Graz University of Technology
Institute of Applied Mechanics

Preprint Series

Preprint No 01/2011

Fast Directional Multilevel Summation for Oscillatory Kernels based on Chebyshev Interpolation

Matthias Messner, Martin Schanz
Institute of Applied Mechanics, Graz University of Technology
Eric Darve
Mechanical Engineering Department,
Institute for Computational and Mathematical Engineering,
Stanford University

Published in: *Journal of Computational Physics*, **231**(4), 1175–1196, 2012
Latest revision: September 30, 2011

Abstract

Many applications lead to large systems of linear equations with dense matrices. Direct matrix-vector products become prohibitive, since the computational cost increases quadratically with the size of the problem. By exploiting specific kernel properties fast algorithms can be constructed.

A directional multilevel algorithm for translation-invariant oscillatory kernels of the type $K(x,y) = G(x-y)\,\mathrm{e}^{\imath k|x-y|}$, with $G(x-y)$ being any smooth kernel, will be presented. We will first present a general approach to build fast multipole methods (FMM) based on Chebyshev interpolation and the adaptive cross approximation (ACA) for smooth kernels. The Chebyshev interpolation is used to transfer information up and down the levels of the FMM. The scheme is further accelerated by compressing the information stored at Chebyshev interpolation points using ACA and QR decompositions. This leads to a nearly optimal computational cost with a small pre-processing time due to the low computational cost of ACA. This approach is in particular faster than performing singular value decompositions.

This does not address the difficulties associated with the oscillatory nature of K. For that purpose, we consider the following modification of the kernel $K^u = K(x,y)\,e^{-\imath k u\cdot(x-y)}$, where u is a unit vector (see Brandt [3]). We proved that the kernel K^u can be interpolated efficiently when $x-y$ lies in a cone of direction u. This result is used to construct an FMM for the kernel K.

Theoretical error bounds will be presented to control the error in the computation as well as the computational cost of the method. The paper ends with the presentation of 2D and 3D numerical convergence studies, and computational cost benchmarks.

1 Introduction

A direct evaluation of summations like

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

becomes prohibitive because the complexity grows quadratically with the problem size N. Numerous methodologies have been developed in the last decades which allow us to perform these summations efficiently such that the complexity grows like $\mathcal{O}(N \log N)$. Probably the most established ones are the fast multipole method (FMM, e.g., [12]), the panel clustering method (e.g., [15]) and those based on hierarchical matrices ($\mathcal{H}$-matrices, e.g., [14]). All these approaches take advantage of specific properties of the kernel function $K(x,y)$, such as low-rank properties of certain blocks in the matrix $K(x_i, y_j)$. Proper admissibility conditions provide a priori information on when these properties apply. The most common one is based on the distance between x_i and y_j and typically applies to isotropic kernels which are asymptotically smooth in $|x-y|$, e.g., kernel functions of elliptic boundary integral operators are of such type. An example of such admissibility condition is that if x and y are in clusters of radius R, then the distance between the two clusters must be at least $2\alpha R$, $\alpha > 1$. An admissibility condition for kernel functions of retarded boundary integral operators was introduced in [16]. Such functions additionally depend on the time. In the paper at hand, however, we focus on oscillatory kernel functions like

$$K(x,y) = G(x-y)\,\mathrm{e}^{\imath k|y-x|}, \quad \text{where } k \text{ is the wave number} \tag{2}$$

where $G(x-y)$ can be any asymptotically smooth function, that is the derivatives of G decay rapidly as $|x-y|$ becomes large. This will be made more precise later on.

In the low-frequency, the usual FMM analysis based on low-rank properties of the kernel is applicable and the complexity scales like $\mathcal{O}(N)$. However, in the high frequency regime (see [18]), the kernel is no longer low-rank. The rank of $\mathrm{e}^{\imath k|y-x|}$ depends on k and grows like $\mathcal{O}(kR)$, where R is the size of the cluster containing x and y (in the usual FMM setting). This is explained in [21] and also shown in this paper. Several solutions have been proposed to address this issue. Rokhlin [19] developed a high frequency FMM based on diagonal translation operators for the Helmholtz kernel. Diagonal operators were worked out by Greengard et al. [13] in the low and high frequency regimes. Another approach was taken by Darve [7], where they proposed a stable plane expansion for the whole frequency regime. A combined wide-band scheme that switches between different representations in order to cover both the high- and low-frequency regime is also presented in Cheng et al. [5].

Brandt [3] took a different approach. He took advantage of the fact that the modified kernel

$$K^u(x,y) = K(x,y)\,\mathrm{e}^{-\imath k\, u\cdot(x-y)} \tag{3}$$

is low-rank, independent of the wave number k, in the direction of the unit vector u. Along the same line, Engquist and Ying [9] developed the directional admissibility condition for oscillatory kernels (2). Based on this idea they constructed a fast directional multilevel algorithm for oscillatory kernels.

This work builds on this concept of directional admissibility condition. We show how low-rank representations can be constructed using Chebyshev polynomials even for oscillatory kernels. Our approach is applicable to any kernel of the form $G(x-y)\,\mathrm{e}^{\imath k|y-x|}$ where G is smooth. We build our scheme using the modified kernel (3) proposed by Brandt. Section 2 provides an analysis of the smoothness of this potential. In Section 3 we construct the fast directional summation method based on Chebyshev interpolation. The efficient treatment of the multiple-to-local (M2L) operator is covered in Section 4. In Section 5 we show that the overall complexity of the method is $O(N \log N)$. In the last Section, 6, we present numerical tests with interpolation error, low-rank approximation convergence and computational cost benchmarks. We also compare this FMM with the plane wave FMM of [4].

2 Directional low-rank representation of the oscillatory kernel

Our fast summation method is based on finding a low-rank representation of the kernel $K(x,y)$. As a basic tool to establish this property we use Chebyshev polynomials. Given a function $f(x)$ (in 1 dimension), we can approximate it by

$$f(x) \sim \sum_{m=0}^{\ell} a_m T_m(x), \tag{4}$$

where T_m are Chebyshev polynomials of the first kind. The coefficients a_m can be approximated by evaluating the function $f(x)$ at Chebyshev nodes, which serve as interpolation points. This process leads to a low-rank representation of the form:

$$K(x,y) \sim \sum_{m=0}^{\ell} T_m(x) \sum_{n=0}^{\ell} a_{mn}\, T_n(y).$$

The interpolation error can be bounded by:

$$\left| f(x) - \sum_{m=0}^{\ell} a_m T_m(x) \right| \leq \frac{M}{\rho^{\ell}(\rho-1)}, \quad \text{for all } -1 \leq x \leq 1,$$

where $\rho > 1$ and M is such that

$$|f(z)| \leq M, \quad \text{for all } z \in E_\rho, \quad E_\rho = \left\{ z \in \mathbb{C} \mid z = \frac{\rho \mathrm{e}^{\imath\theta} + \rho^{-1}\mathrm{e}^{-\imath\theta}}{2} \right\} \tag{5}$$

holds. In eqn. (5) it is assumed that $f(z)$ is analytic inside the ellipse E_ρ. Hence, a key element is that M should not be too large.

A case that is more difficult for this interpolation procedure is $e^{\imath kx}$. For $z = \imath\,(-\rho + \rho^{-1})/2$, (i.e., $\theta = -\pi/2$ in eqn. (5)), we get

$$\mathrm{e}^{\imath kz} = \mathrm{e}^{k(\rho-\rho^{-1})/2}.$$

The constant M can not be bounded by a constant independent of k if we keep ρ fixed. In fact, it grows exponentially fast with k. This is reflected by the fact that it is not possible to interpolate

$e^{\imath kx}$ using Chebyshev polynomial with a fixed order ℓ and a fixed accuracy, while varying k. As k increases, the wavelength must be resolved by increasing correspondingly the order of the expansion.

Even though the reasoning is a little more complicated for our kernel $K(x,y)$, the main conclusion is that we cannot achieve a k independent low-rank approximation by directly applying the Chebyshev interpolation. However, this problem can be addressed by constructing a slightly different kernel that remains "small" in the complex plane independent of k. Let us decompose $K(x,y)$ into

$$K(x,y) = G(x,y)\, \mathrm{e}^{\imath k(|x-y|-u\cdot(x-y))}\, \mathrm{e}^{\imath k u\cdot(x-y)} \tag{6}$$

$$= K^u(x,y)\, \mathrm{e}^{\imath k u\cdot(x-y)}. \tag{7}$$

Let us introduce some notations. Consider a cluster X centered at c_x and a cluster Y centered at c_y containing x and y, respectively. We denote: $r_x = x - c_x$, $r_y = y - c_y$, $r = r_x - r_y$, $c = c_x - c_y$. We also define a unit vector u. All these notations are shown in Fig. 1. Further, we will assume that the clusters containing x and y have radius 1/2, so that the Cartesian coordinates of r are in the interval $[-1,1]$. This can always be made true by appropriately scaling k.

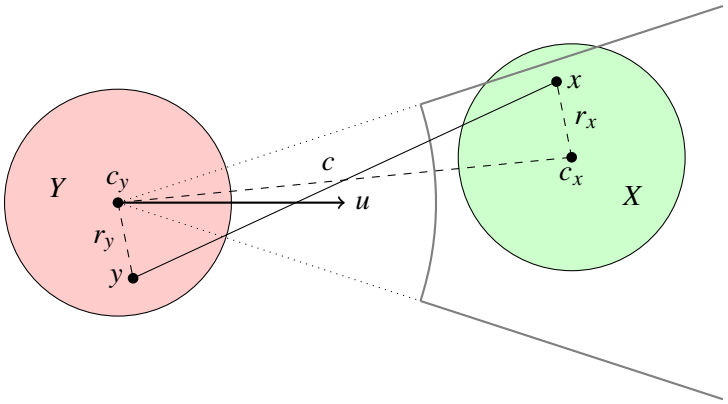


Figure 1: Directional admissible clusters

With this notation we can set $x - y = c + r$, hence we have the following expansion for the argument of the exponential in $K^u(x,y)$

$$|c+r| - u\cdot(c+r) = |c| - u\cdot c + (c/|c| - u)\cdot r + \frac{1}{2}\frac{|r|^2}{|c|} - \frac{1}{2}\frac{(c\cdot r)^2}{|c|^3} + O(|r|^3), \tag{8}$$

by using a Taylor expansion in $|r|$ of the square root. Let us now consider the kernel $K^u(x,y)$ inside the ellipse E_ρ

$$K^u(z) = G(z)\, \exp\Big(\imath k\Big(|c| - u\cdot c + (c/|c| - u)\cdot z + \frac{1}{2}\frac{|z|^2}{|c|} - \frac{1}{2}\frac{(c\cdot z)^2}{|c|^3} + \cdots\Big)\Big),$$

by extending $r \in \mathbb{R}^2$ to $z \in \mathbb{C}^2$. We need to show that it is bounded independently of k. Let us assume that $G(z)$ is analytic in E_ρ for some $\rho > 1$. Let us further assume that:

$$k\left|\frac{c}{|c|} - u\right| \le A, \qquad \frac{k}{|c|} \le A.$$

Then there exists a constant M independent of k that bounds $K^u(z)$ inside E_ρ.

These bounds are normalized since we made the assumption that the radius of the clusters is 1/2. If we consider the general case of a cluster of diameter w (introducing back dimensional variables), we get the corresponding assumptions:

$$kw\left|\frac{c}{|c|}-u\right|\leq A \qquad \frac{kw^2}{|c|}\leq A. \tag{9}$$

The first equation corresponds essentially to a cone of aperture $O(1/kw)$, while the second condition corresponds to a minimum separation of $O(kw^2)$ between two clusters. The situation is depicted in Fig. 1. In summary, we have proved that if eqn. (9) holds, we can find a low rank representation of the kernel $K^u(x,y)$ by using Chebyshev polynomials. The number of terms in the expansion, i.e., the required rank ℓ for a given accuracy, is independent of k and of the cluster diameter w.

3 Fast directional summation method based on Chebyshev interpolation

Let us start with the introduction of some basic notations regarding the Chebyshev interpolation scheme. We have an $\ell-1$ degree polynomial $p_{\ell-1}(x)$, which interpolates the function $f(x)$ and takes the form:

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

with the interpolation operators $S_\ell(x,\bar{x})$ given by the explicit formula

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

The points $\bar{x}_m$ are Chebyshev nodes. If we need to extend this interpolation formula to an arbitrary interval $x\in[a,b]$, we can use the affine mapping

$$\Phi:[-1,1]\mapsto[a,b] \quad \text{with} \quad \Phi(x)=\frac{a+b}{2}+\frac{b-a}{2}\,x$$

For reference the inverse mapping is also needed and is given by $\Phi^{-1}(x)=\frac{2x-b-a}{b-a}$.

For the general case of $\mathbb{R}^d$, we introduce an arbitrary axis-parallel box $[a,b]\subset\mathbb{R}^d$ defined as

$$[a,b]=[a_1,b_1]\times\cdots\times[a_d,b_d]. \tag{10}$$

In this case the interpolation operators $S^d_\ell(x,\bar{x})$ with $x\in[a,b]$ are constructed by means of the tensor product

$$S^d_\ell(x,\bar{x})=S_\ell(x_1,\bar{x}_1)\,\cdots\,S_\ell(x_d,\bar{x}_d). \tag{11}$$

3.1 The far-field in the low- and high frequency regime

Before we start with the construction of a fast summation method we need to partition the computational domain $\Omega \subset \mathbb{R}^d$ into clusters of equal diameter w. This allows us to separate near- and far-field. We introduce two clusters $X \subset \Omega$ and $Y \subset \Omega$ containing M and N particles, respectively. If cluster X and Y are sufficiently separated their interaction can be computed efficiently. Whereas, if X and Y are nearby clusters their interaction must be computed directly. We treat only the far-field case in the following.

Two regimes, depending on the cluster diameter w, are of interest when dealing with oscillatory kernels. There exists a constant B such that we are in the *low frequency regime* whenever $w \leq B/k$ and in the *high frequency regime* whenever $w > B/k$. In the low frequency regime, eqn. (9) is always true. If the kernel $G(x-y)$ has a singularity at $x = y$, we further need the usual well-separated condition for the two clusters: $|c| \geq \alpha 2w$ where $\alpha > 1$ (in the usual FMM setting with an oct-tree, α can be chosen equal to $2/\sqrt{3}$). With this assumption, the kernel $K(x,y)$ can be interpolated accurately using the usual Chebyshev interpolation approach. A fast summation method can be constructed as presented in Fong and Darve [10]

$$K(x,y) \sim \sum_{m=1}^{L} S_\ell(\bar{x}_m, x) \sum_{n=1}^{L} K(\bar{x}_m, \bar{y}_n) S_\ell(\bar{y}_n, y), \tag{12}$$

where $L = \ell^d$ denotes the number of interpolation points on each cluster.

We would like to clarify an important but somewhat confusing point. The entire fast multipole scheme relies fundamentally on the concept of interpolation. However the actual calculation that is carried out involves two different but similar looking operators: the anterpolation and interpolation operators. Consider the sum:

$$f_i = \sum_j K(x_i, y_j)\sigma_j$$

Using the interpolation formula above, Eqn. (12), we calculate an approximation of the sum in three steps:

- Anterpolation or multipole to multipole operator (M2M): $W_n = \sum_j S_\ell(\bar{y}_n, y_j)\sigma_j$
- Kernel evaluation or multipole to local operator (M2L): $F_m = \sum_{n=1}^{L} K(\bar{x}_m, \bar{y}_n) W_n$
- Interpolation or local to local operator (L2L): $f_i \sim \sum_{m=1}^{L} S_\ell(\bar{x}_m, x_i) F_m$

Even though the operators M2M and L2L are different (they are transpose of each other), they are both derived from interpolation operators. The error analysis therefore only requires studying errors introduced by the interpolation algorithm. In that respect the anterpolation (M2M) and interpolation steps (L2L) are not different.

The high frequency regime is more difficult to handle. In section 2, we have shown that we can find a low-rank representation of $K^u(x,y)$ by means of the Chebyshev polynomials

$$K^u(x,y) = \sum_{m=1}^{L} S_\ell(x, \bar{x}_m) \sum_{n=1}^{L} K^u(\bar{x}_m, \bar{y}_n) S_\ell(\bar{y}_n, y) + \varepsilon(\ell),$$

if two clusters X and Y fulfill the minimal separation condition $|c| \geq kw^2/A$ and the maximal cone aperture $|c/|c| - u| \leq A/kw$. The error is bounded by a function $\varepsilon(\ell) = O(1/\gamma^\ell)$, which is independent of k and w. In terms of the original kernel $K(x,y)$ the low rank form becomes

$$K(x,y) \sim \mathrm{e}^{\imath k u\cdot x} \sum_{m=1}^{L} S_\ell(x,\bar{x}_m)\,\mathrm{e}^{-\imath k u\cdot \bar{x}_m} \sum_{n=1}^{L} K(\bar{x}_m,\bar{y}_n)\,\mathrm{e}^{\imath k u\cdot \bar{y}_n} S_\ell(\bar{y}_n,y)\,\mathrm{e}^{-\imath k u\cdot y}. \tag{13}$$

With the notations above, the constant B can be chosen as:

$$B = A\ \min(2\alpha,\ 1/2)$$

3.2 Directional single level summation

Considering the directional low-rank property of $K^u(x,y)$ we need to partition the far-field into a set of cones of direction $\{u_c\}_{c=1}^{C}$ and each of aperture $O(1/kw)$. Each cone contains a set of interacting clusters $\{Y_t\}_{t=1}^{T_c}$. For the sake of readability we present the directional single level summation in the following by means of a single interaction. We consider a cluster Y located in the cone of direction u and centered at cluster X. Hence, we can substitute eqn. (13) into (1) and we get the contribution from the sources σ_j in Y to the field values f_i in X

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

for $i = 1,\ldots,M$. We can efficiently compute the summation by splitting up this equation into three steps:

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

$$W_n = \mathrm{e}^{\imath k u\cdot \bar{y}_n} \sum_{j=1}^{N} S_\ell(\bar{y}_n,y_j)\,\mathrm{e}^{-\imath k u\cdot y_j}\,\sigma_j \quad \text{for } n = 1,\ldots,L. \tag{14}$$

2. *M2L* operation: Compute field values at interpolation points $\bar{x}_m$

$$F_m = \sum_{n=1}^{L} K(\bar{x}_m,\bar{y}_n)\,W_n \quad \text{for } m = 1,\ldots,L. \tag{15}$$

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

$$f_i^{c,t} = \mathrm{e}^{\imath k u\cdot x_i} \sum_{m=1}^{L} S_\ell(x_i,\bar{x}_m)\,\mathrm{e}^{-\imath k u\cdot \bar{x}_m}\,F_m \quad \text{for } i = 1\ldots,M. \tag{16}$$

The computational cost of the steps 1 and 3 are of $O(LN)$ and $O(LM)$, respectively. The cost of step 2 is of $O(L^2)$. Hence for $L \ll N$ and M, this summation algorithm scales like $O((M+N)L)$. A direct evaluation has a complexity of $O(MN)$.

In order to get the influence of the entire computational domain Ω we have to sum up all near- and far-field contributions. The evaluation of the near-field is performed directly between all interacting clusters. For the far-field we have to gather the contributions of all interacting clusters $\{Y_t\}_{t=1}^{T_c}$ in the cones of direction $\{u_c\}_{c=1}^{C}$

$$f_i \sim f_i^{\text{near-field}} + \sum_{c=1}^{C} \sum_{t=1}^{T_c} f_i^{c,t} \quad \text{for all } i = 1, \ldots, M.$$

3.3 Directional multilevel summation

For the construction of a multilevel method we need to hierarchically partition the computational domain $\Omega \subset \mathbb{R}^d$ into clusters. This partitioning is recursive so that we obtain a tree. Tree level 0 is equivalent to the entire computational domain, while tree level $\nu+1$ is obtained from level ν by subdividing each cluster into 2^d equal child clusters. The procedure stops once the low frequency regime is reached, i.e., all but the bottom most tree level are in the high frequency regime. The cluster tree depth depends on the wave number k, but the cluster diameter w only depends on the tree level ν.

Moreover, we need to partition the far field of all clusters in the high frequency regime into a set of cones $\{u_c\}_{c=1}^{C}$, each of aperture $O(1/kw)$. For that we adopt the procedure described in Engquist and Ying [9]. In this procedure, the cone aperture gets divided by two as we go up the tree. This results in the fact that the M2M and L2L operators become *directional*.

These operators are constructed as follows. We discuss the 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 multipole expansion with coefficients W_n^u for direction u at level $\nu+1$ [see Eqn. (14)]:

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

We want to calculate the contribution of W_n^u to the multipole coefficients $W_m^{u'}$ of the parent cluster at level ν. Recall the general high-frequency interpolation formula of the kernel [see Eqn. (13)]:

$$K(x, y_j) \sim \sum_{n=1}^{L} K(x, \bar{y}_n)\, \mathrm{e}^{\imath k u \cdot \bar{y}_n}\, S_\ell(\bar{y}_n, y_j)\, \mathrm{e}^{-\imath k u \cdot y_j}.$$

This formula is accurate as long as K^u 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 $\bar{\bar{y}}_s$. However such an interpolation operator requires using direction u' to maintain the accuracy of the scheme. Indeed as we have proved above, 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 is then as

follows:

$$K(x,\bar{y}_n) \sim \sum_{s=1}^{L} K(x,\bar{\bar{y}}_s)\,\mathsf{e}^{\imath k u'\cdot\bar{\bar{y}}_s}\,S'_\ell(\bar{\bar{y}}_s,\bar{y}_n)\,\mathsf{e}^{-\imath k u'\cdot\bar{y}_n}.$$

where S'_ℓ is the interpolation function 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. Stage 1: we interpolate from $\bar{y}_n$ to y_j. This leads to the following definition:

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

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 ν because the cone u' for the parent cluster is strictly contained inside the cone u at level $\nu+1$ for the child cluster. Stage 2: we interpolate from $\bar{\bar{y}}_s$ to $\bar{y}_n$. This leads to the M2M formula we were after:

$$W_s^{u'} = \mathsf{e}^{\imath k u'\cdot\bar{\bar{y}}_s}\sum_{n} S'_\ell(\bar{\bar{y}}_s,\bar{y}_n)\,\mathsf{e}^{-\imath k u'\cdot\bar{y}_n}\,W_n^u$$

A similar derivation shows that the L2L operator is given by:

$$F_m^u = \mathsf{e}^{\imath k u'\cdot\bar{x}_m}\sum_{l} S'_\ell(\bar{\bar{x}}_l,\bar{x}_m)\,\mathsf{e}^{-\imath k u'\cdot\bar{\bar{x}}_l}\,F_l^{u'}$$

The L2L operator is the transpose and complex conjugate operator of M2M.

We now present the different steps in the multilevel scheme. For the sake of readability we omit the indices c and t indicating direction and interaction. Other notations have been simplified for clarity. This leads to some incorrect notations in places but hopefully the context makes it clear what the different variables are.

1. Construct the **Cluster Tree** such that all but the bottom most level are in the high frequency regime.

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 $O(kw^2)$ becomes larger than the computational domain. The upward pass consists of the M2M operation (anterpolation):

   - Perform the *non directional M2M* operation at the leaf level (low frequency regime): Anterpolate non directional equivalent sources W_n at interpolation points $\bar{y}_n$ from sources σ_j at initial points y_j

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

     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 M2M* operation for all directions $\{u_c\}_{c=1}^{C}$: Anterpolate directional equivalent sources W_s^u at interpolation points $\bar{\bar{y}}_s$ from sources W_n^u at points $\bar{y}_n$ of all child clusters

$$W_s^{u'} = \mathrm{e}^{\imath k u' \cdot \bar{\bar{y}}_s} \sum_{n=1}^{2^d L} S_\ell(\bar{\bar{y}}_s, \bar{y}_n)\, \mathrm{e}^{-\imath k u' \cdot \bar{y}_n}\, W_n^u \quad \text{for } s = 1, \ldots, L.$$

  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** (multipole to local — M2L) consists in calculating all the partial local expansions, $F_m^{u,\mathrm{p}}$, for all clusters at all levels:

$$F_m^{u,\mathrm{p}} = \sum_{n=1}^{L} K(\bar{x}_m, \bar{y}_n)\, W_n^u \quad \text{for } m = 1, \ldots, L.$$

   For low-frequency clusters, use W_n and F_m^{p}.

4. The **Downward Pass** (interpolation) starts at the level the upward pass has ended, and ends at the leaf level. At the top of the tree $F_m^u = F_m^{u,\mathrm{p}}$.

   - For high-frequency clusters, perform the *directional L2L* operation for all directions $\{u'_c\}_{c=1}^{C'}$: Interpolate directional field values F_m^u at interpolation points $\bar{x}_m$ of all child clusters from directional field values $F_l^{u'}$ at interpolation points $\bar{\bar{x}}_l$

$$F_m^u = F_m^{u,\mathrm{p}} + \mathrm{e}^{\imath k u' \cdot \bar{x}_m} \sum_{l=1}^{L} S_\ell(\bar{\bar{x}}_l, \bar{x}_m)\, \mathrm{e}^{-\imath k u' \cdot \bar{\bar{x}}_l}\, F_l^{u'} \quad \text{for } m = 1, \ldots, 2^d L.$$

     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 L2L* operation: Interpolate field values F_m at $\bar{x}_m$ from non directional field values F'_l at interpolation points $\bar{\bar{x}}_l$

$$F_m = F_m^{\mathrm{p}} + \sum_{l=1}^{L} S_\ell(\bar{\bar{x}}_l, \bar{x}_m)\, F'_l \quad \text{for } m = 1, \ldots, 2^d L.$$

5. Evaluate the **Near Field** contribution and add it to the interpolated far field contribution:

$$\sum_{l=1}^{L} S_\ell(\bar{x}_l, x_i)\, F_l$$

4 Efficient treatment of the M2L operator

In each multilevel method the M2L operation adds the largest contribution to the computational cost. The reason for that is the fact that the M2L operation has 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.

The M2L operator evaluates the field due to source values located in cluster Y at points located in cluster X. The efficient treatment of this operator is based on the fact that there exists a low-rank representation. Let $K^u(x,y)$ and $K(x,y)$ be the generating kernels of the M2L operators K^u and K, respectively. We know from section 2 that $K^u(x,y)$ has a low-rank representation. But what about $K(x,y)$? Looking at the decomposition

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

we conclude that $K(x,y)$ has a low-rank representation as well. Hence, both in the low- and the high-frequency regime we can use the original, unmodified kernel $K(x,y)$ to compute the M2L operators. Remember, that each cluster contains $L = \ell^d$ interpolation points. These points might be directional, in the sense that they allow interpolating $K(x,y)$ only within a specific cone (see Eqns. 9). Hence, we write the M2L operator $\mathsf{K} \in \mathbb{C}^{L\times TL}$ for a given interaction list as

$$\mathsf{K} = [\mathsf{K}_1, \mathsf{K}_2, \ldots, \mathsf{K}_t, \ldots, \mathsf{K}_{T-1}, \mathsf{K}_T], \tag{17}$$

with $\mathsf{K}_t \in \mathbb{C}^{L\times L}$ being the M2L operator for cluster X and Y_t. The total number of clusters in the (possibly directional) interaction list is T and these clusters are denoted by $\{Y_t\}_{t=1}^{T}$.

In the following, we first apply the adaptive cross approximation (ACA) to obtain a low rank approximation of the M2L operator (see section 4.1). In section 4.2 we show how we can further reduce the computational cost of this operation by means of QR decompositions. Other methods can be applied as well, however we found ACA combined with QR to be an effective approach.

4.1 Adaptive cross approximation

A remark to the notation in this section: $(\mathsf{K})_{ij}$ denotes the ij-th entry of the matrix K, whereas $(\mathsf{K})_{i:}$ and $(\mathsf{K})_{:j}$ are the i-th row and j-th column of K, respectively. The idea of the ACA is to approximate the M2L operator such that

$$\mathsf{K} = \mathsf{U}\mathsf{V}^* + O(\epsilon), \quad \text{with } \mathsf{K} \in \mathbb{C}^{L\times TL}. \tag{18}$$

The rows and columns of the approximant $\mathsf{U}\mathsf{V}^*$, $\mathsf{U} \in \mathbb{C}^{L\times\kappa}$ and $\mathsf{V} \in \mathbb{C}^{TL\times\kappa}$ are computed for $\kappa = 1, 2, \ldots$ as

$$\begin{aligned}
\hat{\mathsf{u}}_\kappa &= (\mathsf{K})_{:j_\kappa} - \sum_{n=1}^{\kappa-1} (\overline{\mathsf{v}}_n)_{j_\kappa}\, \mathsf{u}_n \quad \text{with } j_\kappa := \mathrm{ArgMax}|(\mathsf{v}_{\kappa-1})_j| \\
\mathsf{u}_\kappa &= \gamma_\kappa\, \hat{\mathsf{u}}_\kappa \quad \text{with the pivot } \gamma_\kappa = (\hat{\mathsf{u}}_\kappa)_{i_\kappa}^{-1} \text{ and } i_\kappa := \mathrm{ArgMax}|(\hat{\mathsf{u}}_\kappa)_i| \\
\mathsf{v}_\kappa &= [(\mathsf{K})_{i_\kappa:}]^* - \sum_{n=1}^{\kappa-1} (\mathsf{u}_n)_{i_\kappa}\, \overline{\mathsf{v}}_n.
\end{aligned} \tag{19}$$

where * is the transpose conjugate operator, and $\overline{\mathsf{v}}$ is the complex conjugate of v. The pivot γ_κ is chosen to be the largest entry in modulus of $\hat{\mathsf{u}}_\kappa$. It determines the row and column indices i_κ and j_κ of the κ-th approximation step, respectively. Further hints for the right choice of the pivots γ_κ and the initial index j_1 can be found in [2]. The approximation stops if the prescribed accuracy ϵ is reached, i.e., the following criterion holds

$$\|\mathsf{u}_{\kappa+1}\|_F \, \|\mathsf{v}_{\kappa+1}\|_F < \epsilon \|(\mathsf{UV}^*)_{\kappa+1}\|_F. \tag{20}$$

Note, the entire matrix K is never generated. By using the definition of the absolute value $|z| = \sqrt{z\overline{z}}$, $z \in \mathbb{C}$, the Frobenius norm of the approximant can be formulated recursively

$$\begin{aligned}\|(\mathsf{UV}^*)_{\kappa+1}\|_F^2 = \|(\mathsf{UV}^*)_\kappa\|_F^2 + \sum_{n=1}^{\kappa} \left[(\mathsf{u}_n \cdot \overline{\mathsf{u}}_{\kappa+1})(\mathsf{v}_n \cdot \overline{\mathsf{v}}_{\kappa+1}) + (\overline{\mathsf{u}}_n \cdot \mathsf{u}_{\kappa+1})(\overline{\mathsf{v}}_n \cdot \mathsf{v}_{\kappa+1})\right] \\ + \|\mathsf{u}_{\kappa+1}\|_F^2 \|\mathsf{v}_{\kappa+1}\|_F^2.\end{aligned}$$

Hence, all algebraic evaluations in (19) and (20) can be performed with $O(\kappa^2(L+TL))$ floating point operations. If the computational costs for generating matrix entries dominates by far the costs needed for the ACA algorithm, the complexity and memory requirement scale like $O(\kappa(L+TL))$ (see [1]).

4.2 Fast convolution based on a QR decomposition

We can write the M2L operator K in a twofold way, either by organizing the individual K_t as a row vector or as a column vector

$$\begin{aligned}\mathsf{K}^{(\mathrm{row})} &= [\mathsf{K}_1, \mathsf{K}_2, \ldots, \mathsf{K}_t, \ldots, \mathsf{K}_{T-1}, \mathsf{K}_T] \\ \text{or} \quad \mathsf{K}^{(\mathrm{col})} &= [\mathsf{K}_1; \mathsf{K}_2; \ldots; \mathsf{K}_t; \ldots; \mathsf{K}_{T-1}; \mathsf{K}_T].\end{aligned} \tag{21}$$

We use the , and ; notations to distinguish column and row ordering. Their low rank representation obtained by means of the ACA reads as

$$\begin{aligned}\mathsf{K}^{(\mathrm{row})} &\sim \mathsf{U}[\mathsf{V}_1^*, \mathsf{V}_2^*, \ldots, \mathsf{V}_t^*, \ldots, \mathsf{V}_{T-1}^*, \mathsf{V}_T^*] \\ \text{or} \quad \mathsf{K}^{(\mathrm{col})} &\sim [\mathsf{A}_1; \mathsf{A}_2; \ldots, \mathsf{A}_t; \ldots, \mathsf{A}_{T-1}; \mathsf{A}_T]\mathsf{B}^*.\end{aligned} \tag{22}$$

Hence, for an individual M2L operator $\mathsf{K}_t \sim \mathsf{U}\mathsf{V}_t^* \sim \mathsf{A}_t\mathsf{B}^*$ is true and by means of QR decompositions we can rewrite it as

$$\mathsf{K}_t \sim \mathsf{Q}_U \mathsf{R}_U (\mathsf{Q}_{V_t} \mathsf{R}_V)^* \sim \mathsf{Q}_{A_t} \mathsf{R}_A (\mathsf{Q}_B \mathsf{R}_B)^*. \tag{23}$$

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 (23) reads as

$$\mathsf{K}_t \sim \mathsf{Q}_U \Phi \mathsf{Q}_{V_t}^* \sim \mathsf{Q}_{A_t} \Psi \mathsf{Q}_B^*. \tag{24}$$

Now, by means of (24) we can rearrange an individual M2L operator K_t as follows

$$\begin{aligned}
\mathsf{K}_t &\sim \mathsf{Q}_{A_t} \Psi \mathsf{Q}_B^* \\
&\sim \mathsf{Q}_{A_t} \Psi (\mathsf{Q}_B^* \mathsf{Q}_B) \mathsf{Q}_B^* \\
&\sim \mathsf{K}_t (\mathsf{Q}_B \mathsf{Q}_B^*) \\
&\sim \mathsf{Q}_U \Phi \mathsf{Q}_{V_t}^* (\mathsf{Q}_B \mathsf{Q}_B^*) \\
&\sim \mathsf{Q}_U (\mathsf{Q}_U^* \mathsf{Q}_U) \Phi \mathsf{Q}_{V_t}^* (\mathsf{Q}_B \mathsf{Q}_B^*) \\
&\sim \mathsf{Q}_U (\mathsf{Q}_U^* \mathsf{K}_t \mathsf{Q}_B) \mathsf{Q}_B^* \qquad \Rightarrow \quad \boxed{\mathsf{K}_t \sim \mathsf{Q}_U \mathsf{C}_t \mathsf{Q}_B^*,}
\end{aligned} \tag{25}$$

with $\mathsf{C}_t \in \mathbb{C}^{\kappa \times \kappa}$ being computed as $\mathsf{C}_t \sim \Phi \mathsf{Q}_{V_t}^* \mathsf{Q}_B$ or $\mathsf{C}_t \sim \mathsf{Q}_U^* \mathsf{Q}_{A_t} \Psi$ (the two expressions are equal with the low-rank approximation errors).

The cost of the pre-computation increases slightly due to this representation of K_t. However, the overall memory requirement and computational cost for M2L operations decrease substantially. Instead of storing UV^*, which is of order $O((L+TL)\kappa)$, only one Q_U and one Q_B per (directional) interaction list, and one C_t per interaction, need to be stored. Hence, the memory requirement per interaction list gets reduced to $O(2L\kappa + T\kappa^2)$. The computational cost must be analyzed in a slightly different way. Due to the fact that we can write $\mathsf{K}_t \sim \mathsf{UV}_t^* \sim \mathsf{Q}_U \mathsf{C}_t \mathsf{Q}_B^*$ we can shift the application of Q_B^* and Q_U to the M2M and L2L operation, respectively. Hence, the M2L operation gets reduced from $O(2L\kappa)$ to only $O(\kappa^2)$ (a multiplication by C_t). Remember that this operation has to be performed for each cluster with all the clusters in its interaction list. Hence, the savings are significant. A similar method is described in [10].

Summary and discussion. We make a brief pause to review the methods described so far. We first introduced an interpolation technique, based on Chebyshev polynomials, which can be used to construct low-rank representations. We explained how the directional admissibility condition could be used to construct low-rank approximations of oscillatory kernels. We then introduced ACA as a way to construct low-rank representations and explained how QR decompositions can be used to further reduce the cost of applying the M2L operator. Essentially, ACA allows to construct an approximate rank κ representation, while QR reduces the cost of M2L to a multiplication by a $\kappa \times \kappa$ matrix.

Stepping back a bit, it seems that the Chebyshev interpolation method is not needed, since ACA could provide in the first place the desired low-rank representation. The difficulty in this approach is the cost of the pre-computation and this is primarily the role the Chebyshev interpolation is playing. With the proposed algorithm, ACA is applied only to matrices constructed from Chebyshev nodes, that is they are independent of the points x_i and y_j. This pre-computation cost grows slower than N (in fact like $O(N^{1/2})$). In addition, this calculation can be re-used for many different calculations. In contrast, obtaining low-rank representations directly using ACA would require a much larger pre-computing time, and the resulting algorithm would, in fact, no longer scale like $O(N \log N)$. The complexity analysis of the proposed algorithm is provided in more details in the next section.

5 Complexity

In this section we present a general estimate of the computational complexity of the presented algorithm. We do not examine the case where as N goes to ∞ the distribution of the points is allowed to become in-homogeneous, 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 domain $\Omega \subset \mathbb{R}^d$ is then L (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 $O(N \log N)$. 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 $O(N)$ computational cost. With our assumption, the number of levels in the cluster tree is of order $O(\log L)$.

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

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

$$O\Big(\sum_{\nu=0}^{\log L} (L/2^\nu)^{d-1}\, 2^{\nu d}\Big) = O\Big(L^{d-1} \sum_{\nu=0}^{\log L} 2^\nu\Big) = O(L^d)$$

The complexity is of order $O(L^d) = O(N)$.

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 $O(w^2)$ and the maximal distance of $O(4w^2)$. Per cone, this represents $O(w)$ clusters. Since there are $O(w^{d-1})$ cones, the total number of interactions per cluster in all directions is $O(w^d)$. Each interacting cluster involves a constant number of operations, thus we end up with a complexity of $O((L/w)^d\, w^d) = O(L^d) = O(N)$ per level, that is the cost is approximately constant at all levels. We conclude that the total complexity is $O(N \log N)$.

We never reach $w = O(L)$, 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 $w = O(\sqrt{L})$. The dimensions don't seem to match because we made our problem dimensionless by choosing $k = 1$. With dimensions, the condition reads: $w = O(\sqrt{L/k})$. 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 $\sqrt{L}$, and therefore becomes small compared to the domain size L. This is in contrast with most FMMs in which the size of the largest clusters is on the order of the domain diameter. The difference is that in our approach

	Plane Wave FMM [7]	Directional FMM
Case 1	$O(N)$	$O(N \ln N)$
Case 2	$O(N \ln N)$	$O(N \ln N)$
Case 3	$O(N^{1+1/(d-2)})$	$O(N \ln N)$

Table 1: Complexity of the plane wave FMM [7] and the directional FMM in 3 cases: 1) points distributed in a volume Ω of dimension d, 2) on a surface of dimension $d-1$ in Ω, and 3) along a curve of dimension $d-2$ in Ω.

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., [6, 7], the rank grows like the size of the cluster (but the size of the interaction list is bounded).

We now outline the proof for case 2) in which 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 with possibly many directions having no interactions. However for all cones the total number of interactions is $O(w^2(w^2)^{d-2}/w^{d-1}) = O(w^{d-1})$. The complexity at a given level is $O((L/w)^{d-1}w^{d-1}) = O(L^{d-1}) = O(N)$. The total complexity is $O(N \log N)$ as before.

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

The complexity of the fast multipole method based on plane waves expansions [7] varies depending on the case. For a cluster of size w, the number of multipole coefficients is $O(w^{d-1})$. However the number of clusters varies depending on the case. In case 1), the total cost at each level is $O(w^{d-1}(L/w)^d) = O(L^d/w)$. The total cost of the method is then $O(L^d) = O(N)$. In case 2), the cost at each level is $O(w^{d-1}(L/w)^{d-1}) = O(L^{d-1})$ and the total cost of the method is $O(N \ln N)$. In case 3), the cost at each level is $O(w^{d-1}(L/w)^{d-2}) = O(wL^{d-2})$, and the total cost of the method is $O(L^{d-1}) = O(N^{1+1/(d-2)})$.

In practice, it is rare to encounter case 3, where points are on a manifold of dimension $d-2$ (a curve). However the analysis applies if we consider the case of an object that is very elongated along one dimension. For example define a matrix $M = U\Gamma U^T$ where U is orthonormal and Γ is a diagonal matrix with entries (1,1,s). Take a set Ω and apply the map M: $x \to Mx$, $x \in \Omega$. Discretize the resulting set $M\Omega$ using a fixed number of points per wavelength (here we assumed $k = 1$ so we keep the density of points constant). Then as $s \to \infty$, the object becomes very elongated; we have $N \to \infty$, and the complexity of the plane wave FMM [7] is $O(N^{1+1/(d-2)}) = O(N^2)$ (with $d = 3$).

The results are summarized in Table 1.

6 Numerical studies

The presented method is based on two different approximations: the low-rank representation of the kernel based on Chebyshev interpolation and the adaptive cross approximation (ACA). In the next sections 6.1 and 6.2 we present error estimates and convergence studies and conclude with some numerical benchmarks in section 6.3 and 6.4.

6.1 Error estimates for Chebyshev interpolation

To investigate the Chebyshev interpolation error we use an estimate from [17]: 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 holds

$$\left| f(x) - \sum_{m=1}^{\ell+1} S_{\ell+1}(x, \bar{x}_m) f(\bar{x}_m) \right| \leq \frac{(\rho + \rho^{-1})\, M}{(\rho^{\ell+1} - \rho^{-\ell-1})(\rho + \rho^{-1} - 2)} \tag{26}$$

for all $-1 \leq x \leq 1$ and with M and E_ρ defined in (5). For a given interpolation order ℓ 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 generic clusters $X, Y \subset \mathbb{R}^d$ that are centered at c_x and c_y, respectively. As illustrated in figure 1 we introduce $c = c_x - c_y$ and $r = x - y - c$. Then, we can state $x - y = c + r$ and the kernel we deal with becomes $K(r)$ and can be extended to $K(z)$ with $z \in \mathbb{C}^d$.

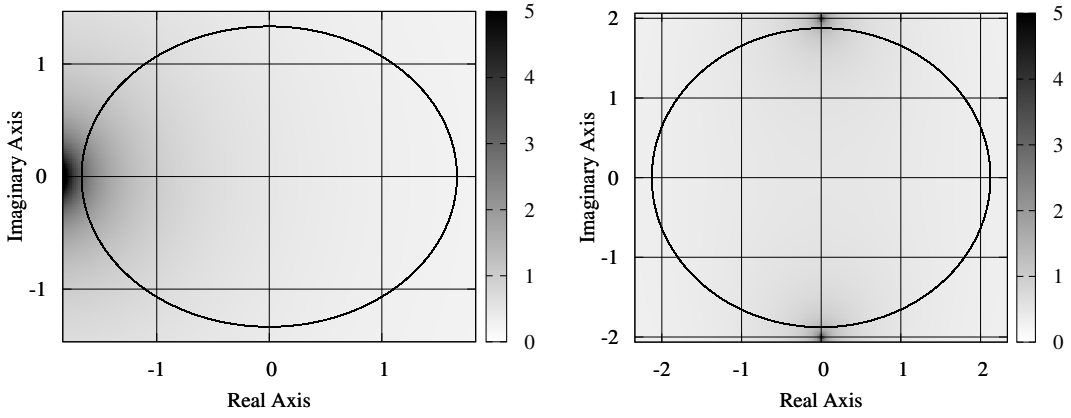
In the following, we analyze the interpolation error of two different kernels. In both cases we let the clusters $X, Y \subset \mathbb{R}^d$ be of diameter $w = 1$ and the cluster X be centered at the origin $c_x = [0,0]$. The center c_y of cluster Y changes.

6.1.1 An asymptotically smooth kernel $G(r) = \frac{1}{|c+r|}$

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

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

Figure 2 shows the modulus of the smooth kernel $G(r)$ in the complex plane for these two



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

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

cases. The ellipse E_ρ is chosen such that $G(r)$ remains analytic on it. 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. 3. The error decay is of order $O(1/\rho^{\ell})$ as expected.

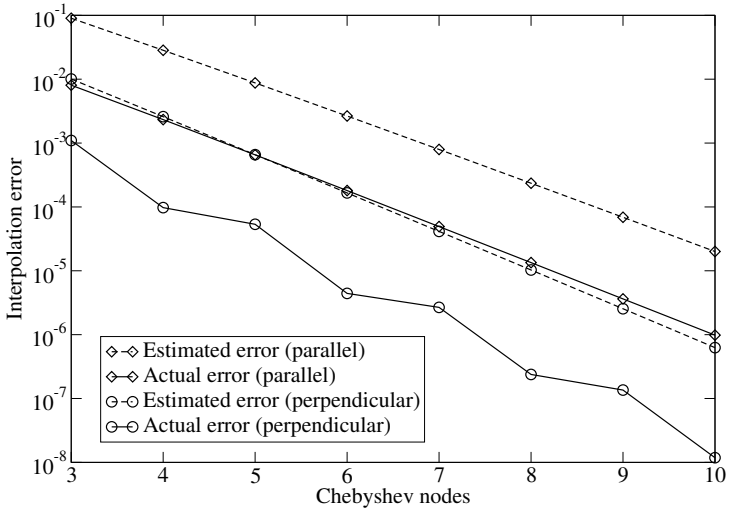


Figure 3: Interpolation error for $G(r) = \frac{1}{|c+r|}$

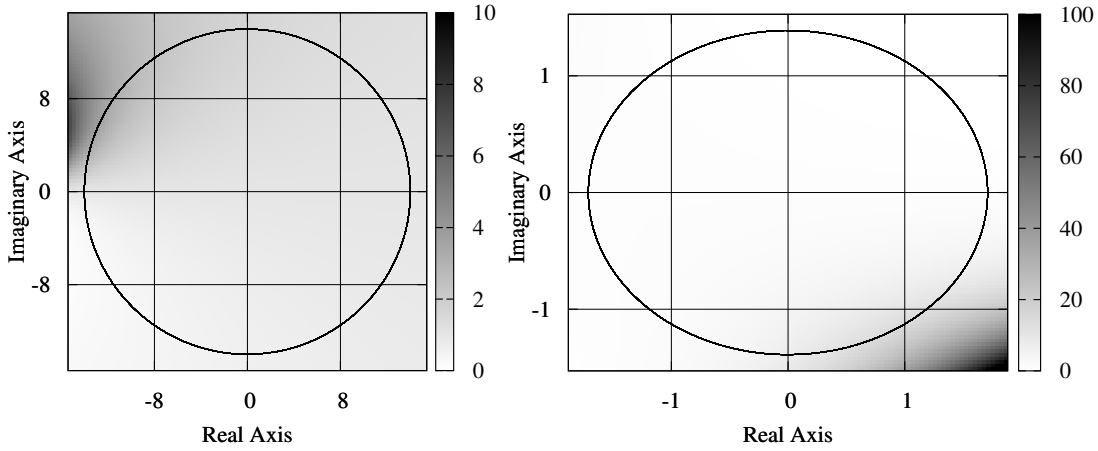
6.1.2 The oscillatory kernel $K^u(r) = \mathrm{e}^{\imath k(|c+r|-u\cdot(c+r))}$

The center of cluster Y depends on the wave number k. We choose it such that the minimal separation $O(k)$ and maximal angular deviation $O(1/k)$ from $u = [1,0]$, i.e., $c = c_y = [k,1]$, are satisfied. Again, we analyze two cases:

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

Figure 4 shows the modulus of the oscillatory kernel $K^u(r)$ 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 (26), the interpolation error must be smaller.

We can observe that in fig. 5. 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 eqn. (8): when k increases, the angle between c and u becomes smaller, and all the terms shown in eqn. (8) become negligible (they cancel out when c and u are exactly 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.



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

Figure 4: Modulus of $K^u(r) = \mathrm{e}^{ik(|c+r|-u\cdot(c+r))}$ with $r \in \mathbb{C}^2$ and for $k = 20$

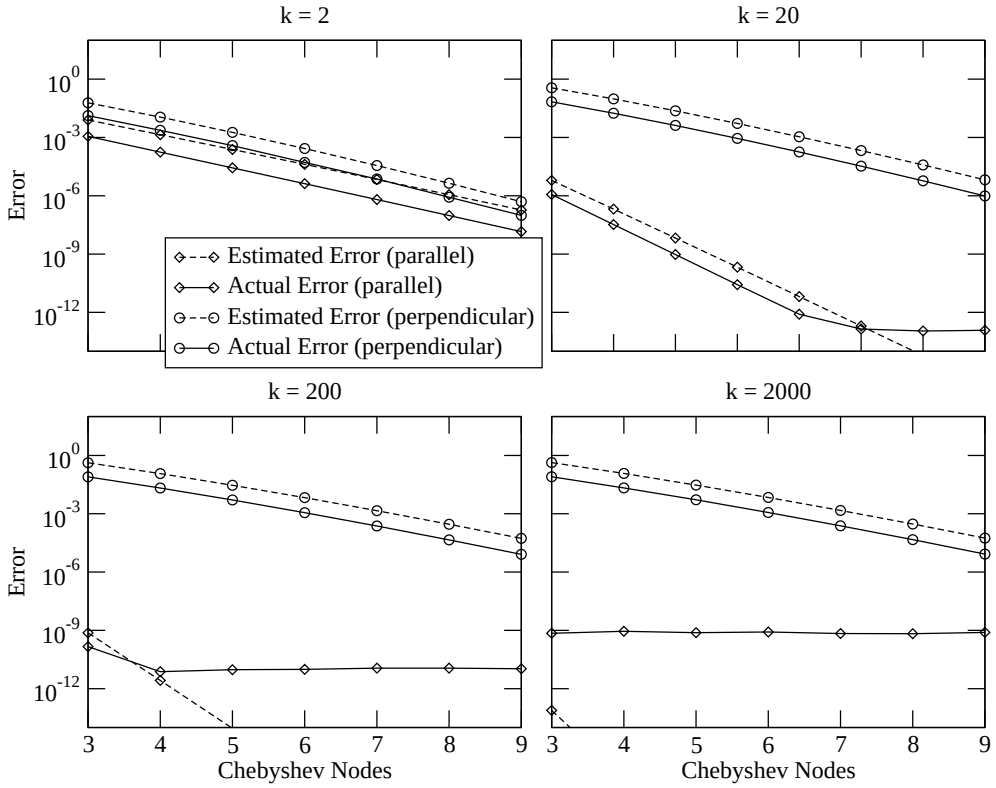


Figure 5: Interpolation error for $K^u(r) = \mathrm{e}^{ik(|c+r|-u\cdot(c+r))}$

6.2 Convergence of ACA and low-rank approximations of the M2L operators

We use the ACA for the compression of M2L operators. Here, we examine only results for the high frequency regime and the kernel

$$K(x,y) = \frac{\mathrm{e}^{\imath k|x-y|}}{|x-y|} \quad \text{for } x,y \in \mathbb{R}^2 \text{ or } \mathbb{R}^3.$$

The set-up is as follows: all clusters are of diameter $w = 1$ and contain 100 randomly distributed points. Cluster X is centered at the origin $c_x = [0,0]$. All directional cones $\{u_c\}_{c=1}^{C}$ of aperture $O(1/k)$ are filled with the set of clusters $\{Y_t\}_{t=1}^{T}$ such that $O(k) \leq |c_{y_t}| \leq O(4k)$ holds for all $t = 1,\ldots,T$. The upper limit is due to the minimal separation of the parent cluster whose diameter is $w = 2$. The size of an M2L operator is determined by the number of interacting clusters T per cone: for instance, for $T = 14$ it becomes a matrix of 100×1400 entries. Table 2 lists the set-up in terms of C and T depending on different wave numbers $k = 2$, 20, 200. Note that when $k = 200$, $C \sim 2\,10^6$ in three dimensions. This seems like a large number. To put things in perspective, in the traditional FMM [6, 7], the number of multipole coefficients grows like $O(k^2)$ and would therefore be of comparable magnitude.

		$d=2$		$d=3$	
k	aperture	C	T	C	T
2	$\pi/8$	16	14	96	44
20	$\pi/64$	128	157	6 144	1 066
200	$\pi/1\,024$	2 048	868	1 572 864	6 939

Table 2: The set-up for ACA convergence studies

Figure 6 shows the low rank κ obtained by means of the ACA for different choices of approximation accuracy ϵ and different wave numbers k. We observe that κ is bounded as k increases; this is consistent with what we have shown theoretically in section 2. Moreover, κ grows approximately like $O(\log(1/\epsilon))$, which is consistent with the rate of convergence of Chebyshev interpolation.

6.3 Particle distributions in $\mathbb{R}^2$

In this section we show that the overall complexity of our method is $O(N \log N)$ as presented in section 5. We scale the wave number k and keep the domain fixed. From this follows the relation $N = O(k^d)$. We choose the wave number k such that exactly one level has to be added as we solve the problem with the next higher k. Moreover, we analyze the complexity for different target accuracies (ℓ, ϵ_{ACA}).

At this stage, we have not developed an algorithm to optimize the parameters in the method. They involve: selecting the low-frequency/high-frequency threshold, the minimum distance between clusters in the interaction list, the cone aperture, the depth of the tree, ℓ and ϵ_{ACA}. As a result the running times are not optimal and can be improved by changing our choice of parameters. In addition, the performance results are very dependent upon the computer implementation of the algorithm.

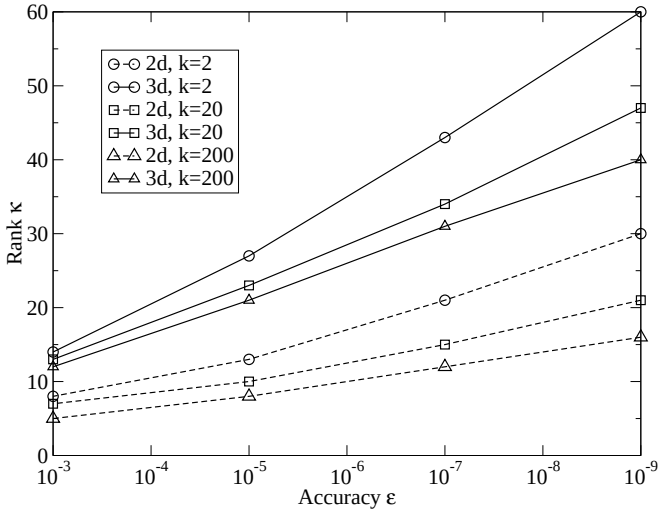


Figure 6: ACA in the high frequency regime

For the 2-dimensional examples we use randomly scattered particles on a unit square, such that the number of particles in a cluster of size one wave length remains constant. We use wave numbers $k = 128, 256, 512, 1024$. Results are presented in Tab. 3 and Fig. 7. All computations were performed on a single 2692 MHz CPU out of a 32 CPU node with 128 GB shared memory.

(ℓ, k, N)	t (sec)	ϵ_{L^2}
(5, 128, 4.00e+4)	22	1.11e-5
(5, 256, 1.60e+5)	111	1.30e-5
(5, 512, 6.40e+5)	585	3.15e-5
(5, 1024, 2.56e+6)	3 256	2.27e-6

Table 3: Results for $\mathbb{R}^2$ using a target accuracy $\epsilon_{\text{ACA}} = 10^{-5}$.

The running time and the relative error ϵ_{L^2} for an target accuracy $\epsilon_{\text{ACA}} = 10^{-5}$ are listed in Tab. 3. Both depend on ℓ. The relative error is defined as

$$\epsilon_{L^2} = \left(\frac{\sum_{i \in M_{\text{ref}}} |f_i - \bar{f}_i|^2}{\sum_{i \in M_{\text{ref}}} |f_i|^2} \right)^{1/2}.$$

M_{ref} is the number of particles in an arbitrary reference cluster at the leaf level, f_i and $\bar{f}_i$ are the approximately and exactly computed field values, respectively.

Fig. 7 compare the running time for different target accuracies (ℓ, ϵ_{ACA}). The predicted $O(N \log N)$ behavior appears. In almost all cases the actual error is as close to the target accuracy ϵ in ACA as presented in Tab. 3.

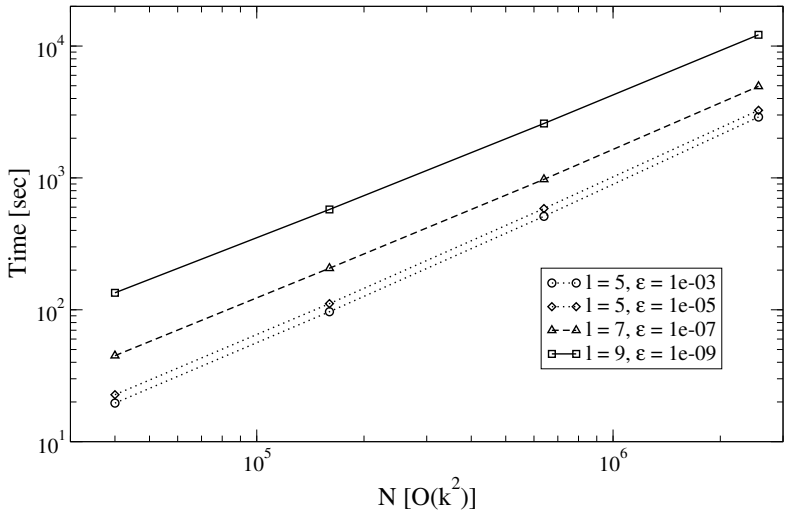


Figure 7: Computational time in $\mathbb{R}^2$; ϵ is the prescribed accuracy for ACA; ℓ is the Chebyshev interpolation order.

6.4 Particle distributions in $\mathbb{R}^3$

Here 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 [11]. In the following we analyze three different example geometries shown in Fig. 8. For completeness, we provide the short Gmsh scripts used to generate these geometries.

1. Sphere of diameter k as shown in Fig. 8a.

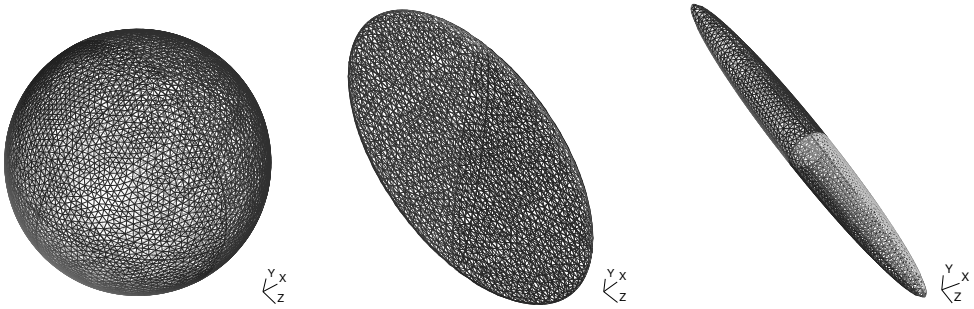
```
lc = 0.1; k = 128;
Point(1) = {0,0,0,lc};   Point(2) = {0,0,-k/2,lc};
Point(3) = {k/2,0,0,lc}; Point(4) = {0,0,k/2,lc};
Circle(1) = {2,1,3};     Circle(2) = {3,1,4};
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{1,2};}
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{3,6};}
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{9,12};}
Extrude{{0,0,1},{0,0,0},Pi/2}{Line{15,18};}}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

2. Oblate spheroid of diameter k in two directions and $k/10$ in the third direction as shown in Fig. 8b.

```
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. 8c.

```
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

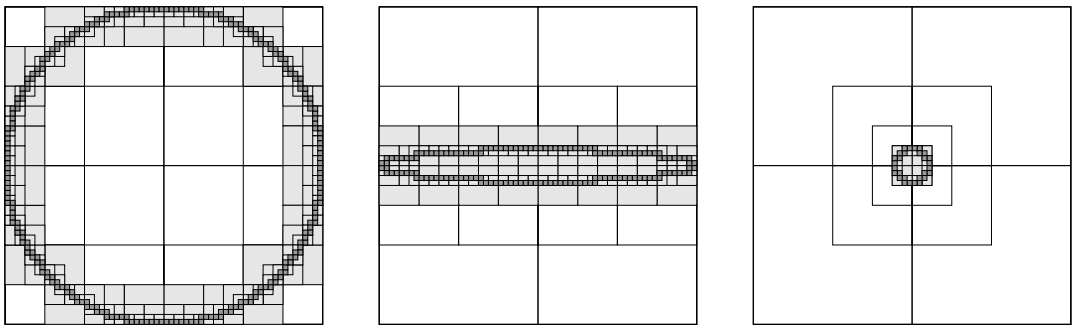
Figure 8: Example geometries

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. 9. Cross-sections through the uniform oct-trees of the surfaces from Fig. 8 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. 9 the depth of the oct-tree is chosen such that only leaf-level clusters are in the low-frequency regime.



(a) Surface 8a (b) Surface 8b (c) Surface 8c

Figure 9: Cross-sections in the z-direction, for uniform oct-trees, of the surface particle distributions from Fig. 8 with $k = 64$.

6.4.1 FMM parameters

At this stage, we have not developed a method to optimize the parameters in the method. As a result the running times are 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.

low-frequency/high-frequency 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 $\text{dist}_{\text{hf}} = \max(2w, kw^2)$.

Interpolation order and ACA accuracy The interpolation order ℓ and the ACA accuracy ϵ_{ACA} are strongly connected in terms of the final error of our method ϵ_{L^2}.

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-field 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 1 000 particles in a cluster having a size of 1 wavelength. Hence, a volume mesh or low accuracy require a deeper

tree. We chose the tree depth for *surface meshes* such that depending on the interpolation order ℓ, 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.

6.4.2 Surface particle distributions

We show computational timings of surface particle distributions for accuracies (4, 10^{-4}) and (7, 10^{-7}) in Table 4, 5 and 6. 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

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

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

Table 4: Timings for surface mesh of fig. 8a

6.4.3 Volume particle distributions

We show timing studies of volume particle distributions for accuracies (4, 10^{-4}) and (7, 10^{-7}) in Table 7, 8 and 9. Again, the shown timings do not include the precomputation time and we can see the almost linear growth by means of roc_t.

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

Table 5: Timings for surface mesh of fig. 8b

$(\ell, \epsilon_{\text{ACA}}, k, N)$	ϵ_{L^2}	t (sec)	roc_{sec}
(4,1e-4, 16,7.66e+3)	3.01e-4	0.5	
(4,1e-4, 32,2.97e+4)	3.25e-4	2	0.977
(4,1e-4, 64,1.19e+5)	1.73e-4	9	0.922
(4,1e-4,128,4.83e+5)	5.46e-4	36	1.010
(4,1e-4,256,1.94e+6)	1.28e-4	155	0.952
(7,1e-7, 16,7.66e+3)	9.30e-7	2	
(7,1e-7, 32,2.97e+4)	5.10e-7	7	1.081
(7,1e-7, 64,1.19e+5)	4.25e-7	32	0.913
(7,1e-7,128,4.83e+5)	9.83e-7	155	0.888
(7,1e-7,256,1.94e+6)	5.43e-7	682	0.938

Table 6: Timings for surface mesh of fig. 8c

Contrary to the surface mesh studies in Section 6.4.2 here we are not able to present results for $k = 64$, 128 and 256, respectively, due to limited memory. In Table 7 the computation for $k = 32$ is only possible for an accuracy $(4, 10^{-4})$, the same holds in Tab. 8 for $k = 64$. In Tab. 9 no computation for $k = 128$ is possible.

$(\ell, \epsilon_{\text{ACA}}, k, N)$	ϵ_{L^2}	t (sec)	roc_{sec}
(4,1e-4, 4,6.14e+4)	8.36e-4	9	
(4,1e-4, 8,3.36e+5)	9.58e-4	53	0.958
(4,1e-4,16,3.56e+6)	5.54e-3	552	1.007
(4,1e-4,32,4.26e+7)	3.87e-3	7302	0.961
(7,1e-7, 4,6.14e+4)	7.14e-7	45	
(7,1e-7, 8,3.36e+5)	3.10e-6	271	0.946
(7,1e-7,16,3.56e+6)	8.80e-6	3362	0.937

Table 7: Timings of volume meshes of fig. 8a

$(\ell, \epsilon_{\text{ACA}}, k, N)$	ϵ_{L^2}	t (sec)	roc_{sec}
(4,1e-4, 8,4.04e+4)	1.55e-3	5	
(4,1e-4,16,3.94e+5)	8.87e-4	52	0.972
(4,1e-4,32,2.21e+6)	1.46e-3	334	0.927
(4,1e-4,64,2.62e+7)	2.49e-3	3999	0.996
(7,1e-7, 8,4.04e+4)	6.38e-6	14	
(7,1e-7,16,3.94e+5)	6.07e-6	290	0.751
(7,1e-7,32,2.21e+6)	6.78e-6	1562	1.024

Table 8: Timings of volume meshes of fig. 8b

6.5 Additional numerical benchmarks in $\mathbb{R}^3$

6.5.1 Comparison with another FMM formulation

We compare our method with another FMM formulation that uses an analytical kernel expansion. The formulation is presented in [4]. We point out that there are many different multipole formulations and implementations that vary in accuracy, stability, and complexity. Therefore comparing different implementations is a difficult task. Each method has many parameters that require correct tuning to obtain the best performance. We chose the method of [4] since the authors are familiar with this implementation. We note that there are many differences between this work and [4]. In particular [4] only applies to kernels of the form e^{ikr}/r. In addition it suffers from a well-known stability problem (common to this class of methods) in the low frequency regime as reported in [8] for example. In contrast, this work applies to a wider class of kernels and arbitrary accuracies can be achieved (essentially down to machine precision). At this stage

$(\ell, \epsilon_{\text{ACA}}, k, N)$	ϵ_{L^2}	t (sec)	roc_{sec}
(4,1e-4,16,2.69e+4)	4.32e-4	3	
(4,1e-4,32,2.02e+5)	1.19e-3	29	0.888
(4,1e-4,64,1.56e+6)	2.26e-4	252	0.945
(7,1e-7,16,2.69e+4)	1.10e-6	13	
(7,1e-7,32,2.02e+5)	2.02e-6	127	0.884
(7,1e-7,64,1.56e+6)	1.12e-6	1125	0.937

Table 9: Timings of volume meshes of fig. 8c

in our work, the directional method is not fully optimized and in particular we currently do not use a precise method to optimize the parameters in the directional approach. As a result the timings for the directional algorithm are sub-optimal. Nevertheless these results provide a general sense of the performance of the method.

It is worth noting that for the surface particle distribution from Fig. 8c the new method is favored since fewer cones are required to calculate all the interactions. In contrast the plane wave FMM has an unfavorable scaling (see Tab. 1 Case 3). In general however the plane wave FMM can be expected to be faster than the directional method, since it uses analytical expansions to approximate the kernel and fast Fourier transforms for the interpolation and anterpolation. As explained before, tuning is required to optimize the directional approach. As a result we cannot provide a definitive answer to the question of which method is the fastest. Note that the highest accuracy we could achieve with the plane wave FMM was $\sim 10^{-7}$. On the contrary, the directional FMM can achieve accuracies up to 10^{-12} as presented in Fig. 11.

Tab. 10 presents the timing comparison. For 8a, the plane wave FMM (pFMM) is faster by a factor 2–3. For 8c, it is slower by about 15–30%. It is noteworthy to observe that the running time of the directional FMM (dFMM) is not affected by the geometry of the object whereas pFMM is (compare numbers from the upper table and the lower table).

6.5.2 Direct matrix-vector product

Figure 10 shows a comparison of the direct matrix-vector product and our method in terms of computational time. The results are obtained from surface meshes from Fig. 8 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. 8a with $k = 4$ and an accuracy $(7, 10^{-7})$. Our method outperforms the direct method in all other cases.

6.5.3 Overall error convergence and timings

Figure 11 shows the convergence of the relative error ϵ_{L^2} for interpolation order up to $\ell = 13$ and accuracies up to $\epsilon_{\text{ACA}} = 10^{-13}$. Both are key parameters for the accuracy of the method. We use the surface mesh from Fig. 8c with a wave number $k = 128$ (483,389 particles). We chose the tree depth as described in Section 6.4.1. 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

ϵ_{L^2}	$\sim 10^{-4}$		$\sim 10^{-7}$	
k	dFMM	pFMM	dFMM	pFMM
Surface from Fig. 8a				
8	2	2	6	4
16	8	6	29	15
32	30	16	143	62
64	139	67	632	252
Surface from Fig. 8c				
32	2	4	7	5
64	9	13	32	55
128	36	52	155	215
256	155	206	682	778

Table 10: Comparison of the directional FMM with a variant of the plane wave FMM [4] in terms of computational time in seconds. pFMM: plane wave FMM; dFMM: directional FMM.

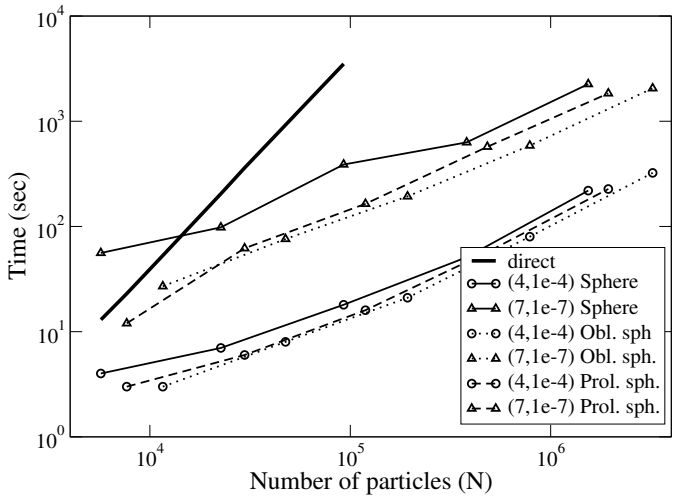


Figure 10: Timings for the direct method and the dFMM. In both cases the timings include the setup of the matrix as well as the matrix-vector product itself. The surface meshes originate from fig. 8.

we fix ϵ_{ACA}, we cannot get beyond a certain accuracy, and vice versa. For example, with an accuracy of $\epsilon_{L^2} \sim 10^{-4}$, there is no point in choosing ℓ greater than 5. If we need an accuracy of $\epsilon_{L^2} \sim 10^{-6}$ we need to use at least $\ell = 7$ and $\epsilon_{\text{ACA}} = 10^{-7}$. Figure 11 shows that accuracies up to $\epsilon_{L^2} < 10^{-11}$ can be achieved. Figure 12 shows timings for the accuracies ϵ_{L^2} presented in

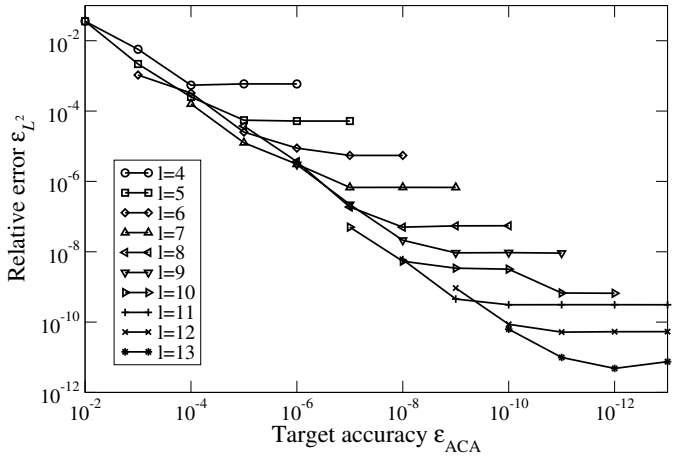


Figure 11: Relative error ϵ_{L^2} for the surface mesh from Fig. 8c with wave number $k = 128$ (483,389 particles); ϵ_{ACA} is the accuracy for ACA; ℓ is the Chebyshev interpolation order.

Fig. 11.

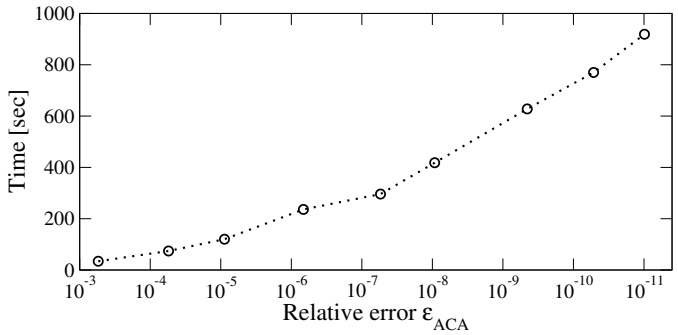


Figure 12: Timings for the matrix-vector product for accuracies up to 10^{-11} for the surface mesh from Fig. 8c and $k = 128$.

6.5.4 ACA plus QR-decomposition vs. SVD

In Tab. 11 we compare the low rank ($\kappa_{\text{ACA}}/\kappa_{\text{SVD}}$) and the precomputation time ($t_{\text{SVD}}/t_{\text{SVD}}$) of ACA plus QR-decomposition (this paper) against the SVD approach (see [10]). We prescribe the accuracy $\epsilon_{\text{ACA}} = \epsilon_{\text{SVD}} = 10^{-4}$. Shown are results from the surface mesh from Fig. 8c. 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 lower (N vs. N^3 for a matrix of size N).

k	κ_{ACA}	κ_{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 11: Low rank κ and timings t for ACA+QR and SVD.

Table 11 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 expansions fastest. The decay behavior of the singular values can be improved by decreasing the cone aperture or increasing the admissible distance. As already mentioned in Section 6.4.1 we have not yet fully optimized the parameter setting of our method.

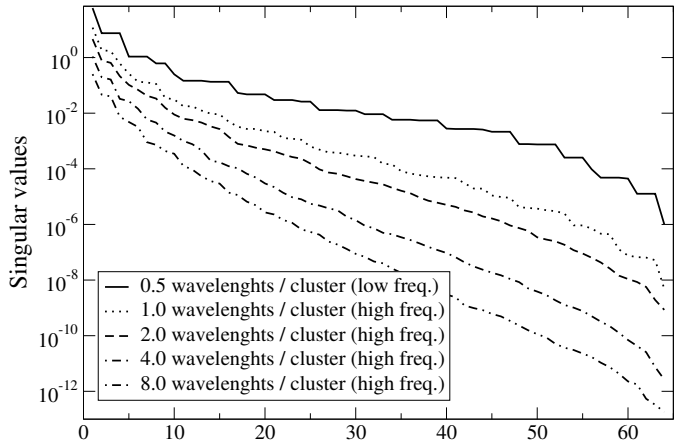


Figure 13: Singular values for the surface mesh of Fig. 8c with $k = 256$ and $\ell = 4$. The index of the singular values is shown on the x-axis. Leaf clusters have a size of $1/2$ wave length.

6.6 Particle distributions on irregular surfaces

We show two numerical studies of particle distributions on irregular surfaces in $\mathbb{R}^3$. The first one is the head of a dinosaur skeleton shown in Fig. 14 and the second one are insect legs shown

in Fig. 16b. We downloaded both meshes from [20]. Figures 15 and 16a show cross-sections through the uniform oct-tree of the dinosaur skeleton and the insect legs, respectively. Again, 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 no expansions are needed. The depth of the oct-tree is chosen such that only leaf-clusters are in the low-frequency regime.

In the previous examples, we were keeping the number of particles per wave length constant as we varied the wave number k. Here, as refining the mesh is practically difficult, we simply varied k, while keeping the depth of the oct-tree constant. In principle the number of levels should be increased. However since we are not refining the mesh, this would lead to leaf clusters containing very few particles. Consequently we decided to keep the number of levels constant, thereby ensuring that the number of particles in the leaf clusters does not vary. Tab. 12 and Tab. 13 show that the target accuracies can be achieved, as predicted.

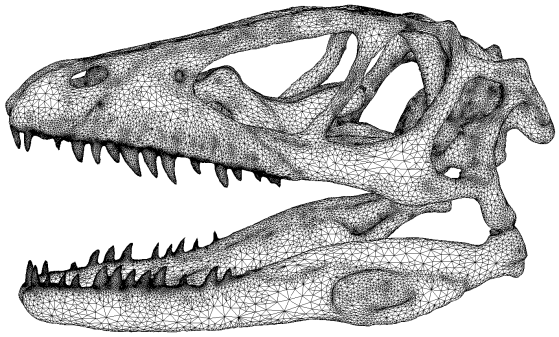
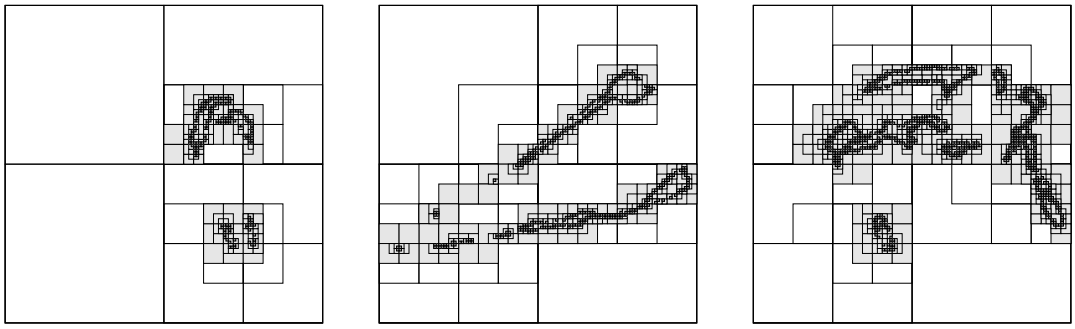


Figure 14: Surface mesh of a dinosaur skeleton consisting of 122,589 vertices

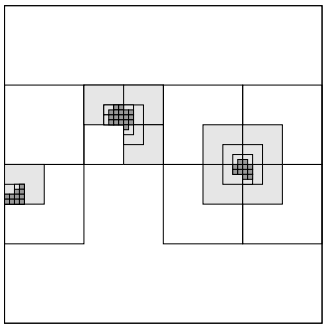


(a) x_1-direction (b) x_2-direction (c) x_3-direction

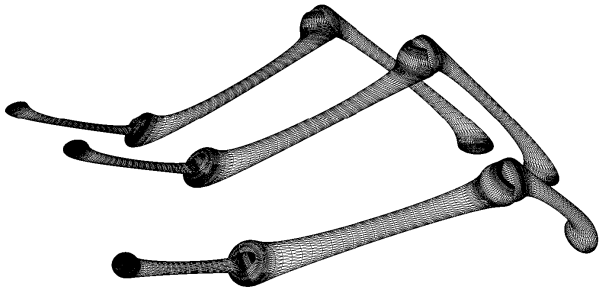
Figure 15: Cross-sections through the uniform oct-tree of Fig. 14

k	$(\ell, \epsilon_{\mathrm{ACA}})$ (4, 1e-4)	(6, 1e-6)	(8, 1e-8)	(10, 1e-10)
10	1.99e-3	7.35e-6	9.45e-8	1.45e-9
20	3.81e-4	4.69e-6	1.94e-7	1.88e-9
40	2.03e-4	4.59e-6	3.61e-8	7.66e-10
80	1.83e-3	2.84e-5	1.25e-7	6.07e-9

Table 12: Dinosaur skeleton from Fig. 14 with 122,589 particles. The depth of the oct-tree is kept at 7 levels for all k.



(a) Cross-section through the uniform oct-tree in the x_2-direction



(b) Surface mesh consisting of 63,029 vertices

Figure 16: Insect legs

k	$(\ell, \epsilon_{\mathrm{ACA}})$ (4, 1e-4)	(6, 1e-6)	(8, 1e-8)	(10, 1e-10)
7	4.56e-4	3.14e-5	2.16e-7	3.94e-9
14	1.71e-3	4.99e-5	2.63e-7	6.41e-9
28	1.17e-3	1.01e-5	1.15e-7	4.03e-10
56	3.04e-4	1.40e-5	2.42e-8	2.48e-10

Table 13: Results for insect legs from Fig. 16b with 63,029 particles. The depth of the oct-tree is kept at 6 levels for all k.

7 Conclusion

We have presented a fast directional summation method for oscillatory kernel functions. The directional idea originates from the publications of [3] and [9]. Our method is based on the interpolation of the kernel function via Chebyshev polynomials. An advantage is the relatively simple error analysis based on the behavior (growth) of the kernel function in the complex plane. Our approach is also convenient because we perform similar operations in the low and high frequency regimes, making the computer implementation less involved. The main changes between low and high frequency regimes are 1) the use of different interpolation operators, and 2) the use of directional multipole and local coefficients in the high frequency regime.

One advantage of the formulation we propose for the interpolation scheme is that it allows using the original kernel without any modification, i.e., the kernel K^u does not appear explicitly. This simplifies the computer programming of the method. This becomes more evident if we consider more complex kernels as in elastodynamics and poroelasticity. However, in these cases another complication arises. Consider the elastodynamic kernel $K^{\mathrm{ed}}(\chi(x,y))$ in $\mathbb{R}^3$ with

$$\chi(x,y) = \frac{1}{4\pi}\frac{1}{k_1^2 - k_2^2}\frac{\mathsf{e}^{-k_1|x-y|} - \mathsf{e}^{-k_2|x-y|}}{|x-y|}$$

with $k_{1,2} = s/c_{1,2}$, $s \in \mathbb{C}$ and the two elastic wave velocities c_1 and c_2. In poroelasticity a third wave of velocity c_3 appears. We see that each wave is connected to one oscillatory term $\mathsf{e}^{k_i|x-y|}$. 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), and finally combine both contributions.

By means of the ACA and the further compression based on the QR decomposition we worked out an M2L operation with a lower computational cost. The main advantage of ACA is that it requires as input only a user defined routine to evaluate the kernel function $K(x,y)$, i.e., it can be applied in a black-box manner.

In section 6 we verified the estimated interpolation error, the convergence of ACA, the overall convergence of the method, and the computational complexity of the proposed algorithm by means of numerical benchmarks. A comparison was proposed with the FMM of [4]. Very few papers compare FMM implementations, because of the difficulty of carrying out such comparisons and the need to carefully tune the many parameters in each method. In our comparison, we did not optimize the choice of parameters in the directional FMM.

References

[1] M. Bebendorf. Approximation of boundary element matrices. *Numerische Mathematik*, 86(4):565–589, 2000.

[2] 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.

[3] A. Brandt. Multilevel computations of integral transforms and particle interactions with oscillatory kernels. *Computer Physics Communications*, 65(1-3):24 – 38, 1991.

[4] C. Cecka and E. Darve. Fourier based fast multipole method for the helmholtz equation. *SIAM Journal on Numerical Analysis*, 2010.

[5] 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.

[6] E. Darve. The fast multipole method i: Error analysis and asymptotic complexity. *SIAM Journal on Numerical Analysis*, 38(1):98–128, 2000.

[7] E. Darve. The fast multipole method: numerical implementation. *Journal of Computational Physics*, 160(1):195–240, 2000.

[8] E. Darve and P. Havé. Efficient fast multipole method for low-frequency scattering. *Journal of Computational Physics*, 197(1):341–363, 2004.

[9] B. Engquist and L. Ying. Fast directional multilevel algorithms for oscillatory kernels. *SIAM Journal on Scientific Computing*, 29(4):1710–1737, 2007.

[10] W. Fong and E. Darve. The black-box fast multipole method. *Journal of Computational Physics*, 228(23):8712 – 8725, 2009.

[11] 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. URL `http://doi.wiley.com/10.1002/nme.2579`.

[12] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[13] 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.

[14] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62:89–108, 1999.

[15] 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.

[16] 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.

[17] J.C. Mason and D.C. Handscomb. *Chebyshev polynomials*. Chapman & Hall/CRC, 2003. ISBN 9780849303555.

[18] 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.

[19] R. Rokhlin. Diagonal forms of translation operators for the helmholtz equation in three dimensions. *Applied and Computational Harmonic Analysis*, 1:82 – 93, 1993.

[20] E. Saltel. 3D Meshes Research Database INRIA GAMMA Group, 2011. URL `http://www-roc.inria.fr/gamma/gamma.php`.

[21] G. Schmidlin, Ch. Lage, and Ch. Schwab. Rapid solution of first kind boundary integral equations in 3. *Engineering Analysis with Boundary Elements*, 27(5):469 – 490, 2003.