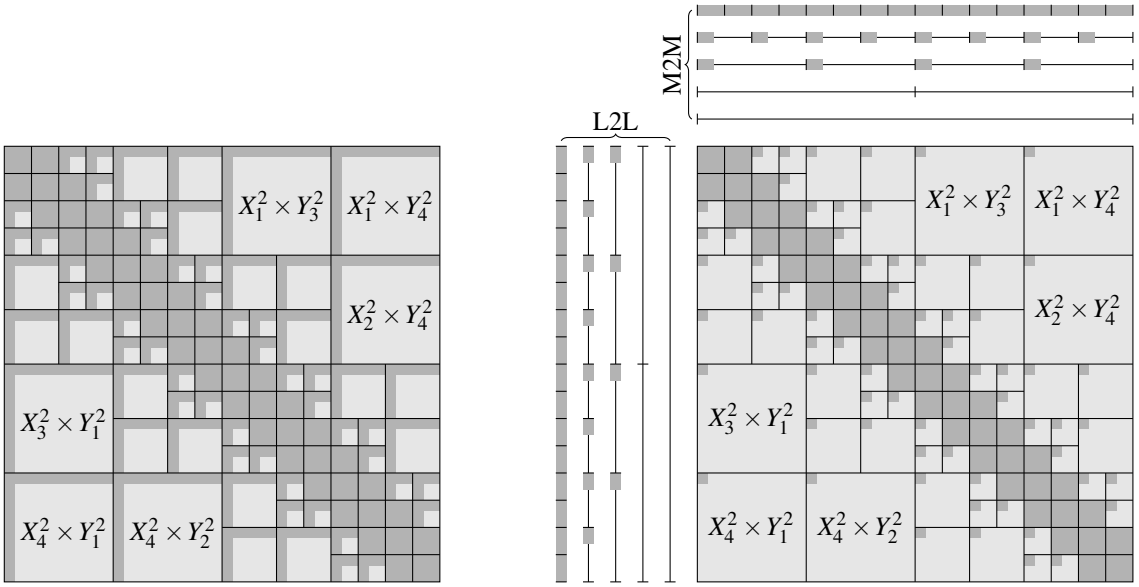


Figure 3.11: Fast multilevel summation in the interval $[a,b]$

multilevel schemes. However, the far-field is evaluated differently. Let us start with setup of the cluster relations starting from the root level. In level 0 and 1 exist no admissible cluster-pairs. Well separated cluster-pairs exist from level 2 on. The difference to single-level schemes comes now: If we step down to level 3 we do only look for admissible clusters pairs in the near-field of the level 2. We repeat this all the way down to the leaf level.

There is no difference to $\mathcal{H}$-matrices, yet. The near-field and the setup of the far-field is identical. The evaluation of the far-field is different in multilevel schemes. We consider fig. 3.11 in the following explanation. First, starting from the leaf level, M2M operators evaluate equivalent sources W at higher levels. Second, M2L operators gather their contribution at all levels. Third, starting from the highest level having admissible clusters, L2L compute field values F at the next lower levels. Moreover, in each level the contribution of admissible clusters is added. At the end the near-field contributions are evaluated directly. The multilevel scheme is elaborated in more detail in sec. 3.3.5.

Matrix structure Let us compare the structure of an $\mathcal{H}$-matrix and a multilevel scheme. Based on the model domain $[a,b]$ used above we schetch the resulting structures in fig. 3.12. Figure 3.12a stems from the $\mathcal{H}$-matrix and fig. 3.12b from the multilevel approach. We as-



(a) $\mathcal{H}$-matrix (ACA) (b) Multilevel scheme (Chebyshev interpolation)

Figure 3.12: Comparison of matrix structures

sume that N points are distributed equidistantly in $[a,b]$. Both matrix structures in fig. 3.12 store the same near-field, i.e., $23/128N^2$ entries. Only the far-field structure differs: Let us adopt the matrix notations for far-field block as introduced in sec. 2: for ACA we use the UV^* and for the Chebyshev interpolation the $\mathsf{S}_X\bar{\mathsf{K}}\mathsf{S}_Y^*$ representation. Moreover, we assume a constant rank r approximation in both cases. The far-field of the $\mathcal{H}$-matrix approach uses in total $12.75Nr$ entries. The far-field of the multilevel approach is syntesized along multiple levels, as depicted in fig. 3.11. The M2L operators $\bar{\mathsf{K}}$, denoted by little dark-gray squares in fig. 3.12b, use $66r^2$ entries. The M2M and L2L operators, which correspond to S_Y and S_X, respectively, use $Nr+24r^2$ entries each. This sums up to $Nr+90r^2$ entries compared to $12.75Nr$ required for the $\mathcal{H}$-matrix.

In a formal complexity analysis the cost of $\mathcal{H}$-matrices approaches grows like $\mathcal{O}(N\log N)$ and the cost or multilevel schemes grows only like $\mathcal{O}(N)$. Representatives are fast multipole methods and their algebraic counterparts $\mathcal{H}^2$-matrices [57, 19].

3.3.2 Directional summation scheme

Two frequency regimes are of interest when dealing with oscillatory kernels as we introduced in Definition 4. We need to choose the correct admissiblity criteria for each regime in order to guarantee a prescribed approximation accuracy. The separation of near- and far-field follows the idea of the hierarchical partitioning used for $\mathcal{H}$-matrices: If a cluster-pair is admissible it is added to the far-field, else the admissibility of their child cluster-pairs is checked. The recursive procedure stops at the leaf-level of the cluster-trees. The data organisation, however, is somewhat different when constructing multilevel schemes.

Consider the matrix vector product $\mathsf{f} = \mathsf{K}\mathsf{w}$ with $\mathsf{K} \in \mathbb{C}^{N\times N}$ written as a direct summation

$$f_i = \sum_{j=1}^{N} K(x_i, y_j)\, w_j \quad \text{for } i = 1,\dots,M \tag{3.3}$$

with all $x_i \in X$ and all $y_j \in Y$. The objective is to construct a fast summation scheme where the kernel functions $K(x,y)$ are oscillatory. The scheme relies fundamentally on the concept of interpolation. However, the actual calculation involves two different but similar looking operators: the anterpolation and interpolation operators. In the following, we present the tripartite scheme for the low and the high frequency regime.

Low frequency regime A cluster-pair $X \times Y$ in the low frequency regime is admissible if it satisfies the *well separation criterion* from (3.1). In that case we can interpolate the kernel as

$$K(x,y) \sim \sum_{m\in\alpha} S_\ell(x,\bar{x}_m) \sum_{n\in\alpha} K(\bar{x}_m,\bar{y}_n)\, S_\ell(y,\bar{y}_n),$$

and insert it in (3.3) and obtain

$$f_i \sim \sum_{m\in\alpha} S_\ell(x,\bar{x}_m) \sum_{n\in\alpha} K(\bar{x}_m,\bar{y}_n) \sum_{j=1}^{N} S_\ell(y,\bar{y}_n)\, w_j \quad \text{for } i = 1,\dots,N.$$

We can efficiently compute the above summation by splitting up this equation in three tasks:

1. M2M: Compute equivalent sources at interpolation points $\bar{y}_n$ by anterpolation

$$W_n = \sum_{j=1}^{N} S_\ell(y,\bar{y}_n)\, w_j \quad \text{for } n \in \alpha.$$

2. M2L: Compute field values at interpolation points $\bar{x}_M$ with the kernel function

$$F_m = \sum_{n\in\alpha} K(\bar{x}_m,\bar{y}_n)\, W_n \quad \text{for } m \in \alpha.$$

3. L2L: Compute field values at final points x_i by interpolation

$$f_i \sim \sum_{m\in\alpha} S_\ell(x,\bar{x}_m)\, F_m \quad \text{for } i = 1,\dots,M.$$

High frequency regime A cluster-pair $X \times Y$ in the high frequency regime is admissible if the *minimum separation criterion* and the *maximum cone aperture criterion* from (3.2) are satisfied. By using the directional low-rank property of $K^u(x,y) = \mathrm{e}^{\imath k(|x-y|-u\cdot(x-y))}$ we can then interpolate the oscillatory kernel $K(x,y)$ as

$$K(x,y) \sim \mathrm{e}^{\imath k u\cdot x} \sum_{m\in\alpha} S_\ell(x,\bar{x}_m)\, \mathrm{e}^{-\imath k u\cdot \bar{x}_m} \sum_{n\in\alpha} K(\bar{x}_m,\bar{y}_n)\, \mathrm{e}^{\imath k u\cdot \bar{y}_n}\, S_\ell(y,\bar{y}_n)\, \mathrm{e}^{-\imath k u\cdot y}. \tag{3.4}$$

We insert it in (3.3) and obtain

$$f_i \sim \mathrm{e}^{\imath k u\cdot x_i} \sum_{m\in\alpha} S_\ell(x_i,\bar{x}_m)\, \mathrm{e}^{-\imath k u\cdot \bar{x}_m} \sum_{n\in\alpha} K(\bar{x}_m,\bar{y}_n)\, \mathrm{e}^{\imath k u\cdot \bar{y}_n} \sum_{j=1}^{N} S_\ell(\bar{y}_n,y_j)\, \mathrm{e}^{-\imath k u\cdot y_j}\, w_j$$

for $i = 1,\dots,M$ and for $j = 1,\dots,N$. We can efficiently compute the summation by splitting the above equation in three tasks:

1. Directional M2M: Compute equivalent sources at interpolation points $\bar{y}_n$ by anterpolation

$$W_n^u = \mathrm{e}^{\imath k u\cdot \bar{y}_n} \sum_{j=1}^{N} S_\ell(\bar{y}_n,y_j)\, \mathrm{e}^{-\imath k u\cdot y_j}\, w_j \quad \text{for } n \in \alpha. \tag{3.5}$$

2. M2L: Compute field values at interpolation points $\bar{x}_m$ with the kernel function

$$F_m^u = \sum_{n\in\alpha} K(\bar{x}_m,\bar{y}_n)\, W_n^u \quad \text{for } m \in \alpha.$$

3. Directional L2L: Compute field values at final points x_i by interpolation

$$f_i \sim \mathrm{e}^{\imath k u\cdot x_i} \sum_{m\in\alpha} S_\ell(x_i,\bar{x}_m)\, \mathrm{e}^{-\imath k u\cdot \bar{x}_m}\, F_m^u \quad \text{for } i = 1\dots,M.$$

Here, α is the multi-index of the interpolation points $\{\bar{x}_m\}_{m\in\alpha}$ and $\{\bar{y}_n\}_{n\in\alpha}$ in the clusters X and Y, respectively. If ℓ denotes the interpolation order, then $|\alpha| \le L = (\ell+1)^d$ in $\mathbb{R}^d$. The steps 1 and 3 in both regimes require $\mathcal{O}(LN)$ and $\mathcal{O}(LM)$ operations, respectively. Step 2 requires $\mathcal{O}(L^2)$ operations. Hence for $L \ll N$ and M, this summation algorithm scales like $\mathcal{O}((M+N)L)$ in contrast to $\mathcal{O}(MN)$ of a direct evaluation.

Even though, the M2M and L2L operators differ (they are transpose of each other), both are derived from the interpolation scheme presented in sec. 2.3. Thus, for the error analysis of these translation operators only errors introduced by the interpolation algorithm need to be studied. Regarding this, the anterpolation (M2M) and interpolation steps (L2L) behave identically. Let us introduce the directional partitioning of the far-field and, then, study the directional translation operators.

3.3.3 Directional far-field partitioning

Due to the directionality of (3.4) we need to partition the far-field in the high frequency regime into a set of cones given by the unit vectors $\{u_c\}_{c=1}^{C}$, where C denotes the number of cones. Every cone has an aperture of $\mathcal{O}(1/kw)$. The partitioning in $\mathbb{R}^2$ is trivial. Let us focus on the partitioning in $\mathbb{R}^3$. We follow the approach presented in [40].

Initialization To ease the understanding of the procedure we introduce an indexing of the tree levels in the high frequency regime. We start with the first level in the high frequency regime. We give it the index $h = 0$ and partition the entire far-field of all its clusters $X \subset \mathbb{R}^3$ in six root cones. We define them by $u_1 = (1,0,0)^\top$, $u_2 = (0,1,0)^\top$, $u_3 = (0,0,1)^\top$, $u_4 = (-1,0,0)^\top$, $u_5 = (0,-1,0)^\top$ and $u_6 = (0,0,-1)^\top$.

For example the first root cone which is defined by u_1 and displayed in fig. 3.13 contains all points whose x_1 coordinate is positive and greater than the absolute values of the x_2 and x_3 coordinates. Let us define the opening angles

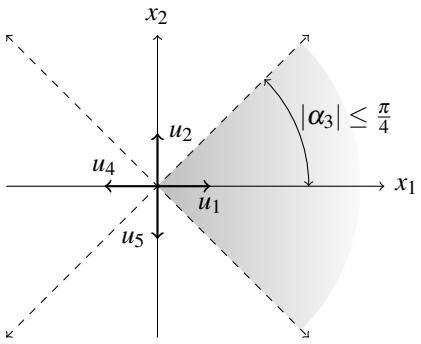


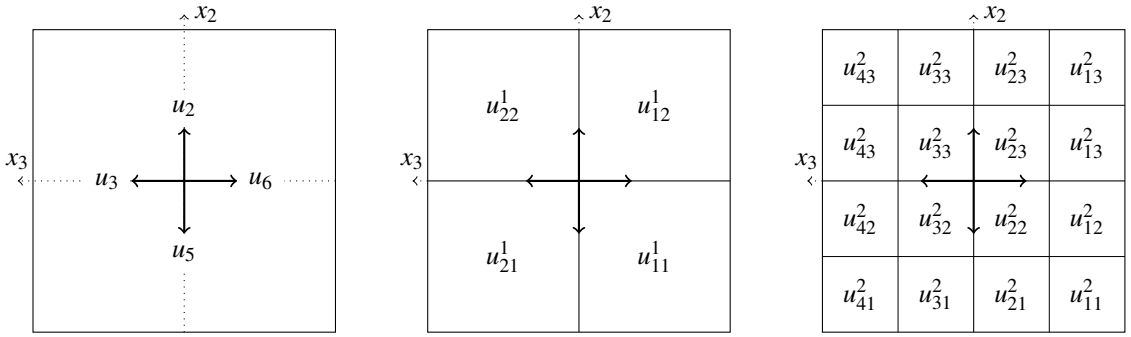
Figure 3.13: Root cone given by $u_1 = (1,0,0)^\top$ at level $h = 0$

$$\alpha_2(x) = \arctan(x_3/x_1) \qquad \alpha_3(x) = \arctan(x_2/x_1) \tag{3.6}$$

for the root cone u_1. Then, a point $x = (x_1,x_2,x_3)^\top$ is in that cone if $|\alpha_2(x)| \leq \pi/4$ and $|\alpha_3(x)| \leq \pi/4$ is true. Thus, the maximum opening angle at level $h = 0$ is $\pi/2$, as indicated in fig. 3.13. The angles α_2 and α_3 denote a wedge each. Their intersection defines the root cone of direction u_1. The index denotes the axis defining the edge of the wedge. The opening angles for the remaining root cones are defined analogously.

Nested cone construction Once, the six root cones at level $h = 0$ are constructed, we pass over to the nested cone construction at all levels $h > 0$ having a non-empty far-field. Recall, we are in the high frequency regime. Because of the uniform cluster-tree the cluster size $w_h = 2w_{h-1}$ doubles as we go from one level to the next higher one. This leads to the

formulation $w_h = 2^h w_0$. Correspondingly, the opening angle of the cones must be halved such that the criterion $\mathcal{O}(1/kw)$ is satisfied. And at the level h each of the six root cones is subdivided in $(2^h)^2$ smaller cones, each of opening angle $\pi/(2^{1+h})$. This is presented in fig. 3.14. Let us reuse the opening angles defined in (3.6). Then, a point x lies in the ij-th



(a) $h = 0$: one initial cone (b) $h = 1$: four sub-cones (c) $h = 2$: 16 sub-cones

Figure 3.14: The nested cones given by u_{ij}^h are exemplarily sketched for the root cone u_1 displayed in (a). Think of the outer boxes as the basis of this root cone. At level h we have $(2^h)^2$ sub-cones.

sub-cone at level h of the root cone u_1 if

$$-\frac{\pi}{4}+\frac{\pi}{2}\frac{i-1}{2^h} \leq \alpha_2(x) \leq -\frac{\pi}{4}+\frac{\pi}{2}\frac{i}{2^h} \quad \text{and} \quad -\frac{\pi}{4}+\frac{\pi}{2}\frac{j-1}{2^h} \leq \alpha_3(x) \leq -\frac{\pi}{4}+\frac{\pi}{2}\frac{j}{2^h}$$

is satisfied. Recall, the indices of the sub-cones are given by $i, j = 1, \ldots, 2^h$. In total, at level h the far field is subdivided in $6 \cdot 2^{2h}$ sub-cones. The remaining root cones are partitioned analogously.

The nested cone nature is advantageous because smaller cones at higher tree levels are entirely contained by larger cones at lower levels. This eases the construction of the directional translation operators as we will see in sec. 3.3.4. However, the cones are not isometric anymore, i.e., they cannot be obtained by applying rotations to an existing cone. In [40], some symmetries are presented which can be exploited – an approach to speed up the computation of the M2L operators.

3.3.4 Directional translation operators

In usual fast multipole methods the far-field constists of one interaction list. The new thing of the directional scheme is the fact that the far-field is partitioned in cones. As a consequence, we obtain as many interaction lists as we have cones. The efficient interpolation scheme presented in sec. 2.3.3 can be applied to each interaction list individually.

The arising M2M and L2L operators are considered translation operators between different tree levels. In our specific case they coincide with anterpolations and interpolations, respectively. The construction of these operators in the low frequency regime is straightforward. In the following, we look at their construction in the high frequency regime. Recall, the nested cone generation from the previous section. Smaller cones at higher tree levels are strictly contained by larger cones at lower three levels. We discuss the construction based on a two level process, in a step by step fashion. We will omit the extension of the proof to an arbitrary number of levels. Let us assume, we have computed the directional expansion with coefficients W_n^u for direction u at level $h+1$ (see (3.5)) as

$$W_n^u = \mathrm{e}^{\imath k u\cdot \bar{y}_n} \sum_{j=1}^{N} S_\ell(\bar{y}_n, y_j)\, \mathrm{e}^{-\imath k u\cdot y_j}\, w_j.$$

We want to calculate the contribution of W_n^u to the multipole coefficients $W_m^{u'}$ of the parent cluster at level h. Recall, the high frequency interpolation formula of the kernel (see (3.4)) reads as:

$$K(x, y_j) \sim \sum_{n\in\alpha} K(x, \bar{y}_n)\, \mathrm{e}^{\imath k u\cdot \bar{y}_n}\, S_\ell(\bar{y}_n, y_j)\, \mathrm{e}^{-\imath k u\cdot y_j}.$$

This formula is accurate as long as $K^u(x, y_i)$ can be interpolated at y_j from data at $\bar{y}_n$.

To obtain the M2M operator, we simply need to consider how $K(x, \bar{y}_n)$ can be interpolated using the interpolation points of the parent cluster, denoted $\bar{\bar{y}}_t$. However, such an interpolation operator requires using direction u' to maintain the accuracy of the scheme. Indeed, as we have shown in sec. 2.3, the interpolation scheme is accurate in a cluster of size w as long as we consider points x and y such that $x-y$ lies in the cone of axis u'. The interpolation formula reads then as

$$K(x, \bar{y}_n) \sim \sum_{t\in\alpha} K(x, \bar{\bar{y}}_t)\, \mathrm{e}^{\imath k u'\cdot \bar{\bar{y}}_t}\, S'_\ell(\bar{\bar{y}}_t, \bar{y}_n)\, \mathrm{e}^{-\imath k u'\cdot \bar{y}_n},$$

where $S'_\ell(\bar{\bar{y}}, \bar{y})$ is the interpolation polynomial for the parent cluster. We assume that the interpolation order ℓ is the same, although in practice ℓ might vary slightly between levels. However, those variations are typically small. Importantly there is an upper bound on ℓ, for a given error tolerance ε, which is independent of the cluster size.

In summary, the evaluation of $K(x, y_j)$ is done in two stages. First, we interpolate from $\bar{y}_n$ to y_j by means of (3.5). A key element of the proof is the fact that this interpolation step is accurate even when the points x_i and y_j interact at level h because the cone u' for the parent cluster is strictly contained inside the cone u at level $h+1$ for the child cluster. Secondly, we interpolate from $\bar{\bar{y}}_t$ to $\bar{y}_n$. This leads to the M2M operator

$$W_t^{u'} = \mathrm{e}^{\imath k u'\cdot \bar{\bar{y}}_t} \sum_{n\in\alpha} S'_\ell(\bar{\bar{y}}_t, \bar{y}_n)\, \mathrm{e}^{-\imath k u'\cdot \bar{y}_n}\, W_n^u.$$

A similar derivation shows that the L2L operator is given by

$$F_m^u = \mathrm{e}^{\imath k u' \cdot \bar{x}_m} \sum_{s \in \alpha} S'_\ell(\bar{\bar{x}}_s, \bar{x}_m)\, \mathrm{e}^{-\imath k u' \cdot \bar{\bar{x}}_s} F_s^{u'},$$

it is the transpose and complex conjugate counterpart of M2M.

3.3.5 Directional multilevel algorithm

By now, we have all tools to construct the directional multilevel algorithm. We have presented fast summation schemes for oscillatory kernels in the low and in the high frequency regime. Then, we have introduced a nested construction of directional cones and, based thereon, we have derived directional translation operators. Now, we pass over to the presentation of the multilevel algorithm. In fig. 3.15, we present a sketch of it.

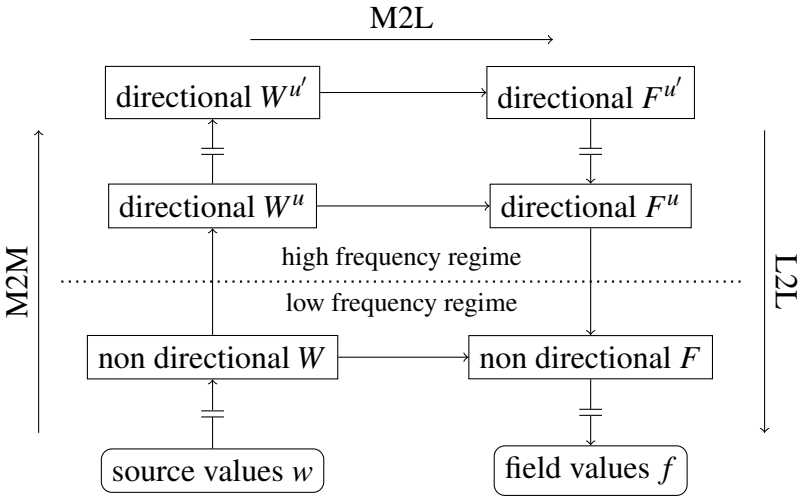


Figure 3.15: Plot of the directional fast multilevel scheme

Starting with the upward pass we first apply the M2M operators. In the low frequency regime, we compute non directional equivalent sources W from the sources at the leaf level. In the high frequency regime, we compute directional equivalent sources W^u. Then comes the traverse pass, we apply the M2L operators. The influence from the equivalent sources is added to the field values at the respective levels. In the third place comes the downward pass, we apply the L2L operators. And finally, after adding the near field contributions, we obtain the sought after field values f.

Algorithm Here, we present in a step-by-step manner how to apply the directional and multilevel scheme. For the sake of readability we omit the indices indicating cone and cluster-pair. This leads to a less complete but clearer notation.

1. Construct the *cluster-tree* such that all but the bottom most level are in the high frequency regime (This is not a must, we use it for the explanation of the procedure only).

2. The *upward-pass* starts at the leaf level and ends when no more long-range cluster interactions are possible, i.e., the minimal separation $\mathcal{O}(kw^2)$ becomes larger than the computational domain. The upward pass consists of M2M operations (anterpolation):

   - Perform the *non directional anterpolation* at the leaf level (low frequency regime): Anterpolate non directional equivalent sources W_n at interpolation points $\bar{y}_n$ from sources w_j at initial points y_j

   $$W_n = \sum_{j=1}^{N} S_\ell(\bar{y}_n, y_j)\, w_j \quad \text{for } n \in \alpha.$$

   This step is repeated going up the tree until we reach a cluster for which the low-frequency approximation breaks down.

   - Starting from the first level in high frequency regime, perform the *directional anterpolation* for all directions $\{u_c : c = 1,\dots,C\}$: Anterpolate directional equivalent sources W_t^u at interpolation points $\bar{\bar{y}}_t$ from sources W_n^u at points $\bar{y}_n$ of all 2^d child clusters

   $$W_t^{u'} = \mathrm{e}^{\imath k u' \cdot \bar{\bar{y}}_t} \sum_{n=1}^{2^d L} S_\ell(\bar{\bar{y}}_t, \bar{y}_n)\, \mathrm{e}^{-\imath k u' \cdot \bar{y}_n} W_n^u \quad \text{for } t \in \alpha.$$

   At the first level in the high frequency regime, we choose $W_n^u = W_n$, the non-directional multipole coefficients.

3. The *transverse-pass* consists of all M2L operations. The aim is to calculate the partial local expansions $\bar{F}_m^u$ for all clusters at all levels:

   $$\bar{F}_m^u = \sum_{n \in \alpha} K(\bar{x}_m, \bar{y}_n)\, W_n^u \quad \text{for } m \in \alpha.$$

   For low-frequency clusters, use W_n and $\bar{F}_m$.

4. The *downward-pass* starts at the level the upward pass has ended, and ends at the leaf level. It consists of L2L operations (interpolation). At the top of the tree $F_m^u = \bar{F}_m^u$.

- For high frequency clusters, perform the *directional interpolation* for all directions $\{u'_c : c = 1,\dots,C'\}$: Interpolate directional field values F_m^u at interpolation points $\bar{x}_m$ of all 2^d child clusters from directional field values $F_s^{u'}$ at interpolation points $\bar{\bar{x}}_s$

$$F_m^u = \bar{F}_m^u + \mathrm{e}^{\imath k u'\cdot \bar{x}_m} \sum_{s\in\alpha} S_\ell(\bar{\bar{x}}_s, \bar{x}_m)\, \mathrm{e}^{-\imath k u'\cdot \bar{\bar{x}}_s} F_s^{u'} \quad \text{for } m = 1,\dots,2^d L.$$

  Keep stepping down in the tree until the last tree level in the high frequency regime is reached.

- For low-frequency clusters, perform the *non directional interpolation*: Interpolate field values F_m at $\bar{x}_m$ from non directional field values F'_s at interpolation points $\bar{\bar{x}}_s$

$$F_m = \bar{F}_m + \sum_{l\in\alpha} S_\ell(\bar{\bar{x}}_s, \bar{x}_m)\, F'_s \quad \text{for } m = 1,\dots,2^d L.$$

5. Evaluate the *near-field* contribution f^N and add it to the interpolated far field contribution:

$$f_i \sim \sum_{m\in\alpha} S_\ell(\bar{x}_m, x_i)\, F_m + f_i^N.$$

Remark. In each multilevel method the M2L operations add the largest contribution to the computational cost. The reason is that they have to be performed many times for each cluster, whereas the M2M and L2L operations have to be performed only once per cluster. As such, the optimization of this operation is important. In sec. 2.3.3 we presented an approach for that.

3.3.6 Complexity

We present a general estimate of the computational complexity of the presented directional multilevel scheme. We do not examine the case where $N \to \infty$. The distribution of the points is allowed to become inhomogeneous, in the sense that the ratio of the largest point density over the smallest point density goes to ∞ (e.g., the points accumulate at a location).

We make our problem non-dimensional by considering a wavenumber k equal to 1. The diameter of the entire domain $\Omega \subset \mathbb{R}^d$ is then W (now effectively measured in wavelength). We assume that the problem is discretized such that the number of points per wavelength is approximately constant (this avoids the issue of accumulation mentioned above).

We will consider three cases, the same conclusion being reached in all cases. 1) The points are distributed more or less uniformly in Ω. 2) The points are distributed on a manifold of dimension $d-1$ in Ω, e.g., on a surface in $\mathbb{R}^3$. 3) The points are distributed on a manifold

of dimension $d-2$, e.g., on a curve in $\mathbb{R}^3$. In all cases, the method scales like $\mathcal{O}(N \log N)$. We give a more detailed proof of case 1) and outline case 2) and 3).

We will omit the analysis of the low-frequency regime since it follows the usual FMM complexity analysis and leads to an $\mathcal{O}(N)$ computational cost. With our assumption, the number of levels in the cluster tree is of order $\mathcal{O}(\log W)$.

Let us scatter points randomly in $\Omega \subset \mathbb{R}^d$. We then have $N = \mathcal{O}(W^d)$. If w denotes the cluster diameter, the number of clusters per level is $\mathcal{O}(W^d/w^d)$.

M2M and L2L operations In the high frequency regime there are $\mathcal{O}(w^{d-1})$ cone directions per tree level and both, the M2M and L2L operation involve $\mathcal{O}(1)$ operations. If we write the cluster diameter $w = W/2^h$ in terms of the wave number and let h denote the tree level, the complexity sums up to

$$\mathcal{O}\Big(\sum_{h=0}^{\log W} (W/2^h)^{d-1}\, 2^{hd}\Big) = \mathcal{O}\Big(W^{d-1} \sum_{h=0}^{\log W} 2^h\Big) = \mathcal{O}(W^d).$$

The complexity is of order $\mathcal{O}(W^d) = \mathcal{O}(N)$.

M2L operation This is the important step in the analysis. In the high frequency regime, the far field is bounded by the minimal distance of $\mathcal{O}(w^2)$ and the maximal distance of $\mathcal{O}(4w^2)$. Per cone, this represents $\mathcal{O}(w)$ clusters. Since there are $\mathcal{O}(w^{d-1})$ cones, the total number of interactions per cluster in all directions is $\mathcal{O}(w^d)$. Each interacting cluster involves a constant number of operations, thus, we end up with a complexity of $\mathcal{O}((W/w)^d\, w^d) = \mathcal{O}(W^d) = \mathcal{O}(N)$ per level. We assume the number of levels grows like $\mathcal{O}(\log N)$, hence, we conclude that the total complexity is $\mathcal{O}(N \log N)$.

We never reach $w = \mathcal{O}(W)$, the size of the domain Ω, in this method. Instead with the directional admissibility condition, at the highest level in the tree where interactions are still being computed, we have $w = \mathcal{O}(\sqrt{W})$. The dimensions do not seem to match because we made our problem dimensionless by choosing $k = 1$. With dimensions, we would have $w = \mathcal{O}(\sqrt{W/k})$. This means that the largest clusters in this method become smaller as k increases. In dimensionless units, the size of the largest clusters increase like $\sqrt{W}$, and therefore becomes small compared to the domain size W. This is in contrast with most FMMs in which the size of the largest clusters is of the same order as the domain diameter. The difference is that in our approach we maintain a low-rank representation (which leads to an increased number of clusters in the interaction list) whereas in the traditional FMM, e.g., [35, 36], the rank grows like the size of the cluster (but the size of the interaction list is bounded).

Now, we outline the proof for case 2), where the points lie on a manifold of dimension $d-1$ in Ω. The number of clusters in the interaction list per cone varies depending on the

direction and possibly many directions havie no interactions. However, for all cones the total number of interactions is $\mathcal{O}(w^2(w^2)^{d-2}/w^{d-1}) = O(w^{d-1})$. The complexity at a given level is $\mathcal{O}((W/w)^{d-1}w^{d-1}) = \mathcal{O}(W^{d-1}) = \mathcal{O}(N)$. The total complexity is $\mathcal{O}(N\log N)$ as before.

For case 3), where the points lie along a curve in $\mathbb{R}^3$, in the M2L operation, the size of the interaction list is $\mathcal{O}\left(w^2(w^2)^{d-3}/w^{d-2}\right) = \mathcal{O}(w^{d-2})$. The complexity at a given level is $\mathcal{O}((W/w)^{d-2}w^{d-2}) = \mathcal{O}(W^{d-2}) = \mathcal{O}(N)$. The total complexity is $\mathcal{O}(N\log N)$.

3.3.7 Numerical examples

We consider particles distributed on a) a surface and b) in a volume in $\mathbb{R}^3$. We construct the particle distributions and the underlying geometries with the meshing tool Gmsh [44]. In the following, we analyze three different example geometries shown in fig. 3.16. For completeness, we provide the short Gmsh scripts used to generate these geometries.

1. Sphere of diameter k as shown in fig. 3.16a.

```
lc = 0.1; k = 128;
Point(1) = {0,0,0,lc};   Point(2) = {0,0,-k/2,lc};
Point(3) = {k/2,0,0,lc}; Point(4) = {0,0,k/2,lc};
Circle(1) = {2,1,3};     Circle(2) = {3,1,4};
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{1,2};}
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{3,6};}
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{9,12};}
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{15,18};}}
```

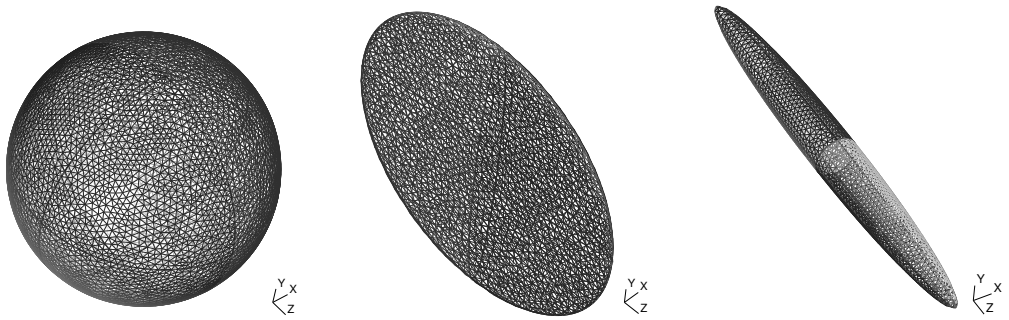
2. Oblate spheroid of diameter k in two directions and $k/10$ in the third direction as shown in fig. 3.16b.

```
lc = 0.1; k = 128;
Point(1) = {0,0,0,lc};   Point(2) = {0,0,-k/2,lc};
Point(3) = {k/20,0,0,lc}; Point(4) = {0,0,k/2,lc};
Point(5) = {-k/20, 0,0,lc};
Ellipse(1) = {2,1,4,3};  Ellipse(2) = {2,1,4,5};
Extrude{{1,0,0},{0,0,0},Pi/2}{Line{1,2};}
Extrude{{1,0,0},{0,0,0},Pi/2}{Line{3,6};}
Extrude{{1,0,0},{0,0,0},Pi/2}{Line{9,12};}
Extrude{{1,0,0},{0,0,0},Pi/2}{Line{15,18};}
```

3. Prolate spheroid of diameter k in one direction and $k/10$ in the other two directions as shown in fig. 3.16c.

```
lc = 0.1; k = 128;
Point(1) = {0,0,0,lc};   Point(2) = {0,0,-k/2,lc};
Point(3) = {k/20,0,0,lc}; Point(4) = {0,0,k/2,lc};
Point(5) = {-k/20,0,0,lc};
Ellipse(1) = {2,1,4,3};  Ellipse(2) = {4,1,1,3};
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{1,2};}
```

```
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{3,6};}
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{9,12};}
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{15,18};}
```



(a) Sphere (b) Oblate spheroid (c) Prolate spheroid

Figure 3.16: Example geometries

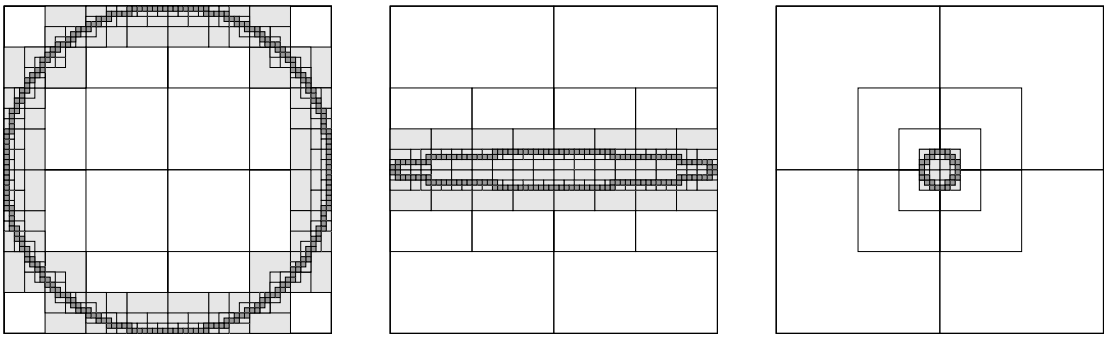
These code fragments only generate the surfaces. In order to end up with the volume, we need to add the following lines to the code fragments:

```
Surface Loop(100) = {26,23,20,17,14,11,8,5};
Volume(200) = {100};
```

The variable `lc` specifies the mesh size; we choose it to be 0.1. Hence, surface meshes have about 100 particles in a cluster of size one wave length, and volume meshes have about 1 000 particles in the same cluster.

The idea of these three different geometries is to analyze the efficiency of the directional algorithm with respect to the dimensionality of the computational domain: the first represents a 3-dimensional, the second a quasi 2-dimensional, and the third a quasi 1-dimensional object in $\mathbb{R}^3$. The more elongated the object is the fewer directional cones are required to cover the computational domain. This becomes visible in fig. 3.17. Cross-sections through the uniform oct-trees of the surfaces from fig. 3.16 are shown. Light gray clusters are in the high-frequency regime, they have directional expansions, dark gray clusters are in the low-frequency regime, they have non-directional expansions. All other clusters are not interacting with any other cluster and are not used in the method. In fig. 3.17, the depth of the oct-tree is chosen such that only leaf-level clusters are in the low-frequency regime.

For further numerical benchmarks, such as the comparison to another FMM (based on a plane-wave formulation, see [28]) and a validation of our implementation on irregular surfaces we refer to [76].



(a) Surface 3.16a (b) Surface 3.16b (c) Surface 3.16c

Figure 3.17: Cross-sections in the z-direction, for uniform oct-trees, of the surface particle distributions from fig. 3.16 with $k = 64$.

FMM parameters

At this stage, we have not developed a method to optimize the parameters in the directional FMM. As a result the running times are cerntainly not optimal and can be improved by changing our choice of parameters. In order to be consistent throughout all studies, however, we stick with the following parameter setting. Recall that w and k denote the length of the side of a cluster and the wave number, respectively.

Frequency regime threshold If $kw < 1$ holds, a cluster is in the low frequency regime, otherwise it is in the high frequency regime. This threshold basically separates clusters which are smaller than one wave length from the others.

Cone aperture The cone aperture is set to $1/kw < \alpha_u \leq 2/kw$ where $2\pi/\alpha_u \in \mathbb{N}$ and $\alpha_u \leq \pi/2$ due to the hierarchical cone construction. Except near the leaf level, the cone aperture is divided by two when going up the tree.

Admissible distance The admissible distance (minimum distance between the centers of two clusters that determines whether two clusters are in each other's interaction list) differs in the low and in the high frequency regime. In the *low frequency regime* it is $\text{dist}_{\text{lf}} = 2w$ (w is the length of one side of a cluster). In the *high frequency regime* it is $\text{dist}_{\text{hf}} = \max(2w, kw^2)$.

Accuracy We define the accuracy Acc of the direction FMM as $(\ell, \varepsilon_{\text{ACA}}) = (Acc, 10^{-Acc})$. Many numerical tests motivated this fusion of interpolation order ℓ and ACA accuracy ε_{ACA}. Also the numerical results in the ongoing section support it (we mainly refer to fig. 3.19).

Tree depth We vary the tree depth depending on the mesh type, i.e., surface or volume mesh, and the accuracy we want to achieve. The reason is that computational timings for near- and far-field can be balanced in that way. Recall that we chose a mesh size of 0.1, i.e., surface meshes have about 100 and volume meshes about 1000 particles in a cluster having a size of 1 wavelength. Hence, both volume mesh and low accuracy require a deeper tree. We chose the tree depth for *surface meshes* such that leaf clusters are of size

- $\ell < 7$: $1/2$ wavelength
- $7 \leq \ell < 10$: 1 wavelength
- $\ell \geq 10$: 2 wavelengths

and for *volume meshes*

- $\ell < 7$: $1/4$ wavelength
- $7 \leq \ell < 10$: $1/2$ wavelength
- $\ell \geq 10$: 1 wavelength

in order to balance computational timings for near-field and far-field.

Considering this parameter list, a large number of different settings can be obtained. Performance results, when varying the accuracy *Acc* are apparent, less accuracy goes along with faster computations and vice versa. Also varying the parameters from the admissibility criteria leads to significant performance changes. For example, if we increase the cone aperture, we get less cones but the accuracy of the results suffers. The same is true if we decrease the admissible distance.

Surface particle distributions

We show computational timings of surface particle distributions for accuracies $Acc = 4$ and $Acc = 7$ in tab. 3.2, 3.3 and 3.4. All shown timings are those for the matrix-vector product; they do not include the precomputation time. We achieve an almost linear growth as can be seen by means of the convergence rate roc_t which is defined as

$$\mathrm{roc}_t = \frac{\ln(N_{2k}/N_k)}{\ln(t_{2k}/t_k)},$$

(Acc,k,N)	ε_{L^2}	t [sec]	roc_t
(4, 4,5.69e+3)	1.18e-4	0.5	
(4, 8,2.26e+4)	4.64e-4	2	0.995
(4,16,9.26e+4)	4.79e-4	8	1.017
(4,32,3.81e+5)	6.78e-4	30	1.070
(4,64,1.54e+6)	4.47e-4	139	0.911
(7, 4,5.69e+3)	2.13e-7	1	
(7, 8,2.26e+4)	9.73e-7	6	0.770
(7,16,9.26e+4)	5.89e-7	29	0.900
(7,32,3.81e+5)	1.31e-6	143	0.887
(7,64,1.54e+6)	1.07e-6	632	1.034

Table 3.2: Timings for surface mesh of fig. 3.16a

(Acc,k,N)	ε_{L^2}	t [sec]	roc_t
(4, 8,1.16e+4)	5.07e-4	1	
(4, 16,4.75e+4)	6.16e-4	4	1.017
(4, 32,1.93e+5)	1.64e-3	14	1.119
(4, 64,7.87e+5)	9.23e-4	55	1.029
(4,128,3.23e+6)	1.10e-3	232	0.981
(7, 8,1.16e+4)	2.23e-6	3	
(7, 16,4.75e+4)	3.05e-6	15	0.876
(7, 32,1.93e+5)	2.99e-6	59	1.023
(7, 64,7.87e+5)	1.17e-6	235	1.017
(7,128,3.23e+6)	1.40e-6	1105	0.912

Table 3.3: Timings for surface mesh of fig. 3.16b

(Acc,k,N)	ε_{L^2}	t [sec]	roc_t
(4, 16,7.66e+3)	3.01e-4	0.5	
(4, 32,2.97e+4)	3.25e-4	2	0.977
(4, 64,1.19e+5)	1.73e-4	9	0.922
(4,128,4.83e+5)	5.46e-4	36	1.010
(4,256,1.94e+6)	1.28e-4	155	0.952
(7, 16,7.66e+3)	9.30e-7	2	
(7, 32,2.97e+4)	5.10e-7	7	1.081
(7, 64,1.19e+5)	4.25e-7	32	0.913
(7,128,4.83e+5)	9.83e-7	155	0.888
(7,256,1.94e+6)	5.43e-7	682	0.938

Table 3.4: Timings for surface mesh of fig. 3.16c

Volume particle distributions

We show timing studies of volume particle distributions for accuracies $Acc = 4$ and $Acc = 7$ in tab. 3.5, 3.6 and 3.7. Again, the shown timings do not include the precomputation time and we can see the almost linear growth by means of roc_t.

Contrary to the surface mesh studies above, here, we are not able to present results for $k = 64,\ 128$ and 256, respectively, due to limited memory. In tab. 3.5 the computation for $k = 32$ is only possible for an accuracy $Acc = 4$, the same holds in tab. 3.6 for $k = 64$. In tab. 3.7 no computation for $k = 128$ is possible.

(Acc,k,N)	ε_{L^2}	t [sec]	roc_t
(4, 4,6.14e+4)	8.36e-4	9	
(4, 8,3.36e+5)	9.58e-4	53	0.958
(4,16,3.56e+6)	5.54e-3	552	1.007
(4,32,4.26e+7)	3.87e-3	7302	0.961
(7, 4,6.14e+4)	7.14e-7	45	
(7, 8,3.36e+5)	3.10e-6	271	0.946
(7,16,3.56e+6)	8.80e-6	3362	0.937

Table 3.5: Timings of volume meshes of fig. 3.16a

(Acc,k,N)	ε_{L^2}	t [sec]	roc_t
(4, 8,4.04e+4)	1.55e-3	5	
(4,16,3.94e+5)	8.87e-4	52	0.972
(4,32,2.21e+6)	1.46e-3	334	0.927
(4,64,2.62e+7)	2.49e-3	3999	0.996
(7, 8,4.04e+4)	6.38e-6	14	
(7,16,3.94e+5)	6.07e-6	290	0.751
(7,32,2.21e+6)	6.78e-6	1562	1.024

Table 3.6: Timings of volume meshes of fig. 3.16b

(Acc,k,N)	ε_{L^2}	t [sec]	roc_t
(4,16,2.69e+4)	4.32e-4	3	
(4,32,2.02e+5)	1.19e-3	29	0.888
(4,64,1.56e+6)	2.26e-4	252	0.945
(7,16,2.69e+4)	1.10e-6	13	
(7,32,2.02e+5)	2.02e-6	127	0.884
(7,64,1.56e+6)	1.12e-6	1125	0.937

Table 3.7: Timings of volume meshes of fig. 3.16c

Direct matrix-vector product

Figure 3.18 shows a comparison of the direct matrix-vector product and our method in terms of computational time. The results are obtained with the surface meshes from fig. 3.16 and all timings include *precomputation and matrix-vector product*. The direct matrix-vector product is only faster for the smallest surface mesh from the sphere of fig. 3.16a with $k = 4$ and an accuracy $Acc = 7$. Our method outperforms the direct method in all other cases.

Overall error convergence and timings

Figure 3.19 shows the convergence of the relative error ε_{L^2} for interpolation order up to $\ell = 13$ and accuracies up to $\varepsilon_{\text{ACA}} = 10^{-13}$. Both are key parameters for the accuracy of the method. We use the surface mesh from fig. 3.16c with a wave number $k = 128$ (483389 particles). We chose the tree depth as described above. We plot the relative error ε_{L^2} vs. the target accuracy ε_{ACA}; each curve corresponds to a given interpolation order ℓ. No matter what ℓ we choose, if we fix ε_{ACA}, we cannot get beyond a certain accuracy, and vice versa.

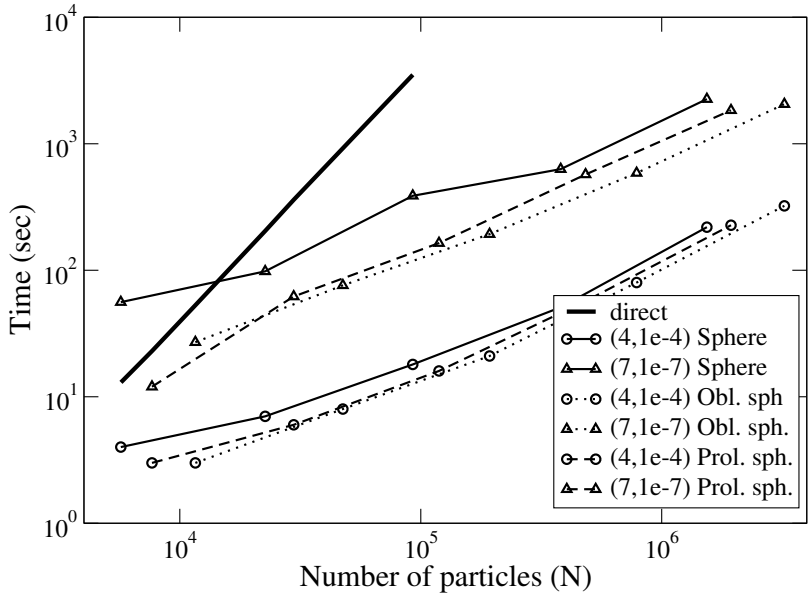


Figure 3.18: Timings for the direct method and the directional FMM. In both cases the timings include the setup of the matrix as well as the matrix-vector product itself. The surface meshes originate from fig. 3.16.

For example, with an accuracy of $\varepsilon_{L^2} \sim 10^{-4}$, there is no point in choosing ℓ greater than 5. If we need an accuracy of $\varepsilon_{L^2} \sim 10^{-6}$ we need to use at least $\ell = 7$ and $\varepsilon_{\text{ACA}} = 10^{-7}$. Figure 3.19 shows that accuracies up to $\varepsilon_{L^2} < 10^{-11}$ can be achieved. The motivation for the fusion of the interpolation order ℓ and the ACA accuracy ε_{ACA} to define the accuracy *Acc* of the directional FMM stems mainly from fig. 3.19. Figure 3.20 shows timings for the accuracies ε_{L^2} presented in fig. 3.19.

ACA plus QR-decomposition vs. SVD

In tab. 3.8 we compare the low rank $(r_{\text{ACA}}/r_{\text{SVD}})$ and the precomputation time $(t_{\text{ACA}}/t_{\text{SVD}})$ of ACA plus QR-decomposition against the SVD approach (see [42]). We prescribe the accuracy $\varepsilon_{\text{ACA}} = \varepsilon_{\text{SVD}} = 10^{-4}$. Shown are results from the surface mesh from fig. 3.16c. In the second and third column, we compare the low rank obtained by ACA and SVD, respectively. The left most values in each column represents the average low rank at the highest level in the tree, the next value is the next lower level and so on. The right-most value corresponds to a leaf where the non-directional scheme is used. Switching from the directional method to the non-directional one results in a jump of about 60% of the rank. This jump would presumably be reduced if we optimized the parameters in the method. The SVD approach provides the smallest possible rank for a prescribed L^2 error. The rank obtained with ACA is close. However, the precomputation time for ACA is significantly

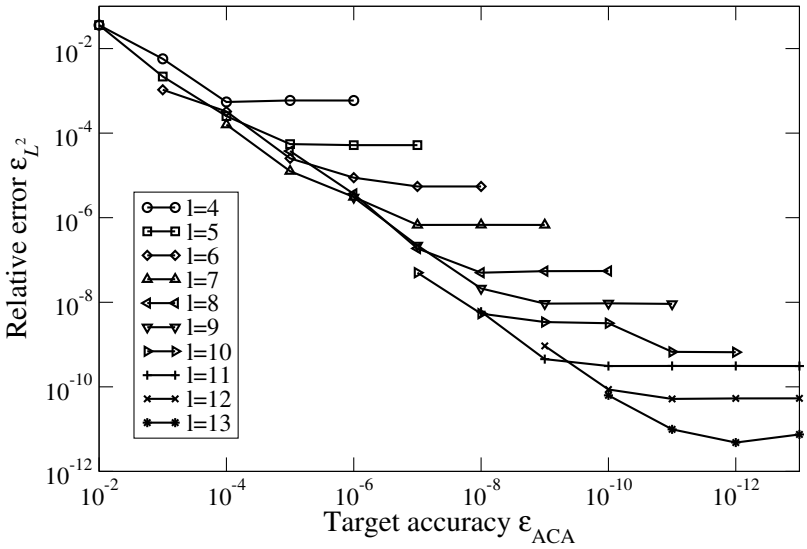


Figure 3.19: Relative error ε_{L^2} for the surface mesh from fig. 3.16c with wave number $k = 128$ (483389 particles); ε_{ACA} is the accuracy for ACA; ℓ is the Chebyshev interpolation order.

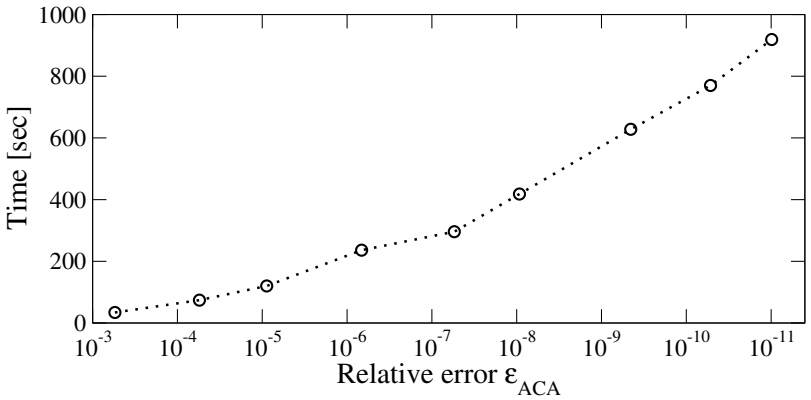


Figure 3.20: Timings for the matrix-vector product for accuracies up to 10^{-11} for the surface mesh from fig. 3.16c and $k = 128$.

lower (N vs. N^3 for a matrix of size N).

k	r_{ACA}	r_{SVD}	t_{ACA}	t_{SVD}
16	26, 26, 53	19, 24, 42	2	4
32	20, 26, 31, 51	15, 20, 25, 42	3	7
64	21, 29, 32, 51	15, 21, 25, 42	5	13
128	19, 19, 30, 34, 49	14, 15, 22, 25, 42	11	40
256	19, 23, 31, 32, 51	14, 17, 22, 25, 42	22	93

Table 3.8: Low rank r and timings t for ACA+QR and SVD.

Table 3.8 shows the decay of the singular values for different tree levels. The singular values in the leaf level decay slowest, those from the highest level having a non-empty far-field fastest. The decay behavior of the singular values can be improved by decreasing the cone aperture or increasing the admissible distance. As already mentioned earlier in this section, we have not yet fully optimized the parameter setting of our method.

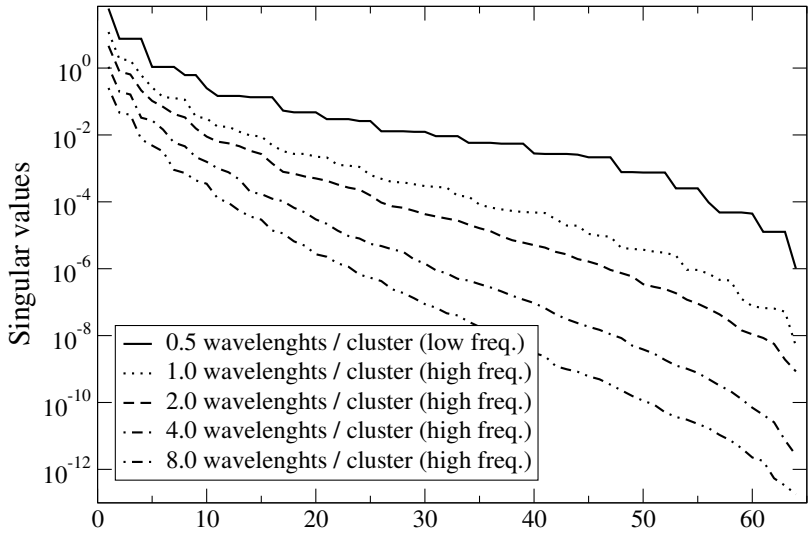


Figure 3.21: Singular values for the surface mesh of fig. 3.16c with $k = 256$ and $\ell = 4$. The index of the singular values is shown on the x-axis. Leaf clusters have a size of $1/2$ wave length.

4 BOUNDARY ELEMENT METHODS FOR ACOUSTICS

4.1 Basic equations for acoustics

In the following, we go through the laws of mechanics which we need to explain how acoustic waves propagate from the source to the receiver. Let us start with some introductory remarks. A fluid particle represents an infinitesimal volume of the fluid, large enough that contained fluid may be thought of a continuous medium, small enough that the acoustic variables are uniform. Even without the presence of an acoustic wave such particles feature an average motion. We introduce the displacement $U(x,t)$ of a fluid particle and its velocity $V(x,t) = \partial U(x,t)/\partial t$. When considering only small perturbations of the equilibrium state (acoustic waves of relatively small amplitudes), changes in the instantaneous density $\rho_i(x,t)$ and in the instantaneous pressure $P_i(x,t)$ will be small compared to its equilibrium values

$$\rho_i = \rho_0 + \rho, \quad P_i = P_0 + P, \quad \text{with} \quad \rho \ll \rho_0, \quad P \ll P_0.$$

These assumptions are necessary to derive the simplest equations for acoustic wave propagation in a fluid. Fortunately, experiments have shown that such simplification adequatly describe acoustic phenomena (see [63]).

Constitutive equation This equation describes the relation between pressure and density. We use the relation for barotropic fluids $P_i(\rho_i)$, where the pressure depends only on the density. After expanding this relation into a Taylor series around ρ_0 and truncating it after the linear term we end up with $P_i(\rho_i) \sim P_i(\rho_0) + P_i'(\rho_0)(\rho_i - \rho_0)$ With the condition $P_0 = P_i(\rho_0)$ we obtain the linearized constitutive equation

$$P = c^2 \rho \tag{4.1}$$

with $c^2 = P_i'(\rho_0) = B/\rho_0$ where B denotes the bulk modulus of the acoustic fluid. Recall, the index i of the instantaneous pressure $P_i(x)$ and density $\rho_i(x)$ does not denote any axis direction of the coordinate system, both are scalar fields depending on $x \in \mathbb{R}^d$.

Continuity equation To connect the motion of the fluid with its compression or expansion we need use the equation of continuity or equation of mass conservation. It relates the velocity of a fluid element $V(x,t)$ and its instantaneous density $\rho_i(x,t)$. Consider a

infinitesimal element $d\Omega \subset \mathbb{R}^d$ being fixed in space and fluid elements flowing through it. The net influx of mass $-\nabla \cdot (\rho_i V)\, d\Omega$ is the difference between the mass which enters and leaves $d\Omega$. The equation of continuity states that the net influx of mass must be equivalent to the rate with which the mass in $d\Omega$ increases $(\partial \rho_i / \partial t)\, d\Omega$. Assuming that ρ_0 is a sufficiently weak function of time and space the relation simplifies to

$$\frac{\partial \rho}{\partial t} + \rho_0 \nabla \cdot V = 0. \tag{4.2}$$

Euler equation The Euler equation (momentum conservation equation) connects the velocity $V(x,t)$ of a fluid element with the instantaneous pressure $P_i(x,t)$ acting on it. Consider the same infinitesimal element $d\Omega$ as before, but now moving along with the fluid element and containing fluid of mass $dm = \rho_0\, d\Omega$. The net force dF acting on the fluid element corresponds to the pressure difference acting on it. Sometimes also body forces have to be included in the computations, however, we neglect them here. The net force reads as $dF = -\nabla P_i\, d\Omega$. It accelerates the fluid element according to Newton's second law $dF = A\, dm$. The acceleration $A(x,t)$ of the fluid element is obtained from its velocity $V(x,t)$, it reads as $A = \partial V/\partial t + (V \cdot \nabla)V$. As in [63] we assume that the convective acceleration is sufficiently small $|(V \cdot \nabla)V| \ll |\partial V/\partial t|$ and P_0 is a sufficiently weak function of space. After equating the resulting net force and acceleration terms we obtain the linearized Euler equation

$$\rho_0 \frac{\partial V}{\partial t} = -\nabla P. \tag{4.3}$$

4.1.1 Wave equation

Note, at this point, we have five unknowns (three velocity components, pressure, density) and five equations (one in (4.1) and (4.2), respectively, and three in (4.3)). Hence, the combination of these equations yields to one differential equation with one dependent variable, which is going to be the pressure $P(x,t)$. First, we take the divergence of (4.3)

$$\nabla \cdot \nabla P = -\rho_0 \nabla \cdot \frac{\partial V}{\partial t} = -\rho_0 \frac{\partial (\nabla \cdot V)}{\partial t} \tag{4.4}$$

where $\nabla \cdot \nabla = \Delta$ is the Laplacian in $\mathbb{R}^d$. Second, after plugging (4.2) into (4.4) and using (4.1) we obtain the homogeneous acoustic wave equation

$$\Delta P - \frac{1}{c^2} \frac{\partial^2 P}{\partial t^2} = 0 \tag{4.5}$$

describing acoustic waves as pressure perturbations propagating at a velocity c which depends on the material paramenter of the fluid.

4.1.2 Helmholtz equation

Often, integral transforms, such as the Laplace or Fourier transform, are used to bring partial differential equations like the wave equation (4.5) into a form of the Helmholtz equation and to get rid of the time variable. We make use of the inverse Laplace transform

$$F(t) = \frac{1}{2\pi\imath} \int_{c-\imath\infty}^{c+\imath\infty} f(s)\, \mathrm{e}^{st}\, ds$$

where the integration is conducted along the vertical line $\mathrm{Re}\, s = c$ in the complex plane such that c is greater than the real part of all singularities of $f(s)$. If we apply the above transform to the acoustic pressure $P(x,t)$ and insert it in (4.5) we obtain the following form of the Helmholtz equation

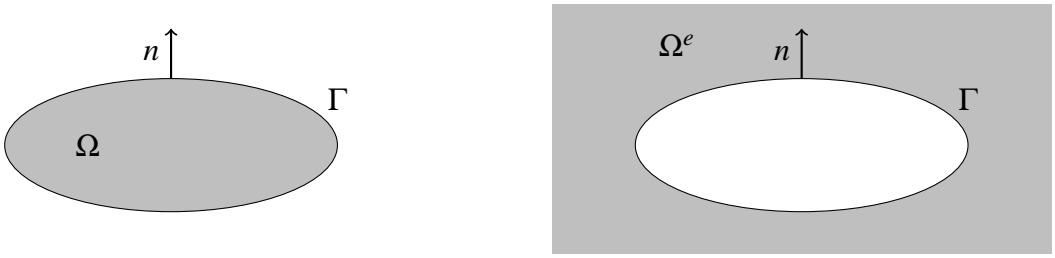
$$\Delta p - \left(\frac{s}{c}\right)^2 p = 0 \tag{4.6}$$

where $s \in \mathbb{C}$. Note, if we set $\mathrm{Re}\, s = 0$ and $\mathrm{Im}\, s = \omega$ we get the commonly known form of the Helmholtz equation $\Delta p + k^2 p = 0$ with the wavenumber $k = \frac{\omega}{c} > 0$ of dimension $1/\mathrm{m}$. This form result also if an inverse Fourier transform would have been used, instead. Also, the common approach to split off the time variable in a time-dependent scalar field as $F(x,t) = f(x)\, \mathrm{e}^{-\imath\omega t}$ can be traced back to the application of the latter transform.

4.1.3 Boundary- and initial conditions

In order to have well-posed problems, partial differential equations require additional well defined constraints. The acoustic wave equation (4.5) needs initial and boundary conditions. The Helmholtz equation (4.6) needs boundary conditions. If such conditiones are provided we speak of well posed boundary-, respectively, initial-boundary value problems which have unique solutions [91]. Generally speaking, such problems are a differential equation together with some constraints. Consequently, a solution to such problems is a solution to the associated differential equation which also satisfies the prescribed constraints.

How does this apply to acoustics? Let us introduce some notations. Think of two possible acoustic domain configurations as depticted in fig. 4.1. On one hand we can have an *interior* acoustic domain $\Omega \subset \mathbb{R}^3$ with its boundary $\Gamma = \partial\Omega$. On the other hand we can have an *exterior* acoustic domain $\Omega^e = \mathbb{R}^3 \setminus (\Omega \cup \Gamma)$. Note, the boundary $\Gamma = \partial\Omega = \partial\Omega^e$ is the equivalent for both configurations, also the normal n on Γ points in the same directions in both configurations.



(a) Interior acoustic domain Ω (b) Exterior acoustic domain Ω^e

Figure 4.1: Interior and exterior acoustic domains

Boundary conditions Boundary conditions are spatial constraints on the boundary. In the following, we distinguish between Dirichlet, Neumann or mixed boundary value problems (e.g., [91]). *Dirichlet problems* have the acoustic pressure p prescribed on the entire boundary

$$p(x) = g_D(x) \quad \text{for } x \in \Gamma,$$

where the acoustic pressure on the boundary is commonly denoted as Dirichlet datum, hence Dirichlet problem. *Neumann problems* have the acoustic flux q prescribed on the entire boundary

$$q(x) = \frac{\partial p(x)}{\partial n_x} = g_N(x) \quad \text{for } x \in \Gamma.$$

The acoustic flux on the boundary is commonly denoted as Neumann datum, hence Neumann problem. *Mixed problems* have the Dirichlet datum g_D prescribed on the Dirichlet boundary Γ_D and the Neumann datum g_N on the Neumann boundary Γ_N. In this case, the boundary $\Gamma = \Gamma_D \cup \Gamma_N$ is split in two non overlapping subsets Γ_D and Γ_N. A mixed boundary value problem for the Helmholtz equation reads as

$$\Delta p(x) - \left(\frac{s}{c}\right)^2 p(x) = 0 \quad \text{for } x \in \Omega, \tag{4.7}$$

with the boundary conditions

$$\begin{aligned} p(x) &= g_D(x) \quad \text{for } x \in \Gamma_D, \\ q(x) &= g_N(x) \quad \text{for } x \in \Gamma_N. \end{aligned}$$

Initial conditions The Helmholtz equation does not feature any time dependency, i.e., only spatial constraints are needed for well-posedness. Conversely, the solution $P(x,t)$ of the wave equation is also a function of time. Hence, well-posed problems additionally need also initial conditions. They prescribe initial values for $P(x,0)$ and its first time derivative

$\dot{P} = \partial P/\partial t$. A mixed initial-boundary value problem for the wave equation reads as

$$\Delta P(x,t) - \frac{1}{c^2}\frac{\partial^2 P(x,t)}{\partial t^2} = 0 \quad \text{for } x \in \Omega,\ t > 0,$$

with the initial conditions

$$P(x,0) = P_0(x) \quad \text{and} \quad \dot{P}(x,0) = \dot{P}_0(x) \quad \text{for } x \in \Omega,$$

and the boundary conditions

$$\begin{aligned} P(x,t) &= G_D(x,t) \quad \text{for } x \in \Gamma_D,\ t > 0, \\ Q(x,t) &= G_N(x,t) \quad \text{for } x \in \Gamma_N,\ t > 0. \end{aligned}$$

Sommerfeld's radiation condition So far, we have only considered interior acoustic domains, i.e., the governing acoustic equations were defined in Ω. From [80, 91] we know that interior Helmholtz problems are well-posed. In order to make exterior problems, as depicted in fig. 4.1b, uniquely solvable we need to impose an additional boundary condition at infinity. It is called Sommerfeld's radiation condition [90] and reads for our form of the Helmholtz equation (4.6) as

$$\left| \frac{x}{|x|} \cdot \nabla p(x) + \frac{s}{c} p(x) \right| = \mathcal{O}\left(\frac{1}{|x|^2} \right) \quad \text{as } |x| \to \infty. \tag{4.8}$$

The radiation condition is also known as outgoing wave condition and it is equivalent to a vanishing acoustic pressure field $p(x)$ at infinity. By substituting s with $-s$ one obtaines the complement of the outgoing wave condition, i.e., the solution is subjected to incoming waves.

In contrast to Helmholtz problems, problems for the wave equation do not require any condition at infinity. Nevertheless, if initial conditions vanish at infinity, what excludes incoming waves, the solution satisfies an outgoing wave condition which reads similarly as Sommerfeld's radiation condition [46].

4.2 Boundary integral formulations

Representation formulae for the solution of the Helmholtz and the wave equation provide the main ingredient for boundary integral equations. The discretization of these boundary integral leads to boundary element formulations, which are the tool we use to numerically solve acoustic problems.

Consider the Helmholtz equation (4.6) in a bounded domain $\Omega \subset \mathbb{R}^3$. Its weak formulation reads as

$$\int_\Omega \left(\Delta p - \left(\tfrac{s}{c}\right)^2 p\right) v\, dx = 0 \quad \text{for } x \in \Omega \tag{4.9}$$

for some suitably chosen test function v. Our first goal is to derive Green's first identity. To do so we first deploy the divergence theorem

$$\int_\Omega \nabla \cdot h\, dx = \int_\Gamma \frac{\partial h}{\partial n}\, ds \tag{4.10}$$

to the weak formulation in (4.9). In the above equation Γ stands for the boundary of Ω and n for the normal vector on Γ pointing out of Ω. The normal derivative is defined as $\partial h/\partial n = n \cdot \nabla h$ with the continuously differentiable vector function $h : \mathbb{R}^3 \to \mathbb{R}^3$. We let this function be $h = v\nabla p$ with $p, v : \mathbb{R}^3 \to \mathbb{C}$ being scalar functions. By applying the divergence operator to h we get

$$\nabla \cdot h = \nabla \cdot (v \nabla p) = v\Delta p + \nabla p \cdot \nabla v$$

with the Laplacian $\Delta = \nabla \cdot \nabla$. Next, we plug the above relation into (4.10) and get

$$\int_\Omega v\Delta p\, dx = -\int_\Omega \nabla p \cdot \nabla v\, dx + \int_\Gamma v \frac{\partial p}{\partial n}\, ds.$$

After inserting the above equation into (4.9) we obtain *Green's first identity* for the Helmholtz equation

$$\int_\Omega \left(\Delta p - \left(\tfrac{s}{c}\right)^2 p\right) v\, dx = a(p,v) + \int_\Gamma v \frac{\partial p}{\partial n}\, ds,$$

with the symmetric bilinear form

$$a(p,v) = -\int_\Omega \nabla p \cdot \nabla v\, dx - \left(\tfrac{s}{c}\right)^2 \int_\Omega pv\, dx.$$

We repeat the same procedure by reformulating Green's first identity with the vector function $h = p\nabla v$ and get

$$\int_\Omega \left(\Delta v - \left(\tfrac{s}{c}\right)^2 v\right) p\, dx = a(v,p) + \int_\Gamma p \frac{\partial v}{\partial n}\, ds,$$

Due to the symmetry of the bilinear forms $a(p,v) = a(v,p)$ we can subtract both equations from each other and obtain *Green's second identity* for the Helmholtz equation

$$\int_\Omega \left(\Delta p - \left(\tfrac{s}{c}\right)^2 p\right) v\, dx - \int_\Gamma v \frac{\partial p}{\partial n}\, ds = \int_\Omega \left(\Delta v - \left(\tfrac{s}{c}\right)^2 v\right) p\, dx - \int_\Gamma p \frac{\partial v}{\partial n}\, ds. \tag{4.11}$$

4.2.1 Representation formulae

The fundamental solution $P_s : \mathbb{R}^3 \to \mathbb{C}$ of the Helmholtz equation is a function which satisfies

$$\Delta P_s(x,y) - \left(\tfrac{s}{c}\right)^2 P_s(x,y) = -\delta(y-x) \tag{4.12}$$

for all $x, y \in \mathbb{R}^3$ and reads as

$$P_s(x,y) = \frac{1}{4\pi} \frac{\mathrm{e}^{-\frac{s}{c}|x-y|}}{|x-y|}. \tag{4.13}$$

First, we use the weak formulation (4.9) to get rid of the first volume integral in (4.11). Then, we replace the test function v in (4.11) with the fundamental solution P_s. With (4.12) and the screening property of the Dirac distribution

$$\int_\Omega p(y)\delta(y-x)\,dy = p(x)$$

we get rid of the second volume integral in (4.11). By introducing the acoustic flux $q = \partial p/\partial n$ as the normal derivative of the acoustic pressure p we obtain the *representation formula*

$$p(x) = \int_\Gamma P_s(x,y)\, q(y)\, ds_y - \int_\Gamma \frac{\partial P_s(x,y)}{\partial n_y} p(y)\, ds_y \quad \text{for } x \in \Omega. \tag{4.14}$$

The derivation of the representation formula for the wave equation is similar. We only provide a very brief outline. For a more detailed derivation we refer the reader to [2]. The starting point is again Green's second identity, however, we consider its time-domain counterpart. The fundamental solution $P : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ reads as

$$P_t(x,y,t,\tau) = \frac{1}{4\pi} \frac{\delta(t-\tau-\frac{|x-y|}{c})}{|x-y|}.$$

Here, the fundamental solution P represents the wave propagation due to an impulse at emission time τ in point y (see, e.g., [79]). It can be obtained via an inverse transformation of the fundamental solution of the Helmholtz equation (4.13). The *time-domain representation formula* reads as

$$P(x,t) = \int_0^t \int_\Gamma P_t(x,y,t,\tau)\, Q(y,\tau)\, ds_y\, d\tau - \int_0^t \int_\Gamma \frac{\partial P_t(x,y,t,\tau)}{\partial n_y} P(y,\tau)\, ds_y\, d\tau \tag{4.15}$$

provides the solution $P(x,t)$ for all $t \in \mathbb{R}^+$ and for all $x \in \Omega \setminus \Gamma$ as long as the Dirichlet datum $P(y,\tau)$ and the Neumann datum $Q(y,\tau)$ for all $\tau \in [0,t]$ and $y \in \Gamma$ are known.

4.2.2 Boundary integral equation

The representation formulae (4.14) and (4.15) provide the solution of the problem once the complete Dirichlet and Neumann data are known on the entire boundary. This is usually not the case, they first need to be computed. The tool therefore are *boundary integral equations*.

We can derive them from the representation formulae by shifting x from the domain Ω to its boundary Γ. This shifting procedure is a rather cumbersome process and has been elaborated in many textbooks, e.g., in [43] and [91]. The difficulty represents the fact that once x gets to the boundary singularities of different types appear in the integrals: Think, for example, of the definition of the fundamental solution $P_s(x,y)$, which becomes singlular as $|x-y| \to 0$. Weakly singular integrals are finite, strong singular integrals can be interpreted as their Cauchy principal value. For other cases, such as hyper-singular integrals, which are rather complex and problem dependent we refer the reader to [62].

We first consider the boundary integral equation for the Helmholtz equation. No problem poses the *single layer potential*, it is well defined as x approaches the boundary. It reads as

$$(\mathcal{V}_s q)(x) = \int_\Gamma P_s(x,y) q(y)\, ds_y \quad \text{for } x \in \Gamma. \tag{4.16}$$

This operator contains a so called *weakly-singular* integral, which is well defined, even though the integrant is not analytic as x approaches y (see [91]).

A more profound analysis requires the second integral on the right hand side of (4.14) as x is shifted to the boundary. It becomes

$$\lim_{\Omega \ni x \to x \in \Gamma} \int_\Gamma \frac{\partial P_s(x,y)}{\partial n_y} p(y)\, ds_y = (-\mathcal{I} + \mathcal{C}(x))\, p(x) + (\mathcal{K}_s p)(x).$$

Usually, the limiting process is performed by augmenting the boundary at the point x with a hemisphere $B_\varepsilon(x)$ of vanishing radius ε. Herewith, the *double layer potential* on the boundary is defined as

$$(\mathcal{K}_s p)(x) = \lim_{\varepsilon \to 0} \int_{\Gamma \setminus B_\varepsilon(x)} \frac{\partial P_s(x,y)}{\partial n_y} p(y)\, ds_y \quad \text{for } x \in \Gamma. \tag{4.17}$$

In the present case this integral exists in an inproper sense (weakly singular integral). Usually such an integration is called *Cauchy principal value* [54]. The integral on the remaining part of the boundary is defined as

$$(\mathcal{C} p)(x) = \lim_{\varepsilon \to 0} \int_{\partial B_\varepsilon(x) \cap \Omega} \frac{\partial P_s(x,y)}{\partial n_y} p(y)\, ds_y \quad \text{for } x \in \Gamma. \tag{4.18}$$

It is often referred to as *integral free term* [43, 91]. It is related to the interior solid angle of Γ at the point x. If Γ is smooth in the vicinity of x the integral free term becomes $\mathcal{C}(x) = 1/2$.

Finally after having applied this shifting process $\Omega \ni x \to \mathbf{x} \in \Gamma$ to the entire representation formula (4.14) we obtain the *boundary integral equation* for the Helmholtz equation which reads in operator notation as

$$\mathcal{C}(x)p(x) + (\mathcal{K}_s p)(x) = (\mathcal{V}_s q)(x) \quad \text{for } x \in \Gamma. \tag{4.19}$$

The same procedure has to be followed to obtain the *time-domain boundary integral equation* for the wave equation. In operator notation it reads as

$$\mathcal{C}(x)P(x,t) + (\mathcal{K}_t * P)(x,t) = (\mathcal{V}_t * Q)(x,t). \tag{4.20}$$

The operator $*$ denotes the convolution in time, e.g., $f * g = \int_0^t f(\tau)g(t-\tau)\,d\tau$. The time-domain counterparts of the operators defined in (4.16) and (4.17) are defined similarly. The single layer potential reads as

$$(\mathcal{V}_t * Q)(x,t) = \int_0^t \int_\Gamma P_t(x,y,t,\tau)Q(y,\tau)\,ds_y\,d\tau$$

and the double layer potential as

$$(\mathcal{K}_t * P)(x,t) = \lim_{\varepsilon\to 0} \int_0^t \int_{\Gamma\setminus B_\varepsilon(x)} \frac{\partial P_t(x,y,t,\tau)}{\partial n_y} P(y,\tau)\,ds_y\,d\tau.$$

The integral free term $\mathcal{C}(x)$ turns out to be the same as in (4.18). Moreover, we notice that the structure of (4.19) and (4.20) are identical up to the convolution integral.

Exterior problems Up to now we have treated interior problems, i.e., we look for the solution in Ω. Do the representation formulae and the boundary integral equations change if we look for the solution in Ω^e? Recall, Sommerfeld's radiation condition (4.8). In order to impose this condition we introduce an artificial boundary at infinity, say $\Gamma^\infty = \partial\mathbb{R}^3$. If we apply the representation formulae to both boundaries, Γ and Γ^∞, and if we impose the radiation condition to Γ^∞, we end up with only a sign change in the representation formulae and only a sign change in front of the integral free term in the boundary integral equations [91].

Solvability There exist frequencies, when solvability or at least uniqueness of boundary value problems for the Helmholtz equation is not given [91].

The single layer operator $\mathcal{V}_s$ (4.16) is singular if k^2 corresponds to an eigenvalue of the corresponding Dirichlet eigenvalue problem. Since $\mathcal{V}$ is the same for interior and exterior problems, Dirichlet problems are uniquely solvable whenever such frequencies are excluded. The double layer operator $\mathcal{C}+\mathcal{K}_s$ (4.18) and (4.17) is not invertible if k^2 corresponds to eigenvalues of the corresponding Neumann eigenvalue problem. In all other cases interior Neumann problems have unique solutions. Exterior Neumann problems are not solvable if k^2 is an eigenvalue of the corresponding interior Dirichlet problem, that is when the operator $-\mathcal{C}+\mathcal{K}_s$ becomes singular.

There exist various possibilities to overcome these phenomena of so called *spurious eigenfrequencies*. We do not treat them in this work, we refer the reader to [91] to find a detailed listing and elaboration of the solvability of various boundary value problems.

4.3 Convolution quadrature method

This section affects only time dependent acoustic problems. The target is the convolution operator $(*)$ in the time-domain boundary integral equation (4.20). Let us forget about the spatial discretization for the moment. We focus on the temporal discretization of the convolution integral in time. All proposed methodologies can be split in two groups: The computation of the convolution integrals is either performed directly in time domain (e.g., [72, 38]) or an inverse transformation is combined with a computation in the Laplace domain. Both groups react very sensitively on the chosen parameter settings.

We are going to use the convolution quadrature method (CQM) which has initially been developed by Lubich [69, 70, 71] and was applied to the boundary element method in [87]. The CQM can be ranged somewhere inbetween both above mentioned groups, it exploits their merits and results in a stable time stepping procedure. Aside from that, the very reason, we use this temporal discretisation scheme, is the fact that it relies on the Laplace domain counterpart of the time domain fundamental solution. In the case of the wave equation this is the fundamental solution (4.13) of the Helmholtz equation. Since in chapter 2 and 3, we have constructed fast methods for exactly that type of matrix kernel, this favours the CQM.

4.3.1 Discrete convolution

In the following, only those parts of its theoretical framework which are necessary for the understanding of the present work are recalled. The very basic idea of the CQM is the

discrete approximation of a convolution integral

$$y(t) = p * q = \int_0^t p(t-\tau)q(\tau)\,d\tau \tag{4.21}$$

with $t \in [0,T] \subset R^+$ by means of a quadrature scheme

$$y(n\Delta t) = y_n \sim \sum_{i=0}^{n} \omega_{n-i}^{\Delta t}(\hat{p})\, q_i$$

with $q_i = q(i\Delta t)$ and the Laplace transformed counterpart $\hat{p}$ of p. The time interval $[0,T]$ is split in $N+1$ intervals of equal length $\Delta t = T/N$. The quadrature weights $\omega_{n-i}^{\Delta t}(\hat{p})$ depend on the underlying multistep method which we will address in the following.

Convolution weights Let us derive the explicit formula for computing the convolution weights as presented in [69]. We start with the substitution of the $p(t-\tau)$ in (4.21) with its inverse Laplace transform

$$y(t) = \frac{1}{2\pi\imath} \int_{c-\imath\infty}^{c+\imath\infty} \hat{p}(s) \underbrace{\int_0^t \mathrm{e}^{s(t-\tau)} q(\tau)\,d\tau}_{x(t)}\, ds, \tag{4.22}$$

with $s \in \mathbb{C}$ and an appropriately chosen value $c \in \mathbb{R}^+$. The function $x(t)$ is the solution of the ordinary differential equation of first order

$$x'(t) - s\,x(t) = q(t) \quad \text{for } x(0) = 0. \tag{4.23}$$

This is where the first of two approximations comes in to play: The numerical solution of the above initial value problem is well studied in literature (e.g., [60]). As presented in [4] we will deploy A- and L-stable Runge-Kutta methods here. They are classically given by their Butcher tableaus $\begin{array}{c|c} \mathsf{c} & \mathsf{A} \\ \hline & \mathsf{b} \end{array}$ with $\mathsf{A} \in \mathbb{R}^{m\times m}$ and $\mathsf{c}, \mathsf{b}^T \in \mathbb{R}^m$ and m is the number of stages. Note, if Runge-Kutta methods are used for solving (4.23), results are not only computed at equally spaced timesteps $n\Delta t$ but also at intermediate stages, i.e., at $\{(n + (\mathsf{c})_i)\Delta t : i = 1,\ldots,m\}$. The solution reads as

$$\begin{aligned} x_{n+1} &= x_n + \Delta t\,\mathsf{b}^T(s\mathsf{X}_n + \mathsf{q}_n) \\ \mathsf{X}_n &= x_n\mathbb{1} + \Delta t\,\mathsf{A}(x\mathsf{X}_n + \mathsf{q}_n). \end{aligned} \tag{4.24}$$

Here $x_n \in \mathbb{C}$ is the solution at $n\Delta t$ and $\mathsf{X}_n \in \mathbb{C}^m$ are intermediate solutions at the stages and $\mathsf{q}_n \in \mathbb{R}^m$ is a vector with the entries $(\mathsf{q}_n)_i = q((n + (\mathsf{c})_i)\Delta t)$ for $i = 1,\ldots,m$. Due to the

A- and L-stability of the used Runge-Kutta methods $\mathsf{b}^T\mathsf{A}^{-1} = (0,...,1)$ holds. Hence, with (4.24) the solution at each timestep can be computed by the solution at the intermediate stages by

$$x_{n+1} = \mathsf{b}^T\mathsf{A}^{-1}\mathsf{X}_n. \tag{4.25}$$

For a straightforward use of the above solution in the CQM we would need an relation like $x_n = f(q_n)$, which, however, is not possible to derive. We can only establish a relation for the intermediate solutions X_n via a power series, which reads as

$$\sum_{n=0}^{\infty} \mathsf{X}_n z^n = \left(\frac{\Delta(z)}{\Delta t} - s\,\mathsf{I}\right)^{-1} \sum_{n=0}^{\infty} \mathsf{q}_n z^n, \tag{4.26}$$

where $\Delta : \mathbb{C} \to \mathbb{C}^{m\times m}$ is the characteristic function which reads as

$$\Delta(z) = \left(\mathsf{A} + \frac{z}{1-z}\mathbb{1}\mathsf{b}^T\right)^{-1}.$$

By using (4.25) and (4.26) we finally get a relation between x_{n+1} and q_n. Since we have to plug it into (4.22) we need to multiply it by z^n and sum over all n. It yields to

$$\begin{aligned}\sum_{n=0}^{\infty} y_{n+1} z^n &= \frac{1}{2\pi\imath} \int_{c-\imath\infty}^{c+\imath\infty} \hat{p}(s)\,\mathsf{b}^T\mathsf{A}^{-1} \left(\frac{\Delta(z)}{\Delta t} - s\,\mathsf{I}\right)^{-1} \sum_{n=0}^{\infty} \mathsf{q}_n z^n \\ &= \mathsf{b}^T\mathsf{A}^{-1}\hat{p}\left(\frac{\Delta(z)}{\Delta t}\right) \sum_{n=0}^{\infty} \mathsf{q}_n z^n.\end{aligned} \tag{4.27}$$

In the last step, the Cauchy's residue theorem has been applied. Note, the argument of $\hat{p}$ contains the characteristic function $\Delta(z)$, hence a matrix. Consequently, the resulting expression is a matrix as well.

The next task is to get rid of the sum on the left hand side of (4.27). To do so, we introduce the power series expansion

$$\hat{p}\left(\frac{\Delta(z)}{\Delta t}\right) = \sum_{n=0}^{\infty} \mathsf{W}_n^{\Delta t}(\hat{p})\, z^n. \tag{4.28}$$

Inserting the above equation in (4.27) results in the product of two infinite power series. By using the Cauchy product it can be rewritten as a discrete convolution. And, finally, by comparing coefficients we end up with

$$y_{n+1} = \mathsf{b}^T\mathsf{A}^{-1} \sum_{k=0}^{n} \mathsf{W}_{n-k}^{\Delta t}(\hat{p})\, \mathsf{q}_k. \tag{4.29}$$

The convolution weights $\mathsf{W}_n^{\Delta t}$ are not yet defined. Usually an explicit representation can not be derived from (4.28), however, they can be defined based on Cauchy's integral formula as

$$\mathsf{W}_n^{\Delta t}(\hat{p}) = \frac{1}{2\pi\imath} \oint_C \hat{p}\left(\frac{\Delta(z)}{\Delta t}\right) z^{-n-1}\, dz \tag{4.30}$$

where C can be chosen as a circle centered at the origin of radius $R < 1$. Recall, the above expression gives the result at the time $(n+1)\Delta t$.

So far only one approximation, i.e., the Runge-Kutta method for solving the ordinary differential equation (4.23), has been introduced. Now, the second approximation is introduced: The contour integral in (4.30) is approximated via a trapezoidal rule and the convolution weights read as

$$\mathsf{W}_n^{\Delta t}(\hat{p}) = \frac{R^{-n}}{N+1} \sum_{\ell=0}^{N} \hat{p}(\mathsf{s}_\ell)\, \zeta^{n\ell} \quad \text{with} \quad \zeta = \mathrm{e}^{\frac{2\pi\imath}{N+1}} \quad \text{and} \quad \mathsf{s}_\ell = \frac{\Delta(R\zeta^{-\ell})}{\Delta t}. \tag{4.31}$$

Note, s_ℓ is a matrix whose size depends on the characteristic function Δ. Finally, by using the above convolution weights in (4.29) we can approximate the convolution integral by means of a discrete convolution quadrature.

Appart from the fact that Lubich [69] used multistep methods to approximate (4.23) instead of using Runge-Kutta methods (single-step but multistage), this is where the initial derivation of the CQM stopped.

4.3.2 Decoupled convolution

In order to split up the discrete convolution in (4.29) we follow the approach presented by Banjai and Sauter [6]. They proposed to extend the convolution weights W_n^{N+1} to be valid also for negative indices $n < 0$, utilizing the fact that they have to vanish then due to causality. Thus, the sum in (4.29) can be extended to $k = N$. After inserting the convolution weights (4.31) that equation reads as

$$y_{n+1} = \mathsf{b}^T \mathsf{A}^{-1} \frac{R^{-n}}{N+1} \sum_{\ell=0}^{N} \hat{p}(\mathsf{s}_\ell)\, \hat{\mathsf{q}}_\ell\, \zeta^{n\ell}. \tag{4.32}$$

The vectors $\mathsf{q} \in \mathbb{R}^m$ and $\hat{\mathsf{q}} \in \mathbb{C}^m$ are given by

$$(\hat{\mathsf{q}}_\ell)_i = \sum_{k=0}^{N} R^k (\mathsf{q}_k)_i\, \zeta^{-k\ell} \quad \text{and} \quad (\mathsf{q}_k)_i = \frac{R^{-k}}{N+1} \sum_{\ell=0}^{N} (\hat{\mathsf{q}}_\ell)_i\, \zeta^{k\ell} \quad \text{for } i = 1, \ldots, m.$$

They can be considered as a scaled Fourier transform and its inverse counterpart.

System of decoupled Helmholtz problems Let us recall the boundary integral equation of the wave equation

$$\mathcal{C}(x)p(x,t) + (\mathcal{K}_t * p)(x,t) = (\mathcal{V}_t * q)(x,t). \tag{4.33}$$

At this stage, we have derived all tools we need to unwind the convolution integrals in the above equation. By using the decoupled approximation of the convolution from (4.32) we can rewrite the boundary integral operators of the above equation as

$$(\mathcal{V}_t * q)(x,t_{n+1}) \sim \mathsf{b}^T \mathsf{A}^{-1} \frac{R^{-n}}{N+1} \sum_{\ell+1}^{N} (\mathcal{V}_\mathsf{s}\,\hat{\mathsf{q}})_\ell(x)\zeta^{n\ell}$$
$$(\mathcal{K}_t * p)(x,t_{n+1}) \sim \mathsf{b}^T \mathsf{A}^{-1} \frac{R^{-n}}{N+1} \sum_{\ell+1}^{N} (\mathcal{K}_\mathsf{s}\,\hat{\mathsf{p}})_\ell(x)\zeta^{n\ell}$$

with $t_{n+1} = (n+1)\Delta t$. Next, we insert the above expressions in (4.33) and compare coefficients. Finally, the boundary integral equation for the wave equation dissolves in a system of $N+1$ decoupled boundary integral equations of the Helmholtz equation

$$(\mathcal{C} + \mathcal{K}_\mathsf{s}\,\hat{\mathsf{p}})_\ell(x) = (\mathcal{V}_\mathsf{s}\,\hat{\mathsf{q}})_\ell(x). \tag{4.34}$$

Matrix exponentials Recall the definition of $\mathsf{s}_\ell = \Delta(R\zeta^{-\ell})/\Delta t$ from (4.31): It is a matrix whose size is given by the characteristic function $\Delta : \mathbb{C} \to \mathbb{C}^{m\times m}$. If we look at the definition of the integral operators (4.16) and (4.17) of the Helmholtz equation, we notice that s_ℓ essentially defines the complex frequencies the fundamental solution (4.13) has to be evaluated for. Due to its form a matrix exponential of the type $\mathrm{e}^{\mathsf{s}_\ell}$ has to be evaluated. How do we do that? If a matrix is diagonal then its exponential can be obtained by just exponentiating every entry on the main diagonal [47]. Let us assume that s_ℓ is diagonalizable as $\mathsf{s}_\ell = \mathsf{E}\Lambda\mathsf{E}^{-1}$, where the column vectors of E are eigenvectors of s_ℓ, and the corresponding diagonal entry in the diagonal matrix $\Lambda = \operatorname{diag}(\lambda_1,\ldots,\lambda_m)$ is the corresponding eigenvalue. Hence, the matrix exponential in P_s can be computed as

$$P_\mathsf{s}(x,y) = \mathsf{E}\underbrace{\operatorname{diag}\left(P_{\lambda_1}(x,y),\ldots,P_{\lambda_m}(x,y)\right)}_{P_\Lambda(x,y)}\mathsf{E}^{-1},$$

where the Helmholtz fundamental solution $P_\lambda(x,y)$ is evaluated for any eigenvalue λ. By using the above relation, the operators used in (4.34) can be written as $\mathcal{V}_\mathsf{s} = \mathsf{E}\mathcal{V}_\Lambda\mathsf{E}^{-1}$ and $\mathcal{K}_\mathsf{s} = \mathsf{E}\mathcal{K}_\Lambda\mathsf{E}^{-1}$, the equation finally results in

$$\left(\mathsf{E}\,(\mathcal{C}+\mathcal{K}_\Lambda)\,\mathsf{E}^{-1}\hat{\mathsf{p}}\right)_\ell(x) = \left(\mathsf{E}\mathcal{V}_\Lambda\mathsf{E}^{-1}\hat{\mathsf{q}}\right)_\ell(x). \tag{4.35}$$

Let us sum up the cost. We have $N+1$ systems of equations as the one above. Each of them is a coupled system of m equations, this gives us a total number of $m(N+1)$ equations to solve for.

Remark. Recall the definition of the matrix s_ℓ which needs to be diagonalized in order to obtain the complex frequencies needed in (4.35). However, is s_ℓ always diagonalizable? Exceptions are presented in [4]: For example, for the 2-stage and 3-stage Radau IIA method exist only one, respectively, two values for $R\zeta^\ell$ where s_ℓ is not diagonalizable. It is very unlikely to hit these values during a computation. However, it is advisable to investigate the condition number of the matrix s_ℓ.

4.4 Boundary element formulation

Here, the objective is to construct efficient boundary element formulations for acoustics. In the chapters 2 and 3, we have derived a framework of tools to efficiently evaluate many pairwise interactions based on oscillatory kernels. They have the same properties as the fundamental solution P_k of the Helmholtz equation but not as the fundamental solution P_t of the wave equation. This is the reason why we focus on the discretization of the boundary integral equation (4.19) for the Helmholtz equation, hereafter. All operators appearing therein are *elliptic boundary integral operators* and such have been analyzed extensively by many researchers. Our main reference is [91], it contains a very detailed analysis of such operators and their discretization.

On the other hand, operators appearing in the time-domain boundary integral equation (4.20) belong to the family of *hyperbolic boundary integral operators*. They are mathematically less studied and mainly treated in engineering literature yet. An exemplary but not complete listing of studies is given by [72, 2, 38, 34]. Hyperbolic operators require a different treatment then the elliptic operators, thus, diverse boundary element formulations need to be established. However, the temporal discretization scheme we introduced in sec. 4.3, allows us to reformulated (4.20) as a set of decoupled boundary integral equations (4.19) for the Helmholtz equation. Concluding, no matter whether we need to solve time dependent or steady state problems, we can use the discretization of (4.19). In both cases, we have boundary element formulations for the Helmholtz equation.

4.4.1 Weak solution of the boundary value problem

Let us recall the mixed boundary value problem (4.7) for the Helmholtz equation, it reads as

$$\Delta p(x) + k^2 p(x) = 0 \quad \text{for } x \in \Omega, \tag{4.36}$$

with the boundary conditions

$$\begin{aligned} p(x) &= g_D(x) \quad \text{for } x \in \Gamma_D, \\ q(x) &= g_N(x) \quad \text{for } x \in \Gamma_N. \end{aligned} \tag{4.37}$$

Commonly, an analytical solution to the above problem cannot be found. We use boundary element formulations to find the solution numerically. The idea is not to solve the given boundary value problem in its strong form (4.36), but to solve its weak form

$$\int_\Omega w(x)\,\left(\Delta p(x)+k^2 p(x)\right)\,dx = 0 \quad \text{for } x\in\Omega \tag{4.38}$$

for some suitable chosen test functions $w(x)$. It is well known that the resulting solution $p(x)$ satisfying the boundary conditions (4.37) is also a solution to its strong form [91]. Equation (4.38) can also be found as *weighted residual*.

Collocation method

Demanding the weak form (4.38) to hold for suitable test functions $w(x)$ in the integral sense yields the *Galerkin method*. We choose these test functions to be Dirac delta distributions $w(x)=\delta(x)$. Using their screening property $\int g(x)\delta(x-x^*)\,dx = g(x^*)$ yields the *collocation method* and the weak form results in

$$\Delta p(x^*)+k^2 p(x^*) = g(x^*).$$

In other words, the solution $p(x)$ needs to satisfy (4.36) only at a given set of so called collocation points $\{x^*\}$. Note, due to the choice of test functions the collocation method can be considered a special case of the Galerkin method.

Adapting the collocation method to the boundary integral equation (4.19), which we derived to solve the Helmholtz equation for given boundary conditions, results in a system of equations

$$\left((\mathcal{C}+\mathcal{K}_k)p\right)(x^*) = (\mathcal{V}_k q)(x^*) \tag{4.39}$$

which holds for the Cauchy data $p(x)$ and $q(x)$ at all collocation points $\{x^*\}\subset\Gamma$. Recall, if they were exact solutions of the problem, the boundary integral equation (4.19) would be true everywhere on the boundary and not just at the collocation points.

Up to now, we have as many equations as we have collocation points. However, before we focus on them we need to further investigate the Cauchy data. Given is the mixed problem (4.37), this means that $p(x)$ and $q(x)$ are partly known, partly unknown on the boundary. To find the unknown parts, we introduce the decomposition

$$p(x) = \tilde{p}(x)+\tilde{g}_D(x) \quad \text{and} \quad q(x) = \tilde{q}(x)+\tilde{g}_N(x) \quad \text{for } x\in\Gamma, \tag{4.40}$$

with the unknown Dirichlet and Neumann data $\tilde{p}$, respectively, $\tilde{q}$. The quantities $\tilde{g}_D$ and $\tilde{g}_N$ are arbitrary but fixed extensions of the given Cauchy data to the whole boundary [91].

We insert (4.40) in (4.39) and obtain the following equation where known and unknown Cauchy data are separated

$$(\mathcal{V}_k \tilde{q})(x^*) - (\mathcal{C} + \mathcal{K}_k)\,\tilde{p})\,(x^*) = f(x^*),$$

with the right hand side

$$f(x^*) = ((\mathcal{C} + \mathcal{K}_k)\,\tilde{g}_D)\,(x^*) - (\mathcal{V}_k\,\tilde{g}_N)(x^*).$$

Spatial discretization Next, the discretization of the boundary Γ is introduced. It is commonly given by a set of E boundary elements τ. Usually they do not exactly represent the original boundary but give just an approximation as

$$\Gamma \sim \Gamma_h = \bigcup_{e=1}^{E} \tau_e,$$

where τ_e denotes the e^{th} boundary element having an average size h. Given the domain $\Omega \in \mathbb{R}^3$, its boundary Γ is a 2-dimensional mannifold, hence boundary elements are usually either triangles or quadrilaterals. These geometrical entities can be accessed via a coordinate transformation $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ from a reference element $\bar{\tau}$ to the global element τ.

Cauchy data approximation We introduce the approximation of the Cauchy data as

$$p(x) \sim \sum_{i=1}^{M} (\mathsf{p})_i\,\varphi_i(x) \quad \text{and} \quad q(x) \sim \sum_{j=1}^{N} (\mathsf{q})_j\,\psi_j(x).$$

In other words, the discrete solution to the Cauchy data is represented by the vectors $\mathsf{p} \in \mathbb{C}^M$ and $\mathsf{q} \in \mathbb{C}^N$. The basis functions φ and ψ are defined on the discrete boundary Γ_h. What properties do they have? From physics we know that the acoustic pressure p is continuous, whereas the acoustic flux q, being related to the normal vector on the boundary, is not neccessarily continuous. Think of an arbitrary geometry having edges and corners, there the normal vector on the boundary has jumps. Generally speaking, most approximations of smooth boundaries, e.g., spheres, become non smooth. These restrictions from physics give rise to discrete boundary element spaces

$$S_h^+(\Gamma_h) = \operatorname{span}\{\varphi_i\}_{i=1}^{M} \quad \text{and} \quad S_h^-(\Gamma_h) = \operatorname{span}\{\psi_j\}_{j=1}^{N}, \tag{4.41}$$

where S_h^+ contains the continuous Dirichlet datum p and S_h^- the discontinuous Neumann datum q on the entire discrete boundary Γ_h.

Algebraic equation system The continuous decomposition of the Cauchy data in (4.40) holds also for its discrete counterparts p and q. The vector p containing the discrete Dirichlet datum is split in $\tilde{\mathsf{p}} \in \mathbb{C}^{M_N}$ and $\mathsf{g}_D \in \mathbb{C}^{M_D}$. Equally, the vector q containing the discrete Neumann datum is split in $\tilde{\mathsf{q}} \in \mathbb{C}^{N_D}$ and $\mathsf{g}_N \in \mathbb{C}^{N_N}$. Note, the size of the vectors containing the complete discrete Cauchy data is still $M = M_D + M_N$, respectively, $N = N_D + N_N$.

What else do we need to obtain an algebraic equation system? The distribution of the collocation points x^* comes into play. We have the unknown vectors $\tilde{\mathsf{p}}$ of size M_N and $\tilde{\mathsf{q}}$ of size N_D. We need the same number of equations. We require (4.39) to hold at the collocation points $\{x_i^* : i = 1, \ldots, N_D\} \subset \Gamma_{D,h}$, respectively at the collocation points $\{x_j^* : j = 1, \ldots, M_N\} \subset \Gamma_{N,h}$. This yields the equation system

$$\begin{pmatrix} \mathsf{V}_D & -\mathsf{K}_D \\ \mathsf{V}_N & -\mathsf{K}_N \end{pmatrix} \begin{pmatrix} \tilde{\mathsf{q}} \\ \tilde{\mathsf{p}} \end{pmatrix} = \begin{pmatrix} \mathsf{f}_D \\ \mathsf{f}_N \end{pmatrix} \tag{4.42}$$

with the discrete single layer potentials $\mathsf{V}_D \in \mathbb{C}^{N_D \times N_D}$, $\mathsf{V}_N \in \mathbb{C}^{M_N \times N_D}$ and double layer potentials $\mathsf{K}_D \in \mathbb{C}^{N_D \times M_N}$, $\mathsf{K}_N \in \mathbb{C}^{M_N \times M_N}$ and the right hand side vectors $\mathsf{f}_D \in \mathbb{C}^{N_D}$ and $\mathsf{f}_N \in \mathbb{C}^{M_D}$. The entries are computed as

$$\begin{aligned}
(\mathsf{V}_D)_{ij} &= (\mathcal{V}_k \psi_j)(x_i^*) && \text{for } \psi \in S_h^-(\Gamma_D), x^* \in \Gamma_D \\
(\mathsf{V}_N)_{ij} &= (\mathcal{V}_k \varphi_j)(x_i^*) && \text{for } \varphi \in S_h^+(\Gamma_N), x^* \in \Gamma_D \\
(\mathsf{K}_D)_{ij} &= \left((\mathcal{C} + \mathcal{K}_k)\psi_j\right)(x_i^*) && \text{for } \psi \in S_h^-(\Gamma_D), x^* \in \Gamma_N \\
(\mathsf{K}_N)_{ij} &= \left((\mathcal{C} + \mathcal{K}_k)\varphi_j\right)(x_i^*) && \text{for } \varphi \in S_h^+(\Gamma_N), x^* \in \Gamma_N
\end{aligned}$$

The entries of the right hand side vectors f_D and f_N are computed analogously.

The domain of integration in the above integrals corresponds to the support of the respective basis funtion. That is where it is evaluated to non-zero entries. Recall, ψ belongs to S_h^-, i.e., the discontinuous boundary element space for the Neumann datum. Hence, the support of ψ is strictly associated to the corresponding boundary element τ, no matter what polynomial order ψ has. On all other boundary elements the discontinuous ψ is evaluated to zero. This is different for φ, which belongs to S_h^+, the continuous boundary element space for the Dirichlet datum. Its support usually affects the set of neighboring elements. For more information we refer the reader to [43] or [91].

Solution procedure For the solution of the block matrix system (4.42) we apply a similar nested solution procedure as presented in [91] for such block systems. By inserting the first equation

$$\tilde{\mathsf{q}} = \mathsf{V}_D^{-1}\left(\mathsf{f}_D + \mathsf{K}_D \tilde{\mathsf{p}}\right)$$

into the second, we obtain a Schur complement system

$$\underbrace{\left(\mathsf{V}_N\mathsf{V}_D^{-1}\mathsf{K}_D - \mathsf{K}_N\right)}_{\mathsf{S}}\tilde{\mathsf{p}} = \underbrace{\mathsf{f}_N - \mathsf{V}_N\mathsf{V}_D^{-1}\mathsf{f}_D}_{\mathsf{y}}.$$

The goal is to solve the resulting system $\mathsf{S}\tilde{\mathsf{p}} = \mathsf{f}_D$ for the unknown vector $\tilde{\mathsf{p}}$. Instead of performing a direct inversion of V_D we deploy a nested iterative solution procedure. First, we compute the right hand side

$$\mathsf{y} = \mathsf{f}_N - \mathsf{V}_N\,\mathsf{c}$$

with the solution c of $\mathsf{V}_D\,\mathsf{c} = \mathsf{f}_D$. Then, the matrix-vector product for the Schur complement system is defined as

$$\mathsf{S}\,\tilde{\mathsf{p}} = \mathsf{V}_N\,\mathsf{b} - \mathsf{K}_N\,\tilde{\mathsf{p}}$$

with the solution b of $\mathsf{V}_D\,\mathsf{b} = \mathsf{K}_D\,\tilde{\mathsf{p}}$. Neither V_D nor S are symmetric.

5 NUMERICAL EXAMPLES

In this chapter we present numerical examples of acoustic computations. We bring together fast summation schemes for oscillatory kernels presented in chap. 3 and the boundary element formulation presented in sec. 4.4. This yields to *fast boundary element formulations* which open the door to the numerical simulation of large-scale acoustic problems.

The present section is split in two parts: In sec. 5.1, we present an example of an initial-boundary value problem for the wave equation. In sec. 5.2, we present two examples of exterior boundary value problems for the Helmholtz equation. All examples were solved using the C++ software library HyENA [77].

5.1 Hierarchical matrices and Runge-Kutta CQM

An acoustic fluid is given in the domain

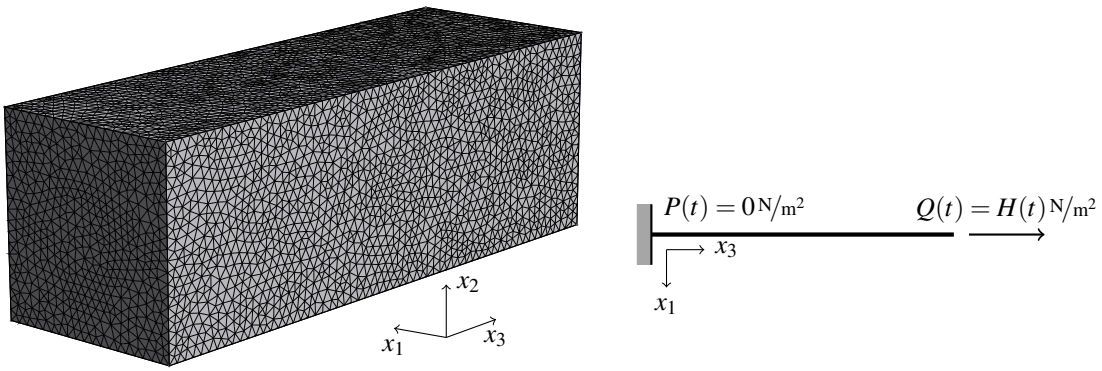
$$\Omega = \{x \in \mathbb{R}^3 : x_1, x_2 \in [-0.5\,\mathrm{m}, 0.5\,\mathrm{m}], x_3 \in [0\,\mathrm{m}, 3\,\mathrm{m}]\}$$

as depicted in fig. 5.1. The speed of sound is set to $c = 346\,\mathrm{m/s}$ and corresponds to the one of air. The mixed initial-boundary value problem reads as

$$\begin{aligned}
\Delta P(x,t) - \frac{1}{c^2}\left(\frac{\partial^2 P}{\partial t^2}\right)(x,t) = 0 \qquad & \text{for } x \in \Omega,\ t > 0, \\
P(x,t) = 0 \qquad & \text{for } x \in \Gamma_D,\ t > 0, \\
Q(x,t) = G_N(x,t) \qquad & \text{for } x \in \Gamma_N,\ t > 0. \\
P(x,0) = \left(\frac{\partial P}{\partial t}\right)(x,0) = 0 \qquad & \text{for } x \in \Omega,
\end{aligned}$$

with zero initial conditions and mixed boundary conditions.

When looking at the system shown in fig. 5.1b we note that the front face is the Dirichlet boundary $\Gamma_D = \{x \in \mathbb{R}^3 : x_1, x_2 \in [-0.5\,\mathrm{m}, 0.5\,\mathrm{m}], x_3 = 0\} \subset \Gamma$ with zero acoustic pressure $P = 0$ at any time. The remaining part of the boundary is the Neumann boundary $\Gamma_N = \Gamma \cap \Gamma_D$. However, only the subset $\{x \in \mathbb{R}^3 : x_1, x_2 \in [-0.5\,\mathrm{m}, 0.5\,\mathrm{m}], x_3 = 3\,\mathrm{m}\} \subset \Gamma_N$ has non-zero acoustic flux $Q(t) = H(t)\,\mathrm{N/m^2}$, where $H(t)$ is the Heaviside step function.



(a) 12032 triangular boundary elements (b) Mixed boundary conditions

Figure 5.1: Acoustic problem

5.1.1 Setup of the numerical method

We deploy the CQM (see sec. 4.3) to perform the temporal discretization. It leads to a system of decoupled Helmholtz problems (see sec. 4.3.2) and we can deploy the $\mathcal{H}$-matrix approach (see sec. 3). Therefore, we used the C++ library AHMED [11]. For the spatial discretization we use the collocation method. Therefore, the boundary is approximated by 12032 linear triangular elements as depicted in fig. 5.1a. Moreover, we use linear continuous basis functions to approximate the Dirichlet datum and piecewise constant basis functions to approximate the Neumann datum. We end up solving for the unknown vectors $\tilde{\mathrm{p}}$ and $\tilde{\mathrm{q}}$ of size $N = 5529$, respectively, $M = 896$ (the Neumann datum is unknown only on Γ_D, i.e., the front face). In order to get an algebraic equations system as presented in (4.42), we need to collocate on Γ_D at 896 points located in the center of the boundary elements and on Γ_N at 5529 points located on the vertices of the boundary elements (the same collocation point is shared by all boundary elements possessing the same vertex).

In the following, the behavior of the CQM with respect to the chosen Runge-Kutta method is numerically studied. We compare Runge-Kutta methods like the 2- and 3-stage Radau IIA and the 4-stage Lobatto IIIC method. The respective Butcher tables can be found in [60]. Moreover, the efficiency of the $\mathcal{H}$-matrix approach for the Runge-Kutta methods is compared to BDF2 (backward differential formula of order 2, a single-stage but multi-step method). A similar study, however, for elastodynamics, a symmetric Galerkin BEM and only the BDF2 has been presented in [75].

Two parameters have to be set in the CQM. The first one is the radius, we chose it to be $R = \sqrt{\varepsilon}$ with $\varepsilon = 10^{-5}$ [69]. The total number of time steps of equal length Δt is given by N_T. The overall time is given by $N_T \Delta t = 8.67 \cdot 10^{-2}$ seconds. The second parameter is the time step size Δt. It is natural to set the speed of sound in relation to the size of the boundary elements, which is here $h = 0.05$. Commonly used is the dimensionless

Courant-Friedrichs-Lewy number β. It tells us how many boundary elements are traversed by a sound wave in one time step. A remark to our notation hereafter: We associate the overall number of problems $N_T(\beta)$ to a given β. For example, if we use BDF2, which is a single-stage method, and $\beta = 0.1$ a total of $N_T(0.1) = 6000$ time steps are required to cover the overall time of $8.67 \cdot 10^{-2}$ seconds. This involves the solution of 6000 decoupled Helmholtz problems. On the other hand, if we use the 2-stage Radau IIA method and the same β only $N_T(0.1)/2 = 3000$ twice as large time steps are required to cover the same time. The difference is that in the latter case 3000 systems of 2 problems each, have to be solved. Finally, this amounts to the same overall cost of $N_T(0.1) = 6000$ problems. That is, why we use the definition

$$\beta = \frac{c\Delta t}{mh},$$

where m denotes the number of stages of the used Runge-Kutta method. In the case of BDF2 the number of stages is $m = 1$. In the following, we present numerical results for the range of $\beta \in [0.1, 1.2]$.

Remark. If we look at the definition (4.31) of the complex frequencies s_ℓ we notice that they traverse the complex plane symmetrically with respect to the real axis (see fig. 5.7). In other words, there exist $N_T(\beta)/2$ complex conjugate frequencies pairs which yield to analogous solutions. We solve only for that part of the frequency pair with $\mathrm{Im}\, s_\ell \geq 0$ and reuse the result for the remaining part. Hence, we have to effectively solve only for $N_T(\beta)/2$ problems.

5.1.2 Behavior of the Runge-Kutta CQM

The following results have also been published in [7]. We focus on the acoustic flux at the most central element on the Dirichlet boundary Γ_D. It represents the most meaningful result, as we will see in a moment. The numerical solutions for the overall time $T_\beta \Delta t = 8.67 \cdot 10^{-2}\,\mathrm{s}$ are shown in the figs. 5.2, 5.3, 5.4 and 5.5 for BDF2, 2-stage and 3-stage Radau IIA and 4-stage Lobatto IIIC, respectively. We compare them for varying β values and study the efficiency of the $\mathcal{H}$-matrix approach. In the following, we use the accuracy of ACA to be $\varepsilon_{\mathrm{ACA}} = 10^{-5}$ and choose $\eta = 0.7$ in the admissibility criterion (3.1).

The analytic solution for the 1-dimensional counterpart of the present problem is a square curve as presented in [49, 86]. Eventually, the resulting square curve makes clear why we study this very academic looking problem and it explains why we analyze it. It is very costly to generate a square curve by means of the superposition of harmonics. That is, very figuratively speaking, what the CQM does. Let us have a closer look to our results. In fig. 5.6, we zoom in to the interval $t \in [0.059\,\mathrm{s}, 0.080\,\mathrm{s}]$ for BDF2 and all Runge-Kutta methods. This limits the curves to only the last peak.

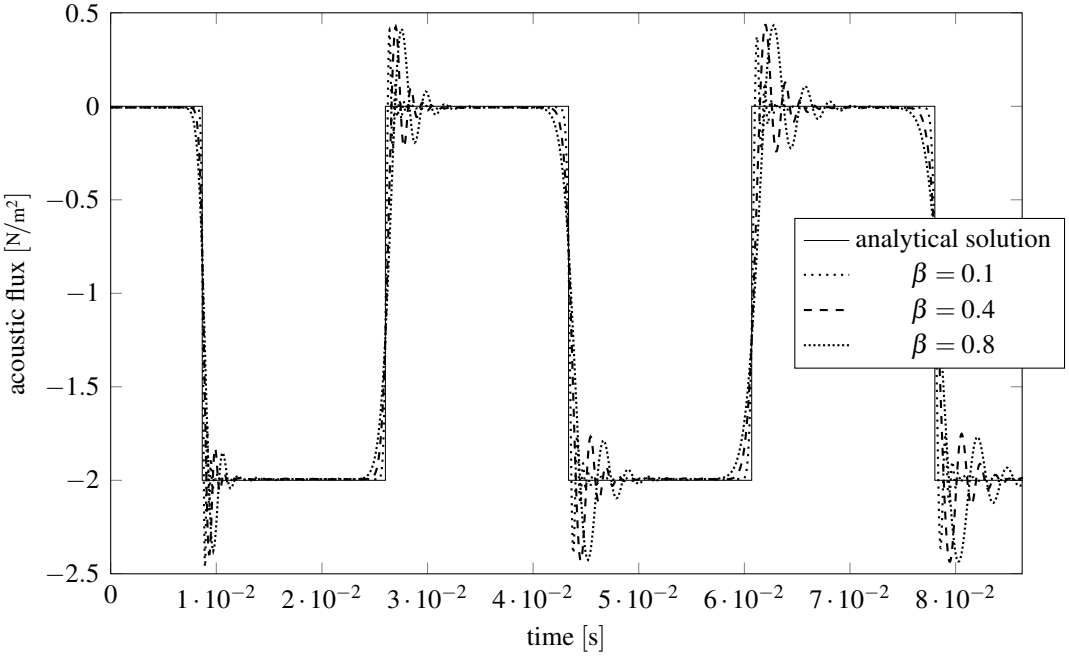


Figure 5.2: BDF2

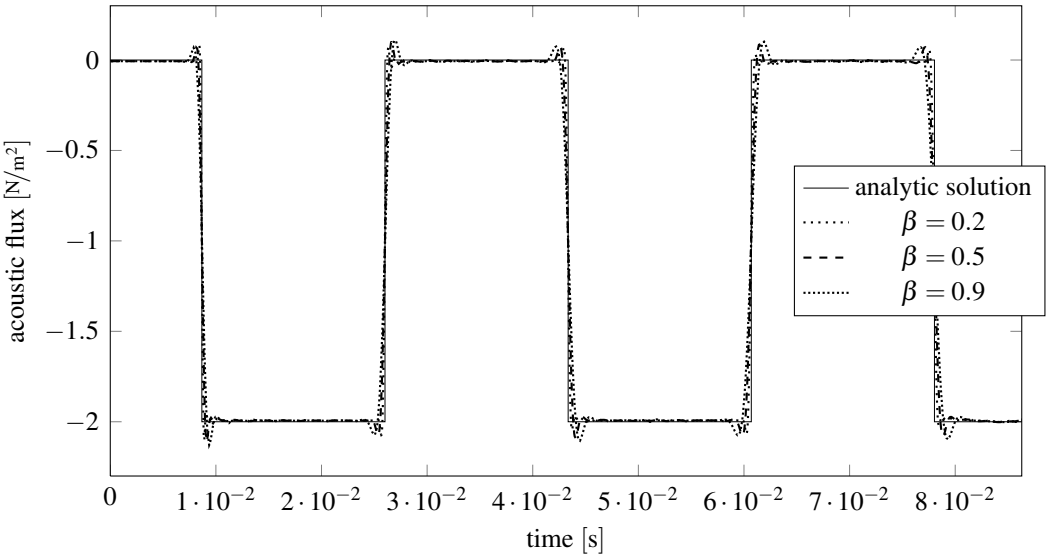


Figure 5.3: Radau IIA 2-stage

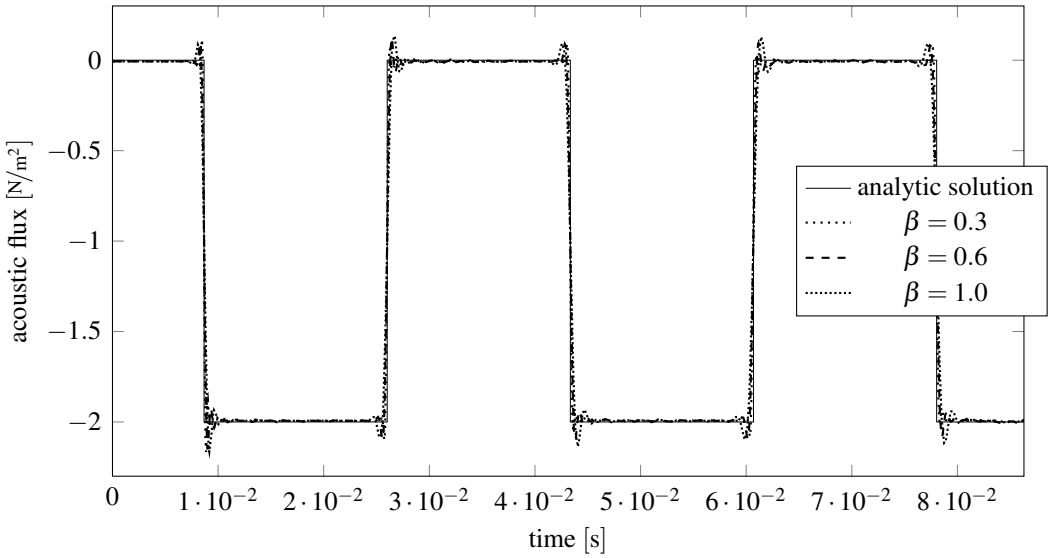


Figure 5.4: Radau IIA 3-stage

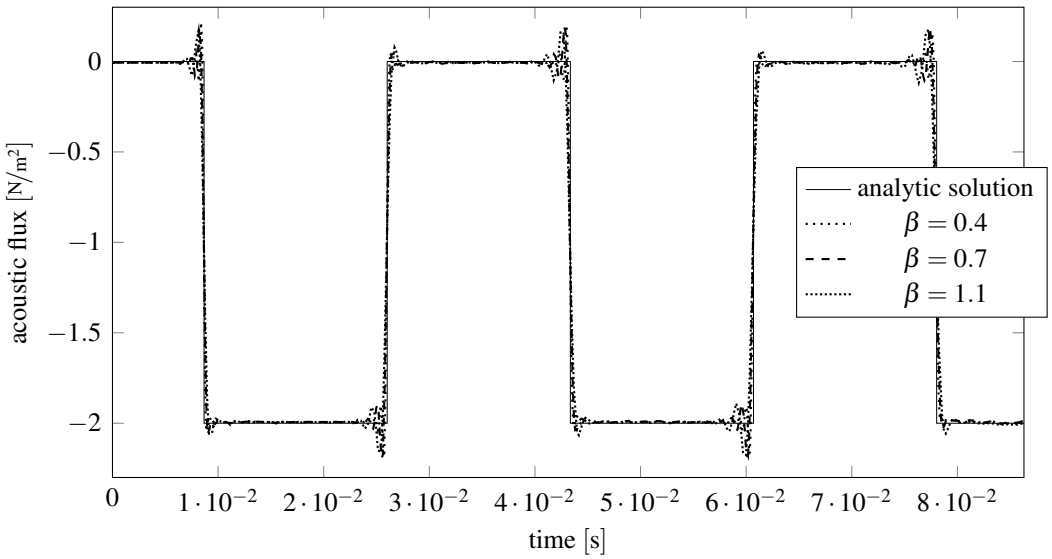
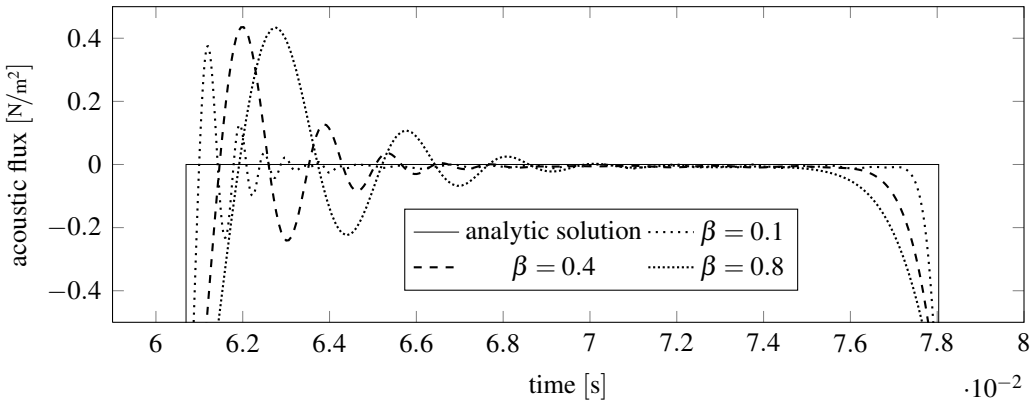
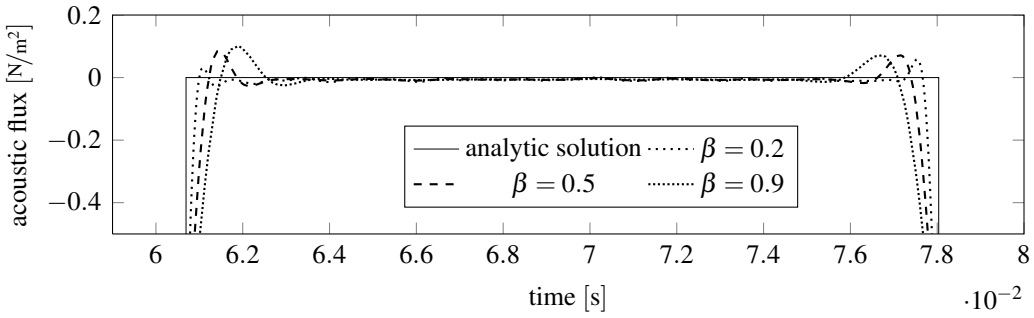


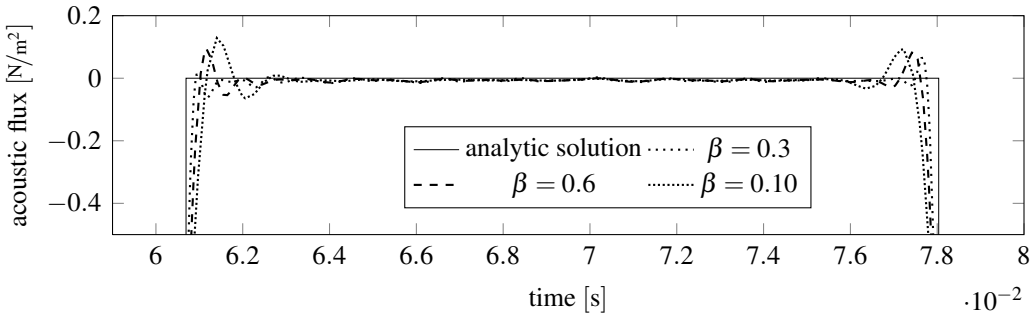
Figure 5.5: Lobatto IIIC 4-stage



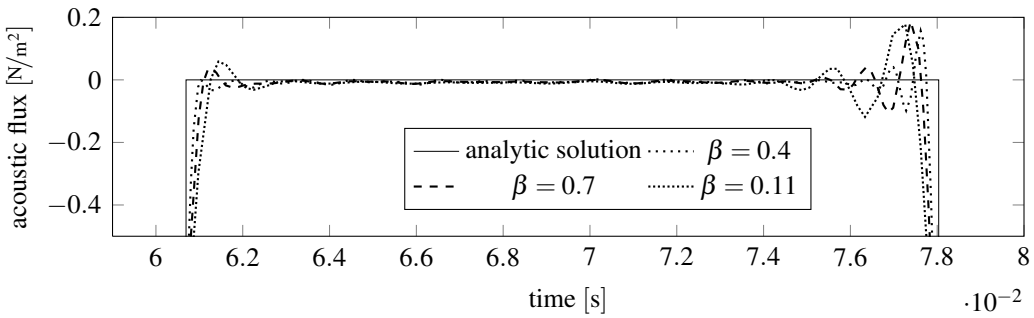
(a) BDF2



(b) Radau IIA 2-stage



(c) Radau IIA 3-stage



(d) Lobatto IIIC 4-stage

Figure 5.6: Comparison of $t \in [0.059\,\mathrm{s}, 0.080\,\mathrm{s}]$

The behavior of truncated Fourier series of piecewise continuously differentiable periodic functions at a jump discontinuities is well studied. They feature large oscillations, which increase the maximum of the truncated series above that of the function itself. The overshoots do not die out as the frequency increases, but approach a finite limit [95]. This is known as Gibb's phenomenon, however commonly known approaches to flatten out these overshoots do not work in the case of the CQM. Only the similarity to Gibbs phenomenon should be pointed out here.

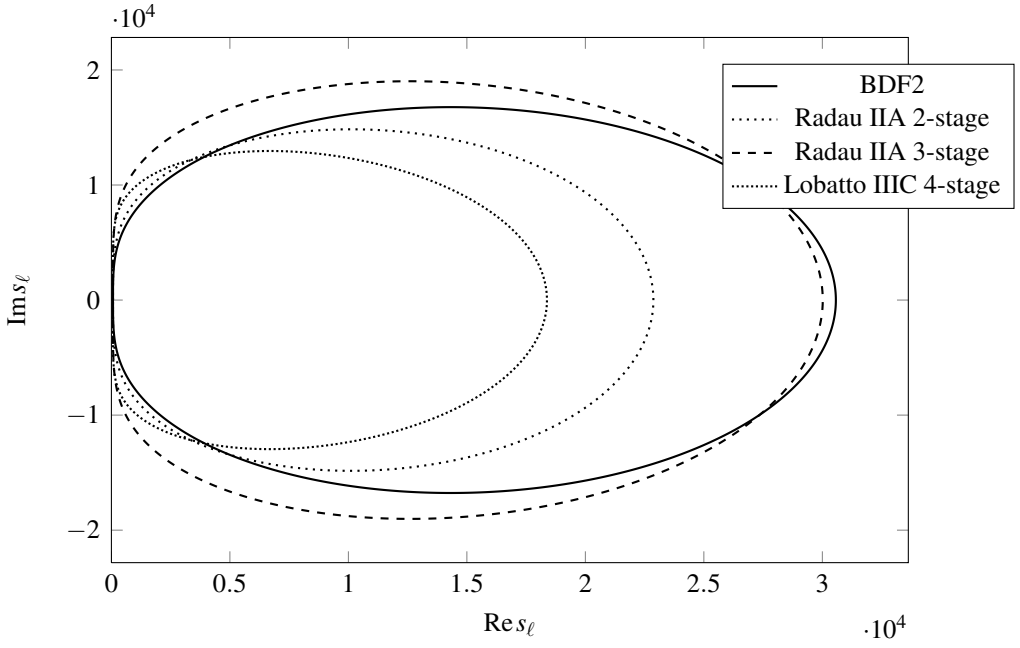
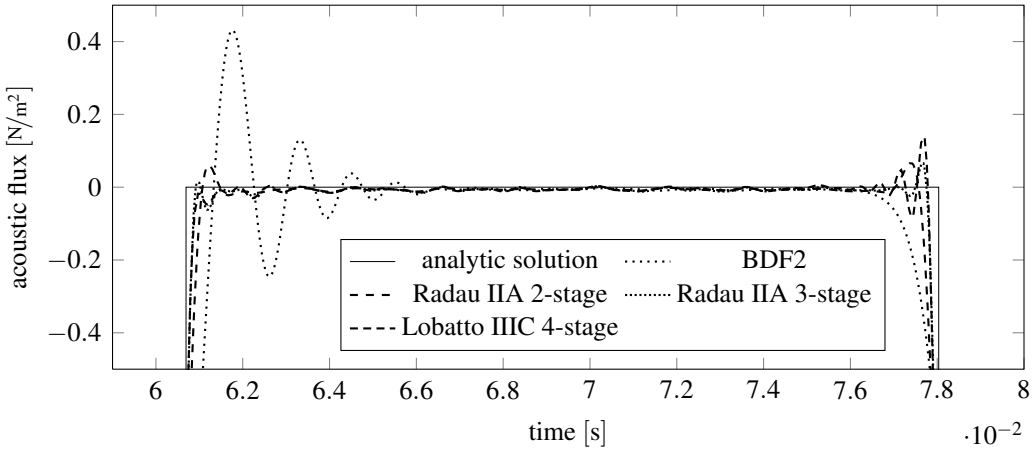


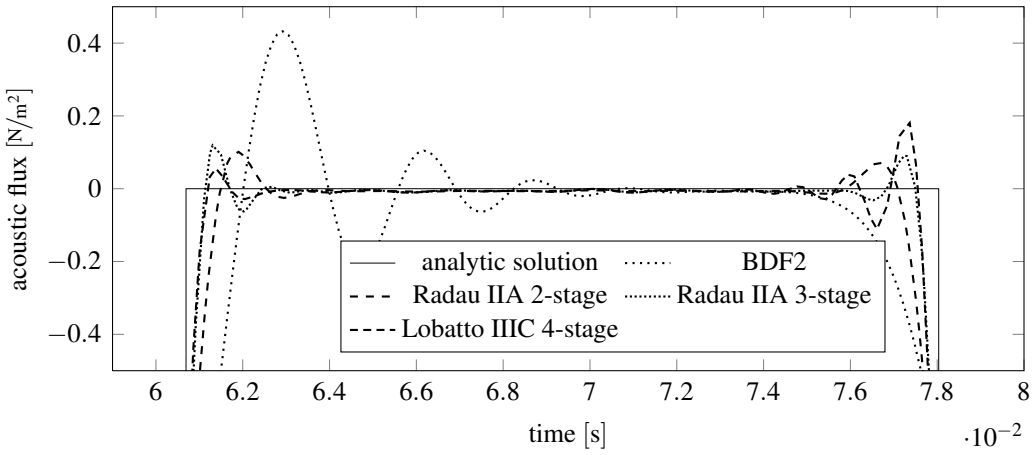
Figure 5.7: Complex frequencies s_ℓ for $\beta = 0.9$

These overshoots can be observed in all figures, however, they behave differently depending on the used time-stepping scheme. Figure 5.7 displays the complex frequencies s_ℓ (4.31) for $\beta = 0.9$ ($T_{0.9} = 666$) and all time-stepping schemes. The ellipses show the arrangement of the resulting $\{s_\ell : \ell = 1, \ldots, T_{0.9}\}$ complex frequencies in the complex plane. We notice that the quality of the resulting square curves depends on the chosen time stepping scheme. For example, if we consider the ellipse for BDF2, we note that the ratio $\mathrm{Re}\, s_\ell / \mathrm{Im}\, s_\ell$ grows faster compared to Runge-Kutta methods. This results in the Runge-Kutta methods to produce better square curves. Especially fig. 5.8, which compares the last peak of the results for $\beta = 0.3$ and $\beta = 0.9$, reflects this behavior.

Computational cost Here, we compare the computational cost with respect to the quality of the numerical result. We count the overall cost in terms of equation systems to be solved. In other words, the solution of $N_T(\beta)$ dense equation systems, each of size N^2,

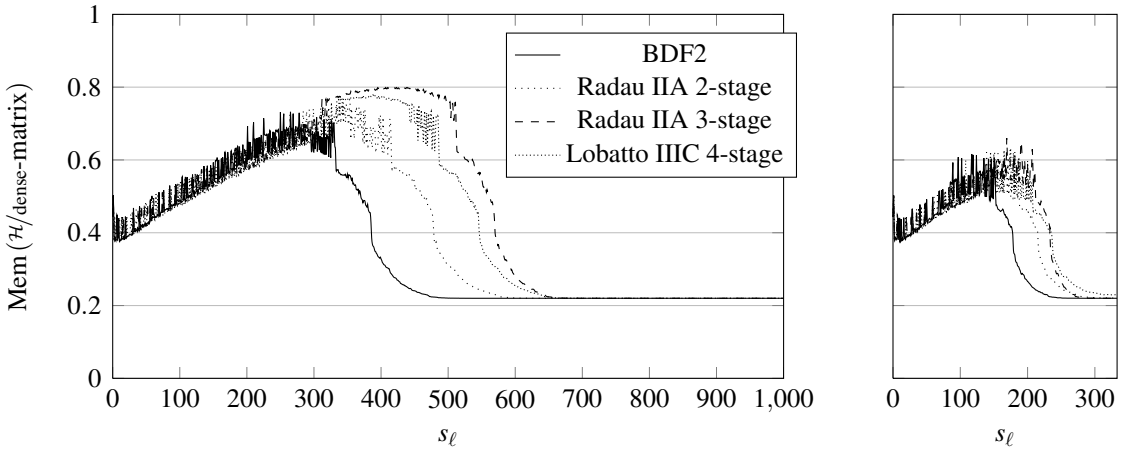


(a) $\beta = 0.3$



(b) $\beta = 0.9$

Figure 5.8: Comparison of interval $t \in [0.059\,\mathrm{s}, 0.080\,\mathrm{s}]$ for all time-stepping schemes



(a) $\beta = 0.3$ (b) $\beta = 0.9$

Figure 5.9: Compression of K_N versus complex frequencies s_ℓ

is equivalent to a cost of $N_T(\beta)$. Recall, we use $\mathcal{H}$-matrices to efficiently compute the solution of the equation system. Hence, if the matrices of the same set of $N_T(\beta)$ equation systems are represented as $\mathcal{H}$-matrices and an average of, say, only $N^2/2$ entries need to be stored, then, the overall cost is equivalent to $N_T(\beta)/2$.

We study the cost in terms of the matrix K_N. It is the largest of the four matrices and has a size of $N = 5529$ rows and columns. Figure 5.9 shows the compression of K_N for BDF2 and all Runge-Kutta methods for $\beta = 0.3$ and $\beta = 0.9$. Recall, we effectively solve only for $N_T(\beta)/2$ problems due to the symmetric arrangement of s_ℓ (see fig. 5.7). That said, fig. 5.9 shows the compression for all $N_T(0.3)/2 = 1000$ and $N_T(0.9)/2 = 333$ problems for $\beta = 0.3$, respectively, $\beta = 0.9$.

Interesting, but at the same time obvious, is on one hand the bad compression in the first half and on the other hand the asymptotic behavior in the second half of the frequency range in fig. 5.9. Let us again consider the ellipses in fig. 5.7, the first frequencies have a fast increasing imaginary part but only a very slow increasing real part. Hence, the matrix kernel becomes very oscillatory but only very little damping is effected due to the small real part. This result in a bad compression. We have already reported on that in sec. 3. In the second half of the frequency range, the real part becomes so large that effectively the far-field looses any influence, it can be set to 0. That is why, the cost in the second half converges to exactly the cost of the near-field.

Optimal setting Let us try to find the optimal setting for our numerical method to solve the present example in terms of efficiency and quality of the result. In fig. 5.11, we have presented the point wise L^2 error ε_{L^2} for BDF2 and all Runge-Kutta methods with respect

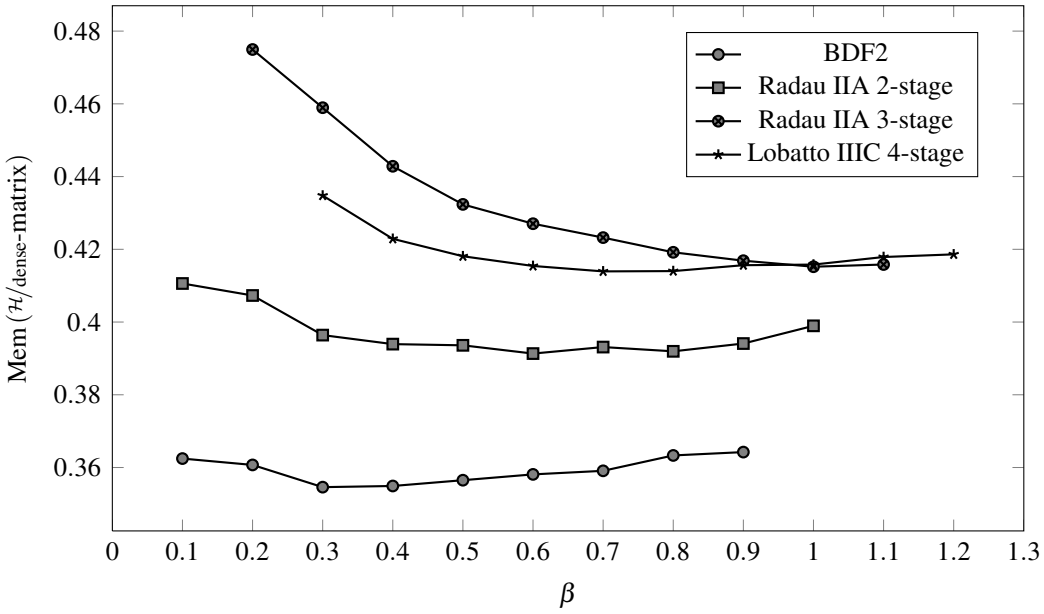


Figure 5.10: Overall compression of K_N

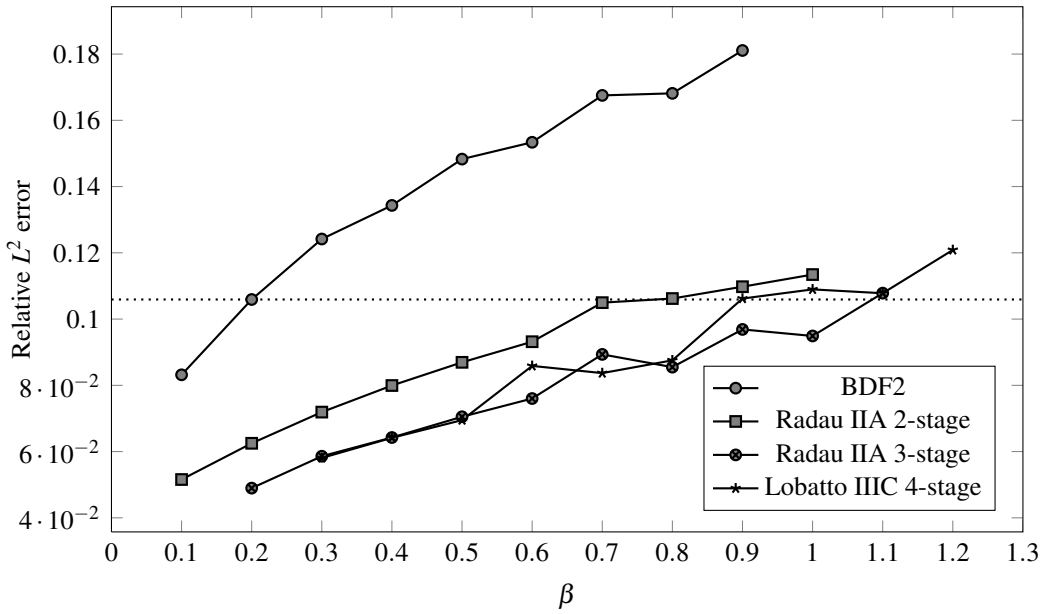


Figure 5.11: Point-wise relative L^2 for different time-stepping schemes

to β. We single out an accuracy $\varepsilon_{L^2} \sim 10^{-1}$ (indicated by the dotted line in fig. 5.11) and align the resulting cost in table 5.1. The third column gives the effective number of problems to be solved. The last column gives the overall cost as defined above. Evidently, the Radau IIA 3-stage method is the best compromise of efficiency and quality of the result. Figure 5.12 shows the same flux results for the four settings presented in table 5.1,

Time stepping scheme	required β	ε_{L^2}	$T_\beta/2$	overall cost
BDF2	0.2	$1.06 \cdot 10^{-1}$	1500	541.4
Radau IIA 2-stage	0.7	$1.05 \cdot 10^{-1}$	432	169.8
Radau IIA 3-stage	1.1	$1.08 \cdot 10^{-1}$	303	125.8
Lobatto IIIC 4-stage	0.9	$1.06 \cdot 10^{-1}$	336	139.6

Table 5.1: Cost for a point-wise relative L^2 error $\varepsilon_{L^2} \sim 10^{-1}$ (see dotted line in fig. 5.11 for required β)

however, for an ACA accuracy $\varepsilon_{ACA} = 10^{-3}$. From $t > 0.060\,\mathrm{s}$ on the resulting square curve starts to break away due to the apparently to bad approximation accuracy.

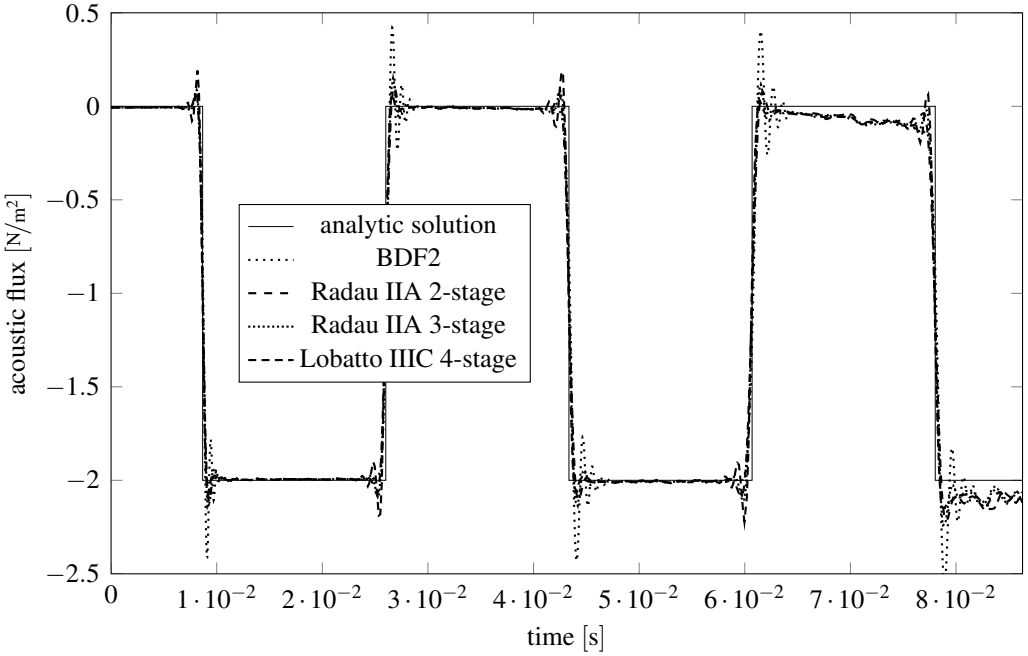


Figure 5.12: Accuracy $\varepsilon_{ACA} = 10^{-3}$

Relation between computational time and required memory We have not given any timing results, because we used the compression to be representative for doing efficiency

studies. Figure 5.13 proofs that this assumption is valid. Indeed the compression of the matrix K_N and the matrix assembly time per frequency s_ℓ behave identically. In the first half of the frequency range a bad compression goes along with large matrix assembly timings, in the second half, where only the near-field needs to be computed, both compression and assembly timings flatten out.

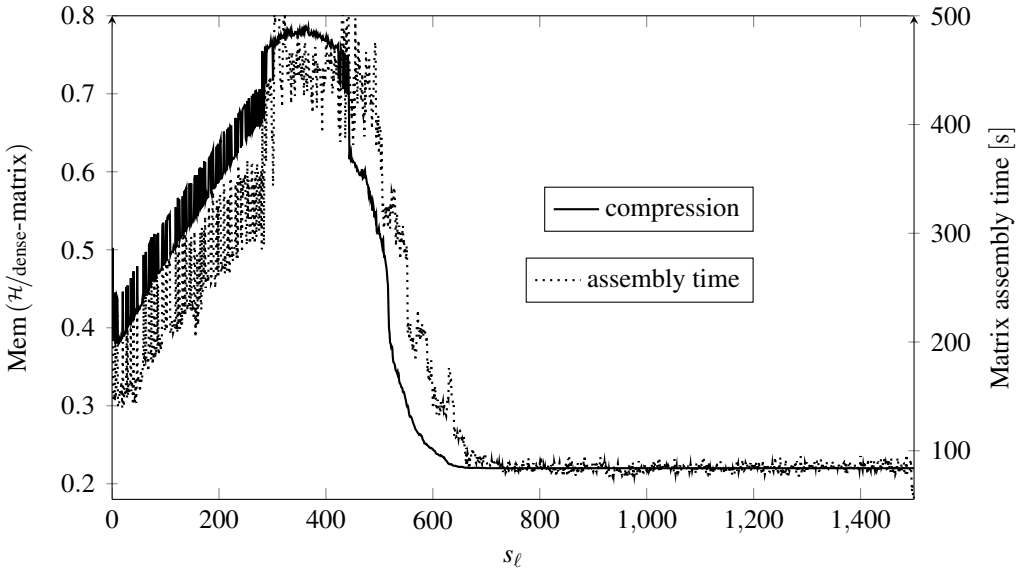


Figure 5.13: Comparison of compression of K_N and matrix assembly time for BDF2 and $\beta = 0.2$

5.2 Directional fast multipole method

In this section, we elaborate how the directional fast multipole method (dFMM) as presented in sec. 3.3 is used to improve the efficiency of the BEM for solving Helmholtz problems, and we present numerical studies. First, we analyze the convergence for Dirichlet and Neumann problems and, then, we present an acoustic scattering problem and the sound radiation from an electric machine.

5.2.1 Realization of the FMM approximation

Here, we assume the real part of the Laplace parameter in (4.6) to vanish, i.e., it becomes $s = \imath\omega$ with the circular frequency ω. Thus, we end up with the more common form of the

Helmholtz equation

$$\Delta p + k^2 p = 0$$

with the wavenumber $k = \frac{\omega}{c} > 0$. Both, the representation formula and the boundary integral equation do not change, they remain the same as (4.14) and (4.19), respectively. With the above met assumptions the fundamental solution simplifies to

$$P_k(x,y) = \frac{1}{4\pi} \frac{\mathsf{e}^{\imath k|x-y|}}{|x-y|}$$

and represents exactly that type of oscillatory kernel we developed the dFMM for (see sec. 3.3). In the following, we elaborate the application of the dFMM to the BEM.

No matter whether we solve a mixed, a Dirichlet or a Neumann boundary value problem for the Helmholtz equation we let the equation system (4.42) be our starting point. Recall, the first block equation $\mathsf{V}_D \tilde{\mathsf{q}} - \mathsf{K}_D \tilde{\mathsf{p}} = \mathsf{f}_D$ holds only on the Dirichlet part of the boundary, whereas the second block equation $\mathsf{V}_N \tilde{\mathsf{q}} - \mathsf{K} \tilde{\mathsf{q}} = \mathsf{f}_N$ holds only on the Neumann part of the boundary. Hence, for solving a Dirichlet problem we use the first equation, otherwise the second one.

In either case we only discretize two boundary integral operators: the single layer potential

$$(\mathcal{V}_k q)(x^*) = \int_\Gamma P_k(x^*,y) q(y) ds_y \tag{5.1}$$

and the double layer potential

$$(\mathcal{K}_k p)(x^*) = \int_\Gamma \frac{\partial P_k(x^*,y)}{\partial n_y} p(y) ds_y \tag{5.2}$$

for all collocation points $x^* \in \Gamma$. Next, we recall the reformulation of the fundamental solution P_k as introduced in sec. 2.3. It reads as

$$P_k(x,y) = \frac{\mathsf{e}^{\imath k(|x-y| - u\cdot(x-y))}}{4\pi|x-y|} \mathsf{e}^{\imath k u\cdot(x-y)} = \mathsf{e}^{\imath k u\cdot x} P_k^u(x,y) \mathsf{e}^{-\imath k u\cdot y}.$$

The modified and directional part P_k^u of the fundamental solution can be interpolated by using Chebyshev interpolation. With it, the approximated fundamental solution reads as

$$P_k(x,y) \sim \mathsf{e}^{\imath k u\cdot x} \sum_{m\in\alpha} S_\ell(x,\bar{x}_m) \sum_{n\in\alpha} P_k^u(\bar{x}_m,\bar{y}_n) S_\ell(y,\bar{y}_n) \mathsf{e}^{-\imath k u\cdot y} \tag{5.3}$$

with the multi-index α with $|\alpha| \leq (\ell+1)^d$. The interpolation operator S_ℓ is elaborated in sec. 2.3.2.

Our aim is not to touch the actual fundamental solution P_k when we approximate the potentials (5.1) and (5.2). For the single layer potential we just need to substitute P_k with its approximant (5.3), and it reads as

$$(\mathcal{V}_k q)(x^*) \sim \mathsf{e}^{\imath ku\cdot x^*} \sum_{m\in\alpha} S_\ell(x^*,\bar{x}_m) \sum_{n\in\alpha} P_k^u(\bar{x}_m,\bar{y}_n) \int_\Gamma S_\ell(y,\bar{y}_n)\, \mathsf{e}^{-\imath ku\cdot y} q(y) ds_y.$$

It works similarly for the double layer potential. Instead of applying the normal derivative to P_k directly we apply it to its interpolation (5.3) and substitute it into the double layer potential

$$(\mathcal{K}_k p)(x^*) \sim \mathsf{e}^{\imath ku\cdot x^*} \sum_{m\in\alpha} S_\ell(x^*,\bar{x}_m) \sum_{n\in\alpha} P_k^u(\bar{x}_m,\bar{y}_n) \int_\Gamma \frac{\partial}{\partial n_y} \left(S_\ell(y,\bar{y}_n)\, \mathsf{e}^{-\imath ku\cdot y} \right) p(y) ds_y$$

where

$$\begin{aligned} \frac{\partial}{\partial n_y} \left(S_\ell(y,\bar{y}_n)\, \mathsf{e}^{-\imath ku\cdot y} \right) &= n(y)\cdot \left((\nabla_y S_\ell(y,\bar{y}))\, \mathsf{e}^{-\imath ku\cdot y} + S_\ell(y,\bar{y}) \nabla_y \mathsf{e}^{-\imath ku\cdot y} \right) \\ &= n(y)\cdot (P_\ell(y,\bar{y}) - \imath ku\, S_\ell(y,\bar{y}))\, \mathsf{e}^{-\imath ku\cdot y}. \end{aligned}$$

Recall, the vector valued gradient $P_\ell = \nabla S_\ell$ of the interpolation operator S_ℓ in the equations above. We observe that P_k^u and, hence, also P_k remains untouched as required, operations are only performed on the interpolation operator S_ℓ.

Matrix entries As presented in sec. 2.3.3 we write the discrete operators for a cluster pair $X \subset \mathbb{R}^3$ and $Y \subset \mathbb{R}^3$ in matrix notation as the product of three matrices

$$\mathsf{V} = \mathsf{S}_X \bar{\mathsf{K}} \mathsf{S}_Y^* \qquad \text{and} \qquad \mathsf{K} = \mathsf{S}_X \bar{\mathsf{K}} \mathsf{P}_Y^*.$$

The entries of the L2L operators $\mathsf{S}_X \in \mathbb{C}^{M\times|\alpha|}$ are point-wise evaluations at collocation points $\{x_i^* : i = 1,\dots,M\} \subset X \cap \Gamma$

$$(\mathsf{S}_X)_{ij} = S_\ell(x_i^*,\bar{x}_j)\, \mathsf{e}^{\imath ku\cdot(x_i^* - \bar{x}_j)}$$

and for all interpolation points $\{\bar{x}_j : j = 1,\dots,|\alpha|\}$ in the cluster X. The M2M operators $\mathsf{S}_Y, \mathsf{P}_Y \in \mathbb{C}^{N\times|\alpha|}$ read as

$$\begin{aligned} (\mathsf{S}_Y)_{ij} &= \mathsf{e}^{\imath ku\cdot\bar{y}_j} \int_{\mathrm{supp}(\phi_i)} \phi_i(y)\, S_\ell(y,\bar{y}_j)\, \mathsf{e}^{-\imath ku\cdot y} ds_y, \\ (\mathsf{P}_Y)_{ij} &= \mathsf{e}^{\imath ku\cdot\bar{y}_j} \int_{\mathrm{supp}(\phi_i)} \phi_i(y)\, n(y)\cdot \left(P_\ell(y,\bar{y}_j) - \imath ku\, S_\ell(y,\bar{y}_j) \right) \mathsf{e}^{-\imath ku\cdot y} ds_y \end{aligned}$$

for $i = 1,\dots,M$ and $\mathrm{supp}(\phi) \subset X \cap \Gamma$ denotes the support of the basis function ϕ. It can be substituted with any basis function from (4.41). The M2L operators $\bar{\mathsf{K}} \in \mathbb{C}^{|\alpha|\times|\alpha|}$ are point-wise evaluations of the unmodified fundamental solution P_k at the interpolation points $\{\bar{x}\} \subset X$ and $\{\bar{y}\} \subset Y$. It reads as

$$(\bar{\mathsf{K}})_{ij} = P_k(\bar{x}_i, \bar{y}_j) \quad \text{for } i,j = 1,\dots,|\alpha|.$$

Apparently, if we evaluate both matrices V and K for the same cluster pair $X \times Y$, the L2L operator S_X and the M2L operator $\bar{\mathsf{K}}$ are the same, only the respective M2M operators S_Y and P_Y differ.

Implementation and computational aspects We have implemented the dFMM as an additional module in HyENA [77]. We have parallelized the M2M, M2L and L2L operation by using shared memory parallelization (see OpenMP [30]). All computations were computed on a 64 CPU node, each of 2261 MHz, with 256 GB shared memory. The effectively used number of CPU is mentioned at the respective examples. As preconditioning strategy, we simply used the inverted diagonal blocks of the near-field. This is clearly suboptimal, but still saves some iterations as we will see in the following.

5.2.2 Convergence studies

We study the convergence of the DFMM by solving interior Dirichlet and Neumann boundary value problems for the Helmholtz equation. We chose Ω to be a unit cube given by $\{x \in \mathbb{R}^3 : |x_1| = |x_2| = |x_3| = 0.5\}$ and $\Gamma = \partial\Omega$ being its surface. We use piecewise linear basis functions $\varphi \in S_h^+(\Gamma)$ to approximate the Dirichlet datum p and piecewise constant basis functions $\psi \in S_h^-(\Gamma)$ to approximate the Neumann datum $q = \partial p/\partial n$. For the solution of the resulting equation systems we use the GMRES method up to a relative accuracy $\epsilon_{\mathrm{GMRES}} = 10^{-8}$. As preconditioning strategy we simply invert the diagonal near-field blocks.

We prescribe the fundamental solution as boundary condition $g_D(x) = P_k(x,\tilde{y})$ for $x \in \Gamma$ and the fixed point $\tilde{y} \in \Omega^e$ given by $\tilde{y} = (0.55, 0.55, 0.55)^T$. Since P_k is an analytic solution of the Helmholtz equation, we can study the error convergence therewith as we refine the approximation of the boundary $\Gamma \sim \Gamma_h = \bigcup_{e=1}^{E} \tau_e$. The average size of the linear triangular boundary elements τ is denoted by h. Apparently, the relation $\mathcal{O}(E^{-1/2}) = \mathcal{O}(h)$ holds, and, due to the constant approximation of the Neumann datum, also $N = E$ holds.

Interior Dirichlet problem We solve the Helmholtz equation (4.6) together with the Dirichlet boundary condition $p(x) = g_D(x)$ for $x \in \Gamma$ and for the wave number $k = 5$. If

we use (4.42) as starting point the equation system to be solved reduces to the first block equation and reads as

$$\mathsf{V}_D\,\tilde{\mathsf{q}} = (\mathsf{C}+\mathsf{K}_D)\,\mathsf{g}_D$$

with the given Dirichlet datum $\mathsf{g}_D \in \mathbb{C}^M$ and the sought after Neumann datum $\tilde{\mathsf{q}} \in \mathbb{C}^N$. The coarsest mesh is composed by $N_0 = 1384$ linear triangles. We introduce five refinement levels up to $N_5 = 1417216$ triangles.

N	M	Acc	$Iter_{\text{diagPrec}}$	ϵ_1	roc_1	ϵ_2	roc_2
1384	694	4	34	3.13e-1	-	6.88e-5	-
5536	2770	4	43	1.69e-1	0.889	1.87e-5	1.879
22144	11074	5	45	7.26e-2	1.219	4.33e-6	2.111
88576	44290	6	56	2.81e-2	1.369	1.10e-6	1.977
354304	177154	7	69	1.05e-2	1.420	2.65e-7	2.053
1417216	708610	8	83	3.90e-3	1.428	6.86e-8	1.952

Table 5.2: Accuracy of the DFMM for the Dirichlet problem on the unit cube

Table 5.2 shows the convergence study. The first and second column list the size of the discrete Neumann and Dirichlet datum at the respective refinement level. Column three shows the chosen accuracy Acc of the DFMM. From sec. 3.3.7 we know that it can be determined by choosing $(\epsilon_{\text{ACA}}, \ell) = (10^{-Acc}, Acc)$. Column four lists the number of GMRES iterations needed to reach the prescribed accuracy ϵ_{GMRES}. Column five shows the relative L_2 error for the computed Neumann data $\tilde{\mathsf{q}}$ on Γ, computed as

$$\epsilon_1 = \frac{\|\mathsf{g}_N - \tilde{\mathsf{q}}\|_{L_2(\Gamma)}}{\|\mathsf{g}_N\|_{L_2(\Gamma)}},$$

where $\mathsf{g}_N \in \mathbb{C}^N$ is the exact Neumann datum computed as $(\mathsf{g}_N)_i = \partial g_D(x_i)/\partial n_{x_i}$. The absolute error of the computed solution $\tilde{\mathsf{p}}_\Omega$ is obtained by formulating the representation formula (4.14) twice, first for the exact Neumann datum g_N and second for the computed counterpart $\tilde{\mathsf{q}}$. After subtracting one from another we obtain

$$(\mathsf{p}_\Omega)_i - (\tilde{\mathsf{p}}_\Omega)_i = \sum_{j=1}^{N} \left((\mathsf{g}_N)_j - (\tilde{\mathsf{q}})_j\right) \int\limits_{\text{supp}(\psi_j)} P_k(\tilde{x}_i, y)\,\psi_j(y)\,ds_y \quad \text{for } y \in \Gamma.$$

with the set $\{\tilde{x}_i : i = 1,\ldots,446\}$ of inner points which are randomly distributed on the surface of a cube given by $\{x \in \mathbb{R}^3 : |x_1| = |x_2| = |x_3| \leq 0.4\}$. Column seven shows the average of the absolute or point-wise L_1 error and is computed as

$$\epsilon_2 = \frac{|\mathsf{p}_\Omega - \tilde{\mathsf{p}}_\Omega|_{L_1}}{446}.$$

Finally, the columns six and eight show the rate of convergence with respect to the mesh size h given by $\text{roc} = \frac{\log(\epsilon_{\nu-1}/\epsilon_\nu)}{\log(h_{\nu-1}/h_\nu)}$, where ν denotes the refinement level. The convergence rate roc_1 approaches 1.4, which is slightly higher than the theoretically guaranteed convergence for the Neumann datum, which would be linear [91]. The convergence rate roc_2 is quadratic as predicted theoretically for the Collocation method (for the Galerkin method it would be cubic).

Multi-frequency analysis of the interior Dirichlet problem Since the Helmholtz equation provides also another parameter, the wave number k, we study its influence. We take refinement level 3 with $N_3 = 88576$ and $M_3 = 44290$ which is sufficient for wavenumbers up to $k \leq 32$. Moreover, we keep the depth of the oct-tree constant at 4 levels and vary the wave number k as can be seen in tab. 5.3. We used 40 CPUs for the computation of this example.

k	$Iter_{\text{diagPrec}}$	$Iter_{\text{noPrec}}$	t_{mult}[s]	(lf,hf)	ϵ_1	ϵ_2
1	46	71	3.13	(3,0)	4.54e-2	3.40e-7
2	46	70	3.12	(3,0)	4.10e-2	3.39e-7
4	52	77	3.44	(2,1)	3.16e-2	5.08e-7
8	69	96	4.05	(1,2)	2.19e-2	2.44e-6
16	161	200	3.48	(0,3)	5.24e-2	1.40e-4
32	362	494	6.69	(0,2)	4.58e-2	8.79e-5

Table 5.3: Behavior of the DFMM ($Acc = 6$) at refinement level 3 for the interior Dirichlet problem for different wave numbers k

Column two and three of tab. 5.3 list the number of GMRES iterations for $\epsilon_{\text{GMRES}} = 10^{-8}$ with and without preconditioning, respectively. Column four presents the required time for a single matrix-vector multiplication and the next column lists the number of low- and high frequency levels having expansions. Obviously, the higher the wave number becomes, the more the low frequency threshold climbs up the oct-tree. This implies a growth of the near-field, what is reflected in growing timings for the matrix-vector multiplications. Noticeable is the value in column four for $k = 8$ compared to the value for $k = 16$. The reason is that the compression of the M2L operators for $k = 16$ is significantly better than for $k = 8$. The last two columns list the obtained errors ϵ_1 and ϵ_2. A possible reason for the relatively large errors for $k = 16$ might be that k^2 comes close to one of the spurious eigen frequencies where the interior Dirichlet problem is not solvable (see sec. 4.2.2 and [91]). We did not study this assumption in detail, though.

Interior Neumann problem We solve the Helmholtz equation (4.6) together with the Neumann boundary condition $g_N(x) = \partial g_D(x)/\partial n_x$ for $x \in \Gamma$ and for the wave number

$k = 5$. We take again (4.42) as starting point and end up with the equation system

$$(\mathsf{C} + \mathsf{K}_N)\,\tilde{\mathsf{p}} = \mathsf{V}_N\,\mathsf{g}_N \tag{5.4}$$

with the given Neumann datum $\mathsf{g}_N \in \mathbb{C}^N$ and the sought after Dirichlet datum $\tilde{\mathsf{p}} \in \mathbb{C}^M$. For the convergence study we choose the same refinement as for the interior Dirichlet problem.

N	M	Acc	$Iter_{\text{diagPrec}}$	ϵ_1	roc_1	ϵ_2	roc_2
1384	694	4	22	1.28e-1	-	2.93e-4	-
5536	2770	4	23	3.01e-2	2.088	6.76e-5	2.116
22144	11074	5	22	7.30e-3	2.044	1.95e-5	1.794
88576	44290	6	24	1.70e-3	2.102	4.85e-6	2.007
354304	177154	7	26	4.07e-4	2.062	1.22e-6	1.991
1417216	708610	8	27	9.43e-5	2.109	2.90e-7	2.072

Table 5.4: Accuracy of the DFMM for the Neumann problem on the unit cube

Results are plotted in tab. 5.4. Column one, two, three and four show the same as in tab. 5.2. Column five presents the relative L_2 error for the computed Dirichlet datum $\tilde{\mathsf{p}}$ on Γ, it is computed as

$$\epsilon_1 = \frac{\|\mathsf{g}_D - \tilde{\mathsf{p}}\|_{L_2(\Gamma)}}{\|\mathsf{g}_D\|_{L_2(\Gamma)}}$$

where $\mathsf{g}_D \in \mathbb{C}^M$ is the exact Dirichlet datum on Γ computed as $(\mathsf{g}_D)_i = g_D(x_i)$. As in tab. 5.2, column seven shows the average of the absolute of point-wise L_1 error of the computed solution $\tilde{\mathsf{p}}_\Omega$ in the domain Ω, but it is obtained slightly differently. Again, the representation formula (4.14) is formulated twice, however, first with the exact Dirichlet datum g_D and second for the computed counterpart $\tilde{\mathsf{p}}$. After subtracting one from another we obtain

$$(\mathsf{p}_\Omega)_i - (\tilde{\mathsf{p}}_\Omega)_i = \sum_{j=1}^{M} \left((\tilde{\mathsf{p}})_j - (\mathsf{g}_D)_j\right) \int\limits_{\text{supp}(\varphi_j)} \frac{\partial P_k(\tilde{x}_i, y)}{\partial n_y} \varphi_j(y)\, ds_y \quad \text{for } y \in \Gamma.$$

with the same set $\{\tilde{x}_i : i = 1, \ldots, 446\}$ of inner points as above. Also ϵ_2 is computed as above. We obtain a quadratic convergence rate for ϵ_1 and ϵ_2 as predicted theoretically in [91].

5.2.3 Acoustic scattering

We assume rigid scattering of sound, i.e., an acoustic wave which is incident on a rigid obstacle is totally reflected. The physical assumptions are shown in fig. 5.14. As usual

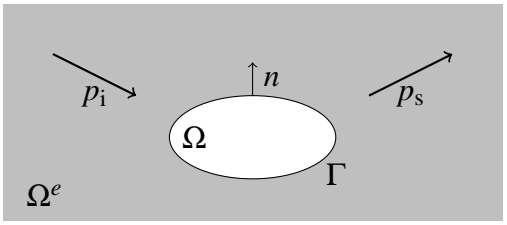


Figure 5.14: Rigid scattering of sound

$\Omega \subset \mathbb{R}^3$ is a bounded domain, called *obstacle*, with the surface $\Gamma = \partial\Omega = \partial\Omega^e$, where Ω^e is the unbounded exterior domain. The unit normal vector on Γ is denoted by n. The acoustic fluid in Ω^e is supposed to be ideal, and is given by its bulk modulus $B\,[\mathrm{N/m^2}]$ and density $\rho\,[\mathrm{kg/m^3}]$. All waves are time harmonic with a circular frequency ω.

As incoming acoustic wave field p_i a plane wave is considered

$$p_\mathrm{i}(x) = \mathrm{e}^{\imath k u \cdot x}$$

with a unit vector u, such that $|u| = 1$ holds. It describes a plane wave with a wave number k moving in direction of u. It is useful to divide the the total acoustic field into the sum of the known incident acoustic field p_i and the unknown scattered acoustic field p_s as

$$p(x) = p_\mathrm{i}(x) + p_\mathrm{s}(x).$$

Rigid acoustic scattering requires the acoustic flux to vanish on the boundary Γ such that

$$\frac{\partial p(x)}{\partial n} = 0 \quad \text{for } x \in \Gamma$$

holds. Finally, the exterior Neumann problem to be solved reads as

$$\begin{aligned} \Delta p_\mathrm{s}(x) + k^2 p_\mathrm{s}(x) &= 0 && \text{for } x \in \Omega^e, \\ \frac{\partial p_\mathrm{s}(x)}{\partial n} &= -\frac{\partial p_\mathrm{i}(x)}{\partial n} && \text{for } x \in \Gamma. \end{aligned} \tag{5.5}$$

The Sommerfeld condition (4.8) is automatically satisfied by the boundary integral equation we use for solving the above stated acoustic scattering problem. It ensures that waves are not reflected at infinity and makes the problem uniquely solvable.

25 spheres Given are 25 unit spheres of diameter 1 m (see, e.g., fig. 5.15). They represent rigid scatterers and form the domain Ω. The exterior domain Ω^e is assumed to be an ideal fluid. For simplicity we choose the bulk modulus $B = 1.0\,\mathrm{N/m^2}$ and the density $\rho =$

$1.0\,\mathrm{kg/m^3}$. The incoming plane wave is chosen to be $p_\mathrm{i}(x) = \mathrm{e}^{\imath k u \cdot x}$. Hence, the imposed Neumann boundary condition in (5.5) reads as

$$g_N(x) = \frac{\partial p_\mathrm{s}(x)}{\partial n_x} = -\imath k \mathrm{e}^{\imath k u \cdot x}(n_x \cdot u) \quad \text{for } x \in \Gamma. \tag{5.6}$$

The equation system to be solved reads as (5.4). The 25 scatterers are discretized in totally $E = 210550$ linear triangles. We choose piecewise linear basis functions for the Dirichlet datum and end up with the unknown vector $\mathsf{p}_\mathrm{s} \in \mathbb{C}^M$ of size $M = 105325$. Moreover, we choose piecewise constant basis functions for the Neumann datum and get the vector $\mathsf{g}_N \in \mathbb{C}^N$ of size $N = E = 210550$ prescribed by (5.6). Once, the p_s is known on the surface of the scatterers we can evaluate the total pressure field $\mathsf{p} = \mathsf{p}_\mathrm{s} + \mathsf{p}_\mathrm{i}$ at 12896 randomly scattered points $\{\tilde{x}\} \subset \Omega^e$ on a plane which is located 0.2 m behind the layer of spheres. This is done by evaluating the representation formula at all points exterior points $\{\tilde{x}\}$ and finally adding the incoming pressure field p_i, the solution reads as

$$p(\tilde{x}) = -(\mathcal{V}_k p_\mathrm{s})(\tilde{x}) + (\mathcal{K}_k g_N)(\tilde{x}) + p_\mathrm{i}(\tilde{x}).$$

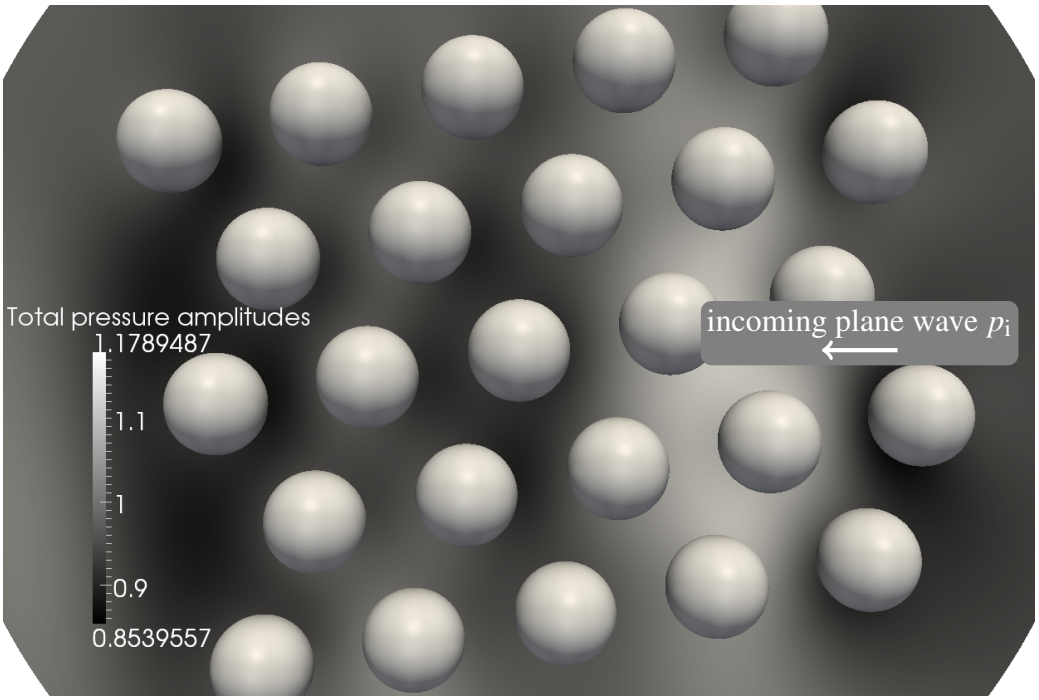


Figure 5.15: $k = 1$

The figs. 5.15, 5.16, 5.17 and 5.18 show the resulting total pressure field $|p(\tilde{x})|$ on the disc in Ω^e for $k = 1, 2, 4, 8$, respectively. All four computation were conducted on 30 CPUs. Table 5.5 lists the number of GMRES iterations for $\epsilon_{\mathrm{GMRES}} = 10^{-5}$, the overall time for

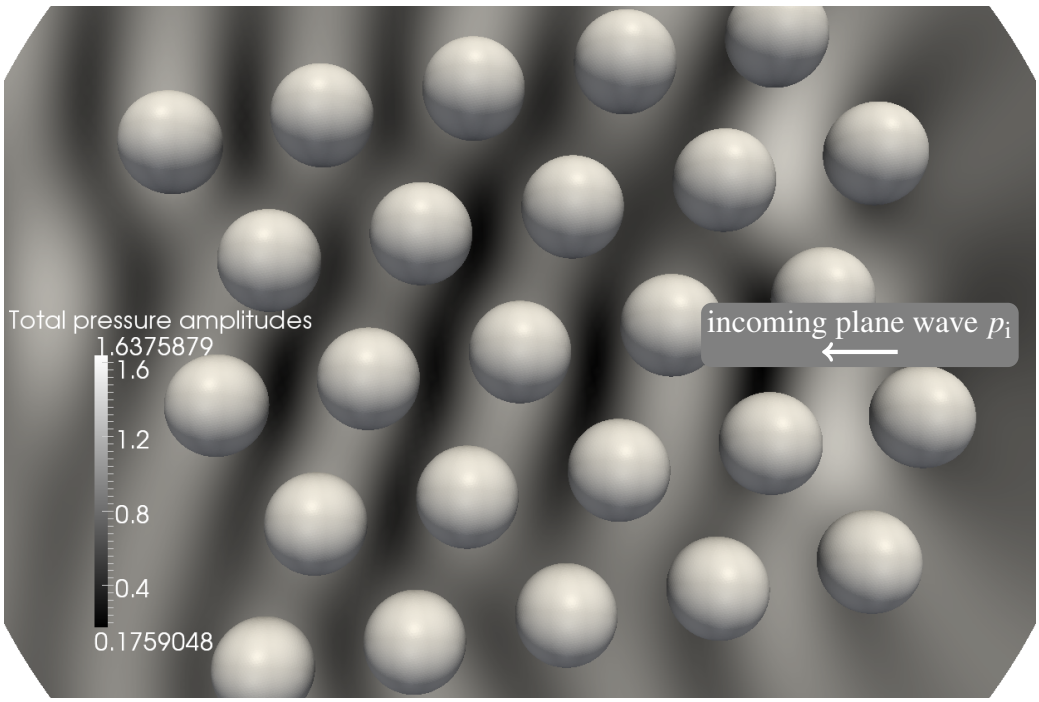


Figure 5.16: $k = 2$

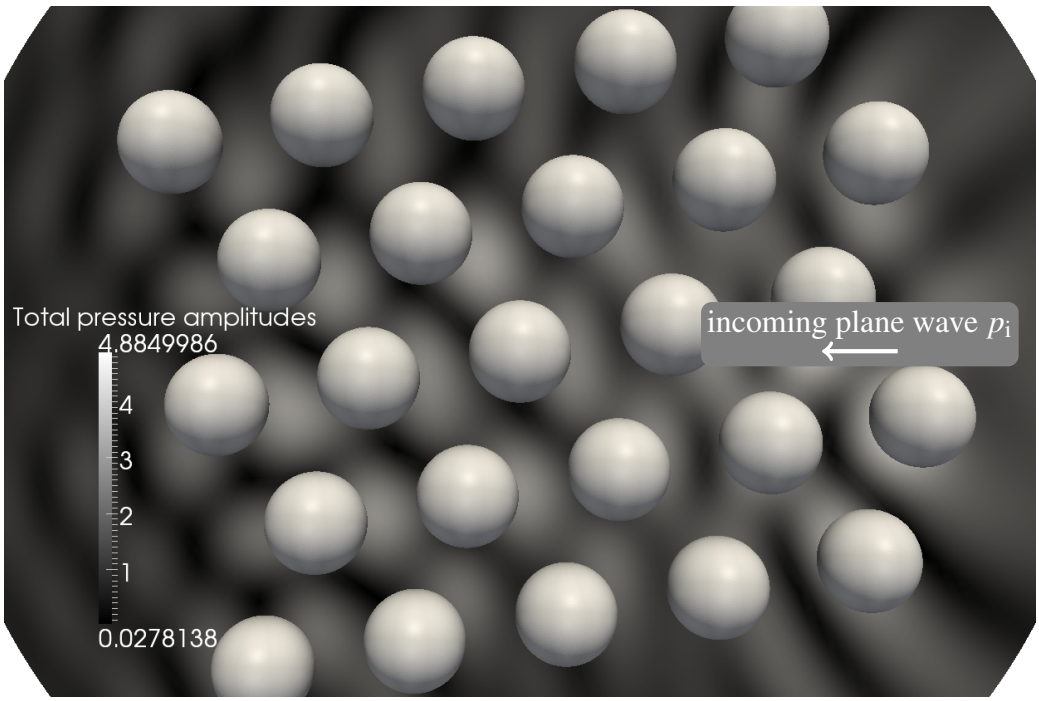


Figure 5.17: $k = 4$

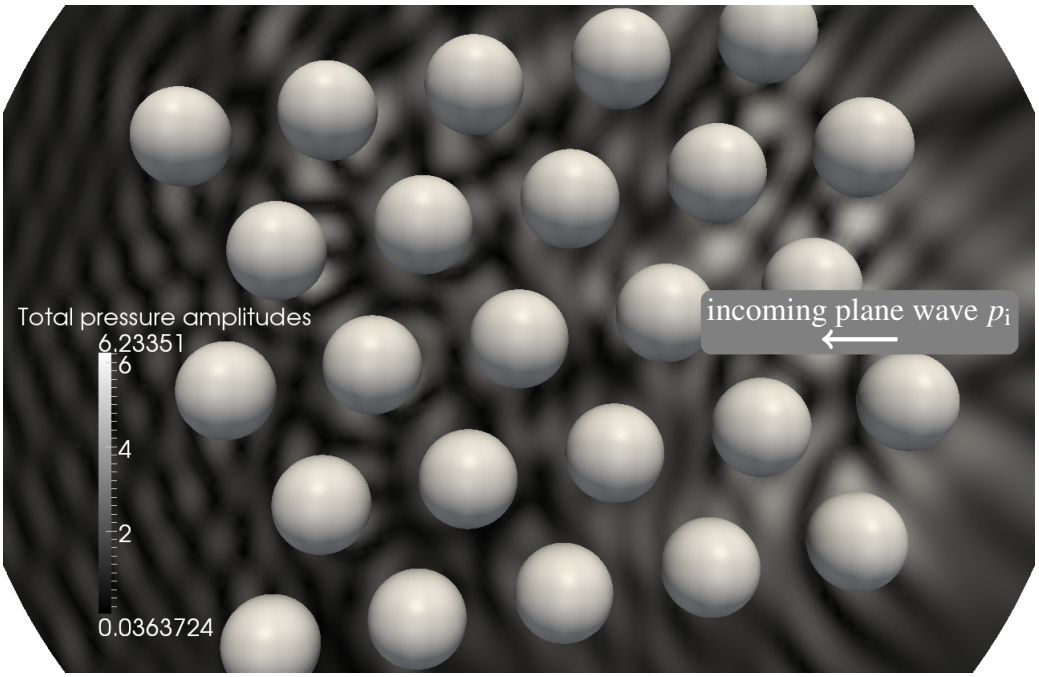


Figure 5.18: $k = 8$

the solution of the linear system and the time for a single matrix-vector multiplication. The accuracy of the dFMM was chosen to be $Acc = 5$. The depth of the oct-tree was kept constant at 5 levels for all wave numbers k. Evidently, the required time for the GMRES solution grows almost linearly with the number of GMRES iterations. This shows clearly the need of an efficient preconditioning strategy when using the dFMM. Column four and

k	*Iter*	t_{sol}[s]	t_{mult}[s]	(lf, hf)
1	7	24.23	3.46	(2, 2)
2	13	48.47	3.73	(1, 2)
4	23	83.18	3.61	(0, 3)
8	39	202.07	5.18	(0, 2)

Table 5.5: Number of GMRES iterations, timings and low- and high frequency levels having expansions

five in tab. 5.5 report on the time for a single matrix-vector multiplication and on the number of low- and high frequency levels having a non-empty far-field. For example, for $k = 1$ we have $(\text{lf}, \text{hf}) = (2, 2)$, meaning that the leaf level and the next higher level are in the low-frequency regime and have interactions. Moreover the next two levels are in the high-frequency regime and have directional interactions. This changes for $k = 8$ where we have $(\text{lf}, \text{hf}) = (0, 2)$. No low-frequency levels exist. The leaf level and the

next higher level are in the high-frequency regime and have directional interactions. All other levels have an empty far-field. As expected, the higher the wave number k is, the larger the near-field becomes. For $k = 8$ only two high frequency levels have expansions, hence, the near-field is much larger compared to $k = 1, 2, 4$. That is why its matrix-vector multiplication takes the most time.

5.2.4 Sound radiation

A multi-physical model is necessary to compute electromagnetically excited sound radiation of electrical machines [94]. An electromagnetic model delivers mechanical forces acting on the machine. Such forces lead to structural vibrations which can be computed by means of an elastodynamic model. Our objective addresses the sound radiation induced by surface vibrations.

Starting point is the solid-fluid interaction [61]. Machine vibrations given as velocities v on its surface Γ excite the exterior acoustic fluid in Ω^e. It picks up the vibrations of the embedded machine in the form of acoustic waves, i.e., sound is radiated from the machine. The acoustic problem to be solved is an *exterior Neumann problem*. The boundary condition is obtained from the compatibility condition requiring the normal velocities of the solid and fluid to be equal on Γ. We multiply the stationary Euler equation (obtained from (4.3)) with the normal vector n and interpret the resulting normal velocities $v \cdot n$ as the velocities of the solid. This leads to the Neumann boundary condition

$$g_N(x) = \frac{\partial p(x)}{\partial n_x} = -\imath\omega\rho_f \, v(x) \cdot n_x \quad \text{for } x \in \Gamma, \tag{5.7}$$

with given velocities v. The final goal is to determine the stationary acoustic field of radiated acoustic pressure p in Ω^e.

We deploy the DFMM for solving the exterior Neumann problem. We assume the acoustic fluid to be ideal. We choose it to be air with the bulk modulus $B = 134081.92\,\mathrm{N/m^2}$ and density $\rho = 1.12\,\mathrm{kg/m^3}$. The electric machine has the dimensions $(0.902\,\mathrm{m}, 0.902\,\mathrm{m}, 1.730\,\mathrm{m})$ and is shown in fig. 5.19. The mesh consists of $E = 103554$ linear triangles. The equation system to be solved is (5.4). We choose piecewise linear continuous basis functions to approximate the Dirichlet datum and end up with the vector $\mathsf{p} \in \mathbb{C}^M$ of size $M = 51763$. Moreover, we choose piecewise constant basis functions to approximate the Neumann datum and get a vector $\mathsf{g}_N \in \mathbb{C}^N$ of size $N = E = 103554$. The given velocities v on the surface of the electric machine are assumed to be equal for various excitation frequencies. Hence, considering (5.7), only the amplitudes of the given Neumann datum changes, as can be seen in fig. 5.20. Once, the Dirichlet datum is computed on the surface Γ of the machine we approach our final goal and determine the stationary acoustic field in the exterior

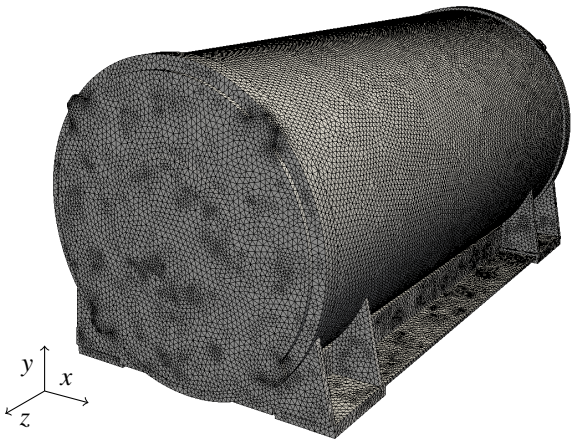
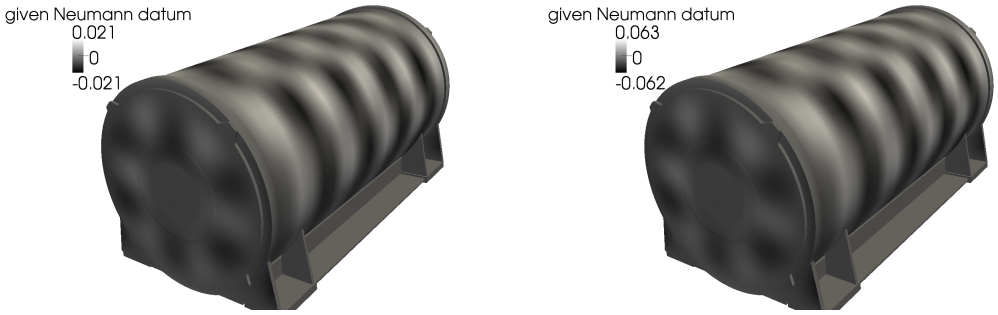


Figure 5.19: Electric machine (103554 linear triangles)



(a) 750 Hz (b) 2200 Hz

Figure 5.20: Real part of given Neumann datum $\mathrm{Re}\, g_N$ in $[\mathrm{N}/\mathrm{m}^2]$

domain Ω^e. The solution is complex-valued and reads as

$$p(\tilde{x}) = -(\mathcal{V}_k p)(\tilde{x}) + (\mathcal{K}_k g_N)(\tilde{x}) \quad \text{for } \tilde{x} \in \Omega^e.$$

All computation were performed on 40 CPUs. The accuracy of the dFMM was set to $Acc = 5$ and the solution accuracy $\epsilon_{\text{GMRES}} = 10^{-5}$. The oct-tree has 5 levels in all computations which are studied in tab. 5.6. Column two in tab. 5.6 shows the ratio of wave length

f [Hz]	λ/h	*Iter*	t_{sol}[s]	t_{mult}[s]	(lf,hf)
750	34.1	112	475.39	4.24	(1,2)
1000	25.6	171	777.69	4.55	(0,3)
1500	17.1	261	1275.99	4.88	(0,3)
2200	11.6	566	4347.20	7.68	(0,2)

Table 5.6: Number of GMRES iterations for $\epsilon_{\text{GMRES}} = 10^{-5}$, timings and low- and high frequency levels having expansions

$\lambda = c/f$ to the average mesh size h. Recall, the circular frequency ω is related to the frequency f as $\omega = 2\pi f$. Yet, for a frequency of 2200 Hz the wave has a length of about 11.6 elements. This is sufficient for its approximation. As in tab. 5.5, the overall solution time grows with the number of GMRES iterations. The jump in the time for a single matrix-vector multiplication at 2200 Hz with respect the other frequencies is caused by the fact that only two levels have a non-empty far-field. This can be seen in the last column.

The figs. 5.21, 5.22, 5.23 and 5.24 show $\operatorname{Re} p$ on 65901 randomly scattered points on a plane given by $\{x \in \Omega^e : x_2 = 0\}$ for the excitation frequencies 750 Hz, 1000 Hz, 1500 Hz and 2200 Hz, respectively. The decrease of the amplitudes is distinctively visible.

The figs. 5.25, 5.26, 5.27 and 5.28 show the absolute value of the radiated sound pressure $|p|$ on the same plane in Ω^e for the same set of frequencies.

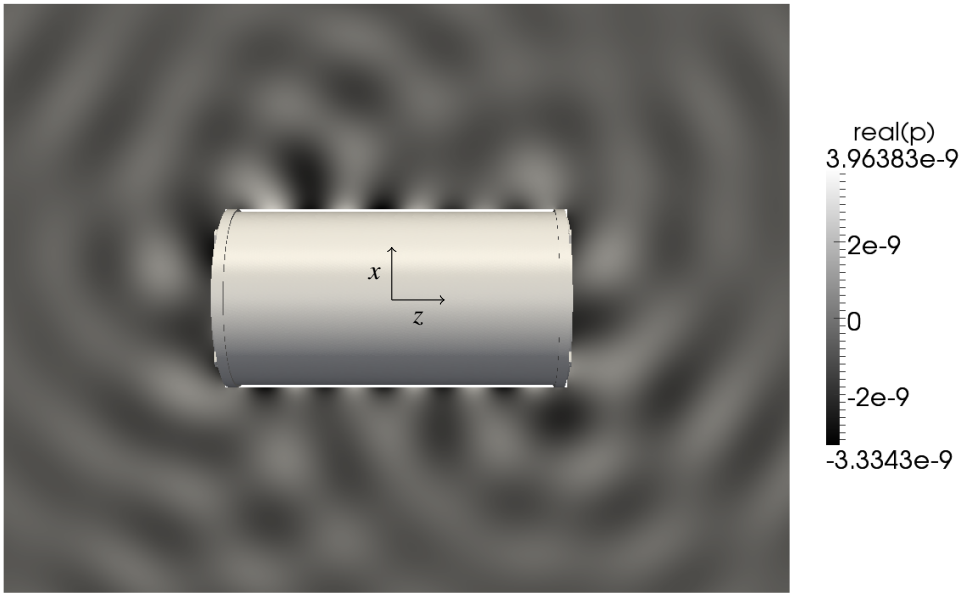


Figure 5.21: Sound pressure Re p in $[\mathrm{N/m^2}]$ on a plane in Ω^e at 750 Hz

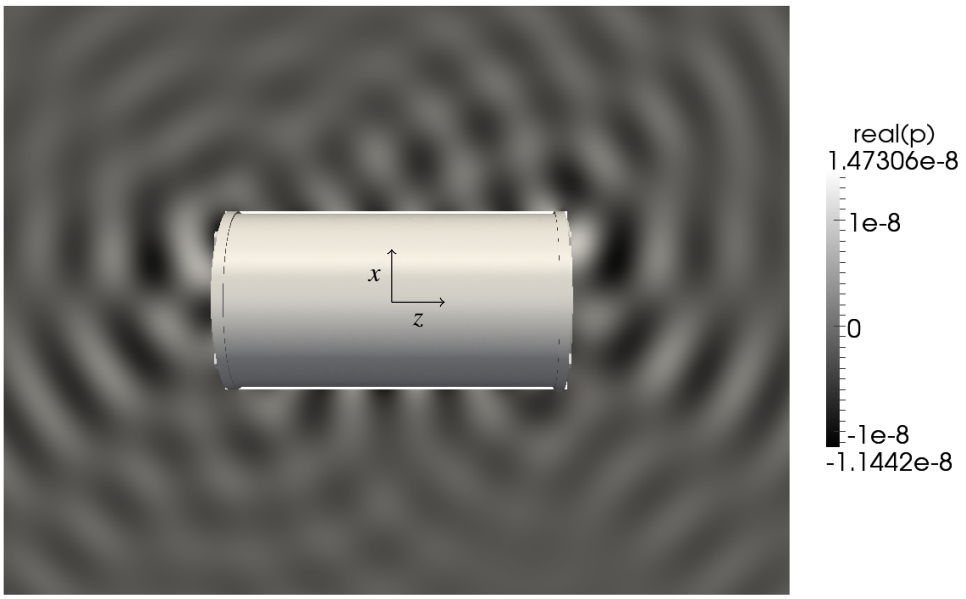


Figure 5.22: Sound pressure Re p in $[\mathrm{N/m^2}]$ on a plane in Ω^e at 1000 Hz

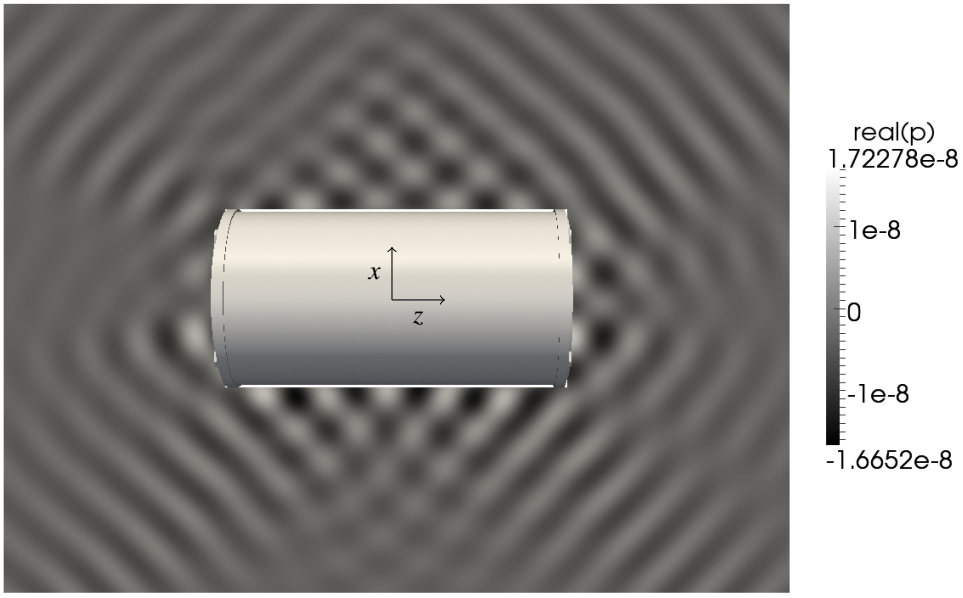


Figure 5.23: Sound pressure $\mathrm{Re}\, p$ in $[\mathrm{N/m^2}]$ on a plane in Ω^e at 1500 Hz

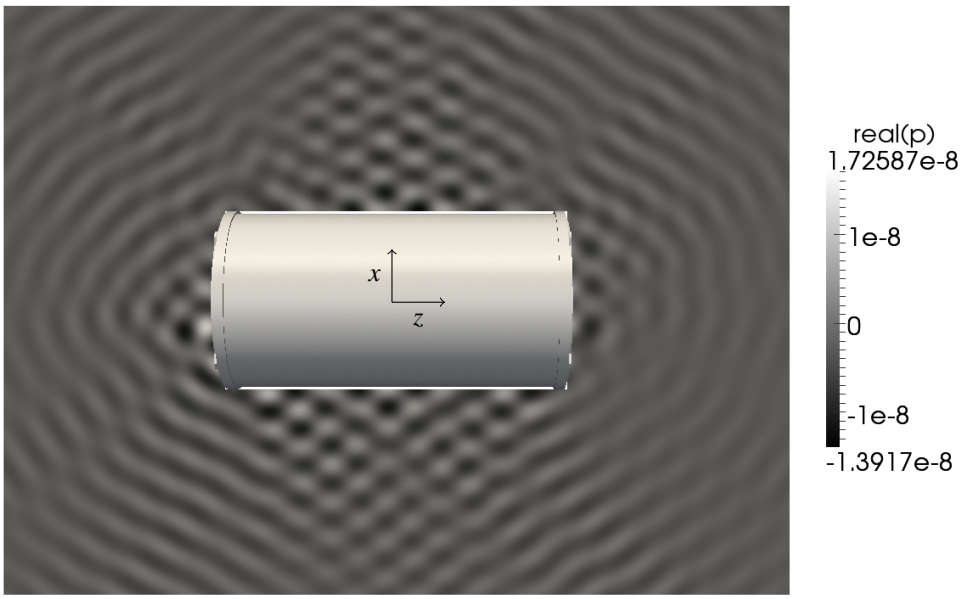


Figure 5.24: Sound pressure $\mathrm{Re}\, p$ in $[\mathrm{N/m^2}]$ on a plane in Ω^e at 2200 Hz

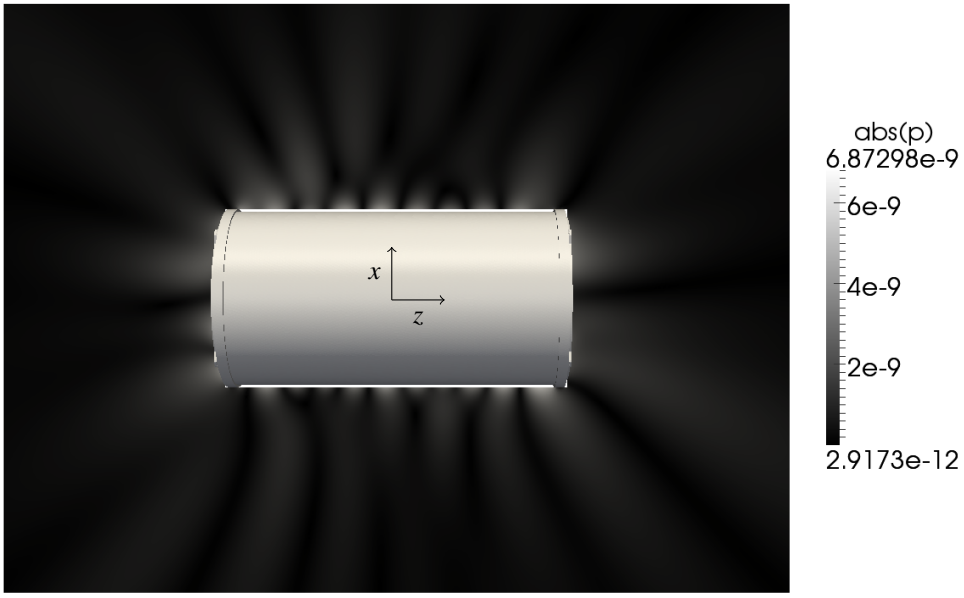


Figure 5.25: Sound pressure $|p|$ in $[\mathrm{N/m^2}]$ on a plane in Ω^e at 750 Hz

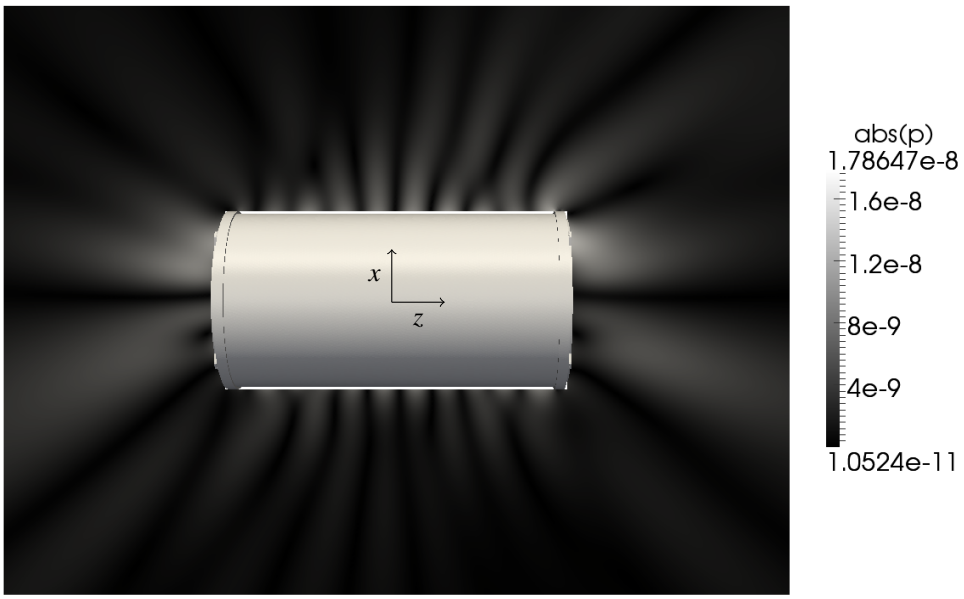


Figure 5.26: Sound pressure $|p|$ in $[\mathrm{N/m^2}]$ on a plane in Ω^e at 1000 Hz

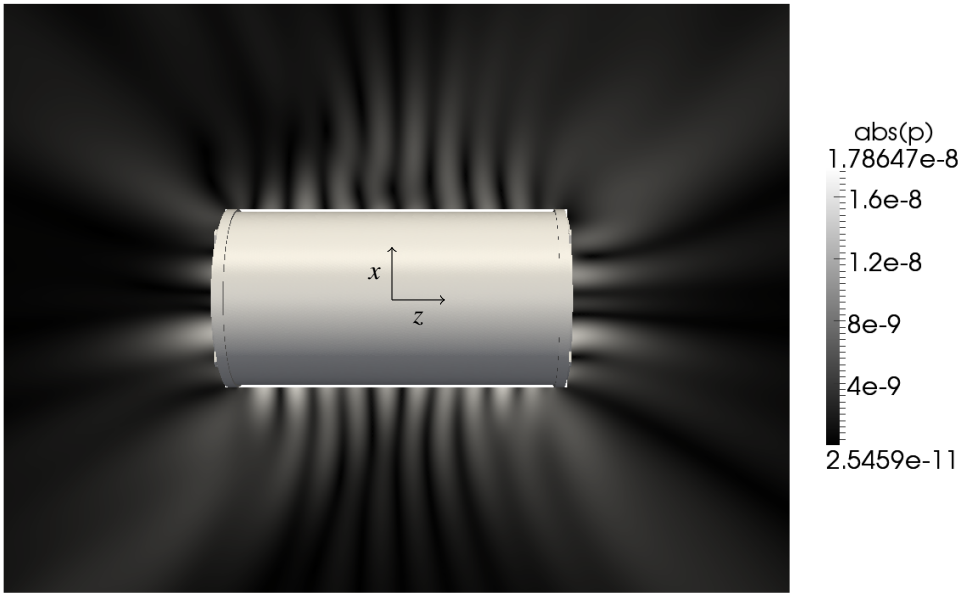


Figure 5.27: Sound pressure $|p|$ in $[\mathrm{N/m^2}]$ on a plane in Ω^e at 1500 Hz

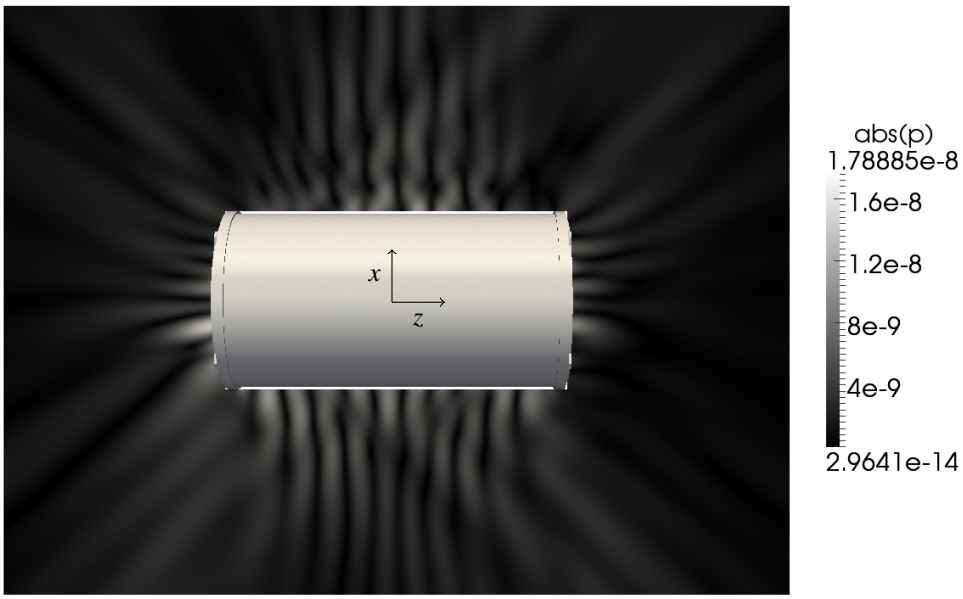


Figure 5.28: Sound pressure $|p|$ in $[\mathrm{N/m^2}]$ on a plane in Ω^e at 2200 Hz

6 CONCLUSION

The thesis consist mainly of two parts. The first part considers efficient and accurate numerical tools for evaluating many pair-wise interactions of oscillatory nature. In the second part, we used them to increase the efficiency of boundary element methods and its capability of simulating large-scale problems in acoustics.

We have presented three approximation strategies for dense matrices which are of oscillatory kernels: The singular value decomposition (SVD), the adaptive cross approximation (ACA) and the Chebyshev interpolation scheme. Even though, the SVD is capable of computing the best-possible approximation, it is prohibitive in terms of its computational cost. The ACA and the interpolation based approach do not have this bottleneck, they are also efficient in terms of computational cost. The ACA is very little code intrusive. This outstanding feature results from its black-box and self-controlling nature. The ACA yields the near-best-possible approximation. Moreover, by combining it with a subsequent application of the SVD also the best-possible approximation can be obtained. These facts explain the high popularity of the ACA in both science and engineering. The second approach is slightly more involving. It is based on the interpolation of oscillatory kernels via Chebyshev polynomials. An advantage is the relatively simple error analysis based on the growth of the kernel function in the complex plane. The effective rank of the approximation is determined by the a priori chosen interpolation order. Thus, it might lead to suboptimal approximations. To remedy this drawback, we have proposed an approach which combines the Chebyshev interpolation with a subsequent application of the ACA and the SVD. In this way the merits of all three schemes are exploited and end up with the best-possible approximation.

Moreover, we have presented two efficient schemes for evaluating summations which have oscillatory kernel functions: The $\mathcal{H}$-matrix approach and the directional fast multipole method (dFMM). Both schemes base on the partitioning of the underlying domain in so called cluster-trees. We have studied balanced and uniform cluster-trees. The former tree type allows to better exploit the advantages of the ACA. The latter one enables a further recompression when using an interpolation based approximation. Clearly, the $\mathcal{H}$-matrix approach, which uses the ACA, is less involving. The reason is that $\mathcal{H}$-matrices, apart from the fact that they have different structure, can be understood as matrices in the usual sense. Meaning that the entire framework of matrix arithmetics is provided. Moreover, they allow for an easy and efficient construction of good preconditioners by computing approximative matrix decompositions. Conversely, the dFMM, which is based on the Chebyshev interpolation scheme, is clearly less straight forward and also more involving to implement.

Nonetheless, both methods rely on the original and unmodified kernel. They are also applicable to non-oscillatory kernels without further treatment. Moreover, the black-box nature of both schemes facilitates a potential extension to vector-valued kernels.

In the second part of the thesis, we have presented acoustic boundary element methods (BEM). We have derived the governing equations, i.e., the wave and the Helmholtz equation, and their respective integral representations. We have adopted the collocation scheme to conduct the spatial discretization. Moreover, we have utilized the convolution quadrature method (CQM) to convert time-domain problems to systems of decoupled Helmholtz problems. Only then, the application of the efficient schemes we presented in the first part of the thesis can be used straight forward.

We have validated the presented methodologies with numerical examples. We have adopted the $\mathcal{H}$-matrix approach to speedup the simulation of a mixed interior acoustic initial-boundary value problem. The problem to be solved is of rather academic nature, however, it points out the necessity of having a stable and accurate numerical scheme. In the context of the CQM we have compared four different time-stepping procedures for its realization: the BDF2 and three Runge-Kutta methods. The numerical examples approve that all four procedures lead to results which match well with the analytical solutions. However, we were interested in the optimal setting in terms of overall cost and quality of the result. Runge-Kutta methods turned out to lead to better results than the BDF2.

Moreover, we have used the dFMM to accelerate BE-formulations to solve steady-state Helmholtz problems. Noteworthy, is the reusability of appearing operator matrices. For example the L2L (interpolation), respectively, the M2L operators are identical when evaluating the discrete single- and double layer operator. Only the M2M (anterpolation) operator differs. We have verified the error convergence of the proposed method for both, interior Dirichlet- and Neumann problems and have reached the theoretically estimated convergence rates. Finally, we have presented the numerical solution of two exterior problems. In the first one, we have considered a rigid acoustic scatterer. The acoustic pressure in the exterior domain has been of interest. In the second example, we have considered the sound radiation from an electrical machine for different excitation frequencies. The accuracy of these two examples has not been validated, however, the resulting images seem to be correct.

The combination of the dFMM with a Galerkin BEM and the incorporation of the second boundary integral equation [91] would be straightforward since all arising trace operators can be shifted to the interpolation operators. However, the critical point is to gain control on the arising singularities in the near-field.

A further use of the dFMM is the adaptation to vectorial problems. The complication which arises here is the following. Consider the elastodynamic kernel $U_{k_i}(x,y)$ in $\mathbb{R}^3$. It features two elastic waves of different velocities c_1 and c_2 (given by $k_{1,2} = \omega/c_{1,2}$) and each wave is connected to an oscillatory term like $\mathrm{e}^{\imath k_i|x-y|}$. Hence, in the high frequency

regime, we are required to split up such kernels, apply the directional summation to each part (with wave numbers k_1 and k_2) separately, and finally combine both contributions.

REFERENCES

[1] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin. Wavelet-like bases for the fast solutions of second-kind integral equations. *SIAM Journal on Scientific Computing*, 14(1):159–184, 1993.

[2] H. Antes. *Anwendungen der Methode der Randelemente in der Elastodynamik und der Fluiddynamik*. Mathematische Methoden in der Technik. B. G. Teubner, 1988.

[3] A. W. Appel. An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, 6:85–103, 1985.

[4] L. Banjai. Multistep and multistage convolution quadrature for the wave equation: Algorithms and experiments. *SIAM Journal on Scientific Computing*, 32(5):2964–2994, 2010.

[5] L. Banjai and W. Hackbusch. Hierarchical matrix techniques for low- and high-frequency Helmholtz problems. *IMA Journal of Numerical Analysis*, 28(1):46–79, 2008.

[6] L. Banjai and S. Sauter. Rapid solution of the wave equation in unbounded domains. *SIAM Journal on Numerical Analysis*, 47(1):227–249, 2008.

[7] L. Banjai, M. Messner, and M. Schanz. Runge-Kutta convolution quadrature for the boundary element method. *Computer Methods in Applied Mechanics and Engineering*, 2011. Submitted.

[8] J. Barnes and P. Hut. A hierarchical $\mathcal{O}(N \log N)$ force-calculation algorithm. *Nature*, 324(6096):446–449, 1986.

[9] M. Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589, 2000.

[10] M. Bebendorf. Hierarchical LU decomposition based preconditioners for BEM. *Computing*, 74:225–247, 2005.

[11] M. Bebendorf. Another software library on hierarchical matrices for elliptic differential equations (AHMED), 2008. URL `http://bebendorf.ins.uni-bonn.de/AHMED.html`. [Online; accessed 17-January-2010].

[12] M. Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, volume 63 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, 2008.

[13] M. Bebendorf and R. Grzibovski. Accelerating Galerkin BEM for linear elasticity using adaptive cross approximation. *Mathematical Methods in the Applied Sciences*, 29:1721–1747, 2006.

[14] M. Bebendorf and R. Kriemann. Fast parallel solution of boundary integral equations and related problems. *Computing and Visualization in Science*, 8(3):121–135, 2005.

[15] M. Bebendorf and S. Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70:1–24, 2003.

[16] I. Benedetti, M. H. Aliabadi, and G. Davi. A fast 3D dual boundary element method based on hierarchical matrices. *International Journal for Solids and Structures*, 45 (7-8):2355–2376, 2008.

[17] D. E. Beskos. Boundary element methods in dynamic analysis. *Applied Mechanics Review*, 40(1):1–23, 1987.

[18] D. E. Beskos. Boundary element methods in dynamic analysis: Part II (1986-1996). *Applied Mechanics Review*, 50(3):149–197, 1997.

[19] S. Börm. *Efficient Numerical Methods for Non-local Operators: $\mathcal{H}^2$-Matrix Compression, Algorithms and Analysis*. EMS tracts in mathematics. European Mathematical Society, 2010.

[20] S. Börm and L. Grasedyck. Low-rank approximation of integral operators by interpolation. *Computing*, 72:325–332, 2004.

[21] S. Börm, L. Grasedyck, and W. Hackbusch. Introduction to hierarchical matrices with applications. *Engineering Analysis with Boundary Elements*, 27(5):405–422, 2003.

[22] S. Börm, M. Löhndorf, and J. M. Melenk. Approximation of integral operators by variable-order interpolation. *Numerische Mathematik*, 2005.

[23] A. Brandt. Multilevel computations of integral transforms and particle interactions with oscillatory kernels. *Computer Physics Communications*, 65(1-3):24–38, 1991.

[24] J. Breuer, O. Steinbach, and W. L. Wendland. A wavelet boundary element method for the symmetric boundary integral formulation. In *IABEM 2002*, UT Austin, TX, USA, May 28-30 2002. International Association for Boundary Element Methods.

[25] L. Brutman. On the Lebesgue function for polynomial interpolation. *SIAM Journal on Numerical Analysis*, 15(4):694–704, 1978.

[26] A. Buchau, W. M. Rucker, O. Rain, V. Rischmüller, S. Kurz, and S. Rjasanow. Comparison between different approaches for fast and efficient 3-D BEM computations. *IEEE Transactions on Magnetics*, 39(3), 2003.

[27] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal on Scientific and Statistical Computing*, 9(4): 669, 1988.

[28] C. Cecka and E. Darve. Fourier based fast multipole method for the Helmholtz equation. *SIAM Journal on Numerical Analysis*, 2010. Submitted.

[29] S. Chaillat, M. Bonnet, and J. F. Semblat. A multi-level fast multipole BEM for 3-D elastodynamics in the frequency domain. *Computer Methods in Applied Mechanics and Engineering*, 197:4233–4249, 2008.

[30] B. Chapman, G. Jost, and R. v. d. Pas. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press, 2007.

[31] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155:468–498, 1999.

[32] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. Greengard, J. F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao. A wideband fast multipole method for the Helmholtz equation in three dimensions. *Journal of Computational Physics*, 216 (1):300–325, 2006.

[33] B. A. Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM News*, 33(4):20–21, 2000.

[34] M. Costabel. *Time-Dependent Problems with the Boundary Integral Equation Method*, volume 1 of *Encyclopedia of Computational Mechanics*, chapter 25. John Wiley & Sons, New York, Chister, Weinheim, 2005.

[35] E. Darve. The fast multipole method I: Error analysis and asymptotic complexity. *SIAM Journal on Numerical Analysis*, 38(1):98–128, 2000.

[36] E. Darve. The fast multipole method: Numerical implementation. *Journal of Computational Physics*, 160(1):195–240, 2000.

[37] E. Darve and P. Havé. Efficient fast multipole method for low-frequency scattering. *Journal of Computational Physics*, 197(1):341–363, 2004.

[38] J. Domínguez. *Boundary Elements in Dynamics*. Computational Mechanics Publications, Southampton, Boston, 1993.

[39] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936.

[40] B. Engquist and L. Ying. Fast directional multilevel algorithms for oscillatory kernels. *SIAM Journal on Scientific Computing*, 29(4):1710–1737, 2007.

[41] B. Engquist and L. Ying. Fast directional algorithms for the Helmholtz kernel. *Journal of Computational and Applied Mathematics*, 234(6):1851–1859, 2010.

[42] W. Fong and E. Darve. The black-box fast multipole method. *Journal of Computational Physics*, 228(23):8712–8725, 2009.

[43] L. Gaul, M. Kögl, and M. Wagner. *Boundary Element Methods for Engineers and Scientists*. Springer-Verlag Berlin Heidelberg, 2003.

[44] Ch. Geuzaine and J. F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[45] K. Giebermann. Multilevel approximation of boundary integral operators. *Computing*, 67(3):183–207, 2001.

[46] D. Givoli. *Numerical Methods for Problems in Infinite Domains*. Elsevier, 1992.

[47] G. H. Golub and C. F. van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[48] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and its Applications*, 261(1-3):1–21, 1997.

[49] K. F. Graff. *Wave Motions in Elastic Solids*. Dover: New York, 1991.

[50] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[51] L. Greengard and V. Rokhlin. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numerica*, 6:229–269, 1997.

[52] L. Greengard, J. Huang, V. Rokhlin, and S. Wandzura. Accelerating fast multipole methods for the Helmholtz equation at low frequencies. *IEEE Computational Science & Engineering*, 5(3):32–38, 1998.

[53] N. A. Gumerov and R. Duraiswami. *Fast Multipole Methods for the Helmholtz Equation in Three Dimensions*. Elsevier Series in Electromagnetism. Elsevier, 2004.

[54] W. Hackbusch. *Integralgleichungen – Theorie und Numerik*, volume 68 of *Leitfäden der angewandten Mathematik und Mechanik LAMM*. B. G. Teubner, 1989.

[55] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62:89–108, 1999.

[56] W. Hackbusch. *Hierarchische Matrizen: Algorithmen und Analysis*. Springer-Verlag, 2009.

[57] W. Hackbusch and S. Börm. Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing*, 35(1):1–35, 2002.

[58] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. part II : Application to multi-dimensional problems. *Computing*, 64(1):21–47, 2000.

[59] W. Hackbusch and Z. P. Nowak. On the fast matrix multiplication in the boundary element method by panel clustering. *Numerische Mathematik*, 54(4):463–491, 1989.

[60] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential - Algebraic problems*, volume 14. Springer-Verlag, 1996.

[61] F. Ihlenburg. *Finite Element Analysis of Acoustic Scattering*. Applied Mathematical Sciences. Springer, 1998.

[62] L. Kielhorn and M. Schanz. CQM based symmetric Galerkin BEM: Regularization of strong and hypersingular kernels in 3-d elastodynamics. *International Journal for Numerical Methods in Engineering*, 2008.

[63] L. E. Kinsler. *Fundamentals of Acoustics*. Wiley, 2000.

[64] W. Kress and S. Sauter. Numerical treatment of retarded boundary integral equations by sparse panel clustering. *IMA Journal of Numerical Analysis*, 28(1):162–185, 2008.

[65] Y. Liu. *Fast Multipole Boundary Element Method: Theory and Applications in Engineering*. Cambridge University Press, 2009.

[66] Y. J. Liu and N. Nishimura. The fast multipole boundary element method for potential problems: A tutorial. *Engineering Analysis with Boundary Elements*, 30(5):371–381, 2006.

[67] Y. L. Liu, N. Nishimura, Y. Otani, T. Takahashi, X .L. Chen, and H. Munakata. A fast boundary element method for the analysis of fiber-reinforced composites based on a rigid-inclusion model. *Journal of Applied Mechanics*, 72:115–128, 2005.

[68] C. V. Loan. *Computational Frameworks for the Fast Fourier Transform*. Society for Industrial and Applied Mathematics, 1992.

[69] C. Lubich. Convolution quadrature and discretized operational calculus I. *Numerische Mathematik*, 52:129–145, 1988.

[70] C. Lubich. Convolution quadrature and discretized operational calculus II. *Numerische Mathematik*, 52:413–425, 1988.

[71] C. Lubich. On the multistep time discretization of linear initial-boundary value problems and their boundary integral equations. *Numerische Mathematik*, 67:365–389, 1994.

[72] W. J. Mansur. *A Time-Stepping Technique to Solve Wave Propagation Problems Using the Boundary Element Method*. PhD thesis, University of Southampton, 1983.

[73] M. Margonari and M. Bonnet. Fast multipole method applied to elastostatic BEM-FEM coupling. *Computers and Structures*, 83:700–717, 2005.

[74] J. C. Mason and D. C. Handscomb. *Chebyshev Polynomials*. Chapman & Hall/CRC, 2003.

[75] M. Messner and M. Schanz. An accelerated symmetric time-domain boundary element formulation for elasticity. *Engineering Analysis with Boundary Elements*, 34 (11):944–955, 2010.

[76] M. Messner, M. Schanz, and E. Darve. Fast directional multilevel summation for oscillatory kernels based on Chebyshev interpolation. *Journal of Computational Physics*, 2011. Published.

[77] Ma. Messner, Mi. Messner, F. Rammerstorfer, and P. Urthaler. Hyperbolic and elliptic numerical analysis BEM library (HyENA), 2010. URL `http://www.mech.tugraz.at/HyENA`. [Online; accessed 22-January-2010].

[78] E. Michielssen and J. M. Song. Fast solution methods in electromagnetics. *IEEE Transactions on Antennas and Propagagation*, 45(3):533–543, 1997.

[79] P. M. C. Morse and H. Feshbach. *Methods of Theoretical Physics*. Number 1 in International Series in Pure and Applied Physics. McGraw Hill, 1953.

[80] J.C. Nédélec. *Acoustic and Electromagnetic Equations: Integral Representations for Harmonic Problems*. Applied mathematical sciences. Springer, 2001.

[81] G. Of, O. Steinbach, and W. L. Wendland. A fast multipole boundary element method for the symmetric boundary integral formulation. In *IABEM 2002*, UT Austin, TX, USA, May 28-30 2002. International Association for Boundary Element Methods.

[82] T. Rivlin. The Lebesgue constants for polynomial interpolation. In H. Garnir, K. Unni, and J. Williamson, editors, *Functional Analysis and its Applications*, volume 399 of *Lecture Notes in Mathematics*, pages 422–437. Springer Berlin / Heidelberg, 1974.

[83] S. Rjasanow and O. Steinbach. *The Fast Solution of Boundary Integral Equations*. Springer-Verlag, 2007.

[84] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60(2):187–207, 1985.

[85] V. Rokhlin. Diagonal forms of translation operators for the Helmholtz equation in three dimensions. *Applied and Computational Harmonic Analysis*, 1:82–93, 1993.

[86] M. Schanz. *Wave Propagation in Viscoelastic and Poroelastic Continua: A Boundary Element Approach*, volume 2 of *Lecture Notes in Applied Mechanics*. Springer-Verlag, Berlin, Heidelberg, New York, 2001.

[87] M. Schanz and H. Antes. A new visco- and elastodynamic time domain boundary element formulation. *Computational Mechanics*, 20:452–459, 1997.

[88] G. Schmidlin, Ch. Lage, and Ch. Schwab. Rapid solution of first kind boundary integral equations in $\mathbb{R}^3$. *Engineering Analysis with Boundary Elements*, 27(5):469–490, 2003.

[89] S. J. Smith. The Lebesgue constants for polynomial interpolation. *Annales Mathematicae et Informaticae*, 33:422–437, 2006.

[90] A. Sommerfeld. *Partial Differential Equations in Physics*. Academic Press, New York, 1949.

[91] O. Steinbach. *Numerical Approximation Methods for Elliptic Boundary Value Problems*. Springer, 2008.

[92] T. Takahashi, N. Nishimura, and S. Kobayashi. A fast BIEM for three-dimensional elastodynamics in time domain. *Engineering Analysis with Boundary Elements*, 28: 165–180, 2002.

[93] J. Tausch. The variable order fast multipole method for boundary integral equations of the second kind. *Computing*, 72:267–291, 2004.

[94] B. Weilharter, O. Bíró, H. Lang, and S. Rainer. Computation of the noise radiation of an induction machine using 3D FEM/BEM. *The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, 30(6):1737–1750, 2011.

[95] A. Zygmund. *Trigonometric Series*. Number 2 in Cambridge Mathematical Library. Cambridge University Press, 1959.