A Directional Equispaced interpolation-based Fast Multipole Method for oscillatory kernels

Igor Chollet, Xavier Claeys, Pierre Fortin, Laura Grigori

A Directional Equispaced interpolation-based Fast Multipole Method for oscillatory kernels

Igor Chollet* Xavier Claeys† Pierre Fortin ‡ Laura Grigori §

February 2023

Abstract

Fast Multipole Methods (FMMs) based on the oscillatory Helmholtz kernel can reduce the cost of solving N-body problems arising from Boundary Integral Equations (BIEs) in acoustic or electromagnetics. However, their cost strongly increases in the high-frequency regime. This paper introduces a new directional FMM for oscillatory kernels (*defmm* - directional equispaced interpolation-based fmm), whose precomputation and application are FFT-accelerated due to polynomial interpolations on equispaced grids. We demonstrate the consistency of our FFT approach, and show how symmetries can be exploited in the Fourier domain. We also describe the algorithmic design of *defmm*, well-suited for the BIE non-uniform particle distributions, and present performance optimizations on one CPU core. Finally, we exhibit important performance gains on all test cases for *defmm* over a state-of-the-art FMM library for oscillatory kernels.

1 Introduction

Considering two point clouds $X, Y \subset \mathbb{R}^d$ with cardinal $N \in \mathbb{N}^*$, $d \in \mathbb{N}^*$ and $q \in \mathbb{C}[Y]$ (named *charges*), where $\mathbb{C}[Y]$ denotes the set of application from Y to $\mathbb{C}$, we are interested in the fast computation of $p \in \mathbb{C}[X]$ (referred to as *potentials*) such that

$$p(\mathbf{x}) := \sum_{\mathbf{y} \in Y} G(\mathbf{x}, \mathbf{y}) q(\mathbf{y}), \qquad \forall\, \mathbf{x} \in X, \tag{1}$$

where $G : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$. Such an *N-body problem* appears in the numerical solving of Boundary Integral Equations (BIEs). We are especially concerned by the oscillatory *Helmholtz kernel* involved in BIEs applied to acoustic or electromagnetics

$$G(\mathbf{x}, \mathbf{y}) := \frac{e^{i\kappa|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|},$$

where $\kappa \in \mathbb{R}^+$ is named the *wavenumber*, i denotes the complex number and $|\cdot|$ refers to the Euclidian norm. Computing p using hierarchical methods can achieve an $\mathcal{O}(N \log N)$ complexity

*Institut des Sciences du Calcul et des Données (ISCD), INRIA Alpines (igor.chollet@inria.fr).

†Laboratoire Jacques-Louis Lions, Sorbonne Université, Inria équipe ALPINES, F-75005 (claeys@ann.jussieu.fr).

‡Sorbonne Université, CNRS, LIP6, F-75005 Paris, France; Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France (pierre.fortin@univ-lille.fr)

§Laboratoire Jacques-Louis Lions, Sorbonne Université, Inria équipe ALPINES, F-75005 (laura.grigori@inria.fr)

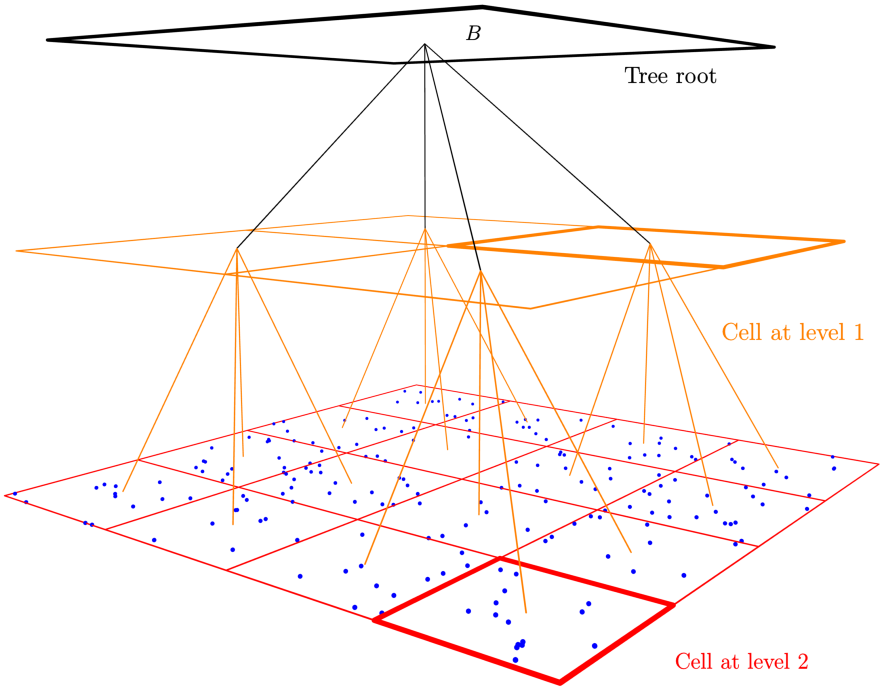


Figure 1: 2^d-tree ($d = 2$) representation of B with particles colored in blue.

on large surface *particle distributions* (i.e. point clouds), that are the distributions generated when discretizing BIEs. However, the cost of the hierarchical methods can still be a bottleneck in the *high-frequency regime*, i.e. when $\kappa D \gg 1$, D denoting the side length of the smallest box encompassing X and Y. We aim here at mitigating this cost in the high-frequency regime for the non-uniform particle distributions that are typical of BIE problems.

1.1 Related work

The direct evaluation of the N-body problem (1) has a complexity $\mathcal{O}(N^2)$. This complexity can be reduced using a hierarchical method, such as the Fast Multipole Method (FMM) [11, 21]. Thanks to a hierarchical decomposition, such as a 2^d-tree[1] representation of the (cubical) computational domain B, and to a multipole acceptance criterion, the computation of (1) is indeed divided into two parts: a near field one which is computed directly and a far field one which is approximated through multipole and local expansions. Nodes of such a 2^d-tree are named *cells* and correspond to cubical subdomains of B whose radii depend on the tree level (see Figure 1).

There mainly exist two approaches for FMMs dealing with the high-frequency Helmholtz kernel. First, the kernel-explicit methods consist in an explicit diagonalization of large far-field matrix blocks using analytic kernel expansions [9,11], considering that the N-body problem in equation 1 is interpreted as a matrix-vector product. These methods perform differently depending on the particle distribution, with complexities ranging from $\mathcal{O}(N)$ on quasi-uniform distributions to $\mathcal{O}(N\,log N)$ or $\mathcal{O}(N^{3/2})$ on homogeneous surface distributions (see [9]), and possibly even to $\mathcal{O}(N^2)$ on highly non-

[1] Binary tree (d=1), quadtree (d=2) or octree (d=3).

uniform distributions (see [25]). Second, the kernel-independent methods (see for instance [16, 32]) allow to use the same formalism, hence the same implementation, for a given class of kernels. Based on a specific definition of the *well-separateness* (i.e. the subsets of $B \times B$ on which the far-field approximations can be applied), the *directional* kernel-independent approaches exploit the low-rank property of particular blocks of a modified kernel [6, 15, 25]. As opposed to the kernel-explicit methods, the kernel-independent directional approaches allow to derive fast algorithms with complexity $\mathcal{O}(N \log N)$ regardless of the particle distribution. According to [25], this results in superior performance for a directional FMM on highly non-uniform distributions, a kernel-explicit FMM being better-suited on quasi-uniform particle distributions and on homogeneous surface ones.

The directional FMM algorithms described in the literature exploit low-rank approximations to compress matrices representing the far-field contributions [15, 24, 25]. However, some highly efficient kernel-independent methods for low-frequency and translation-invariant kernels use Fast Fourier Transforms (FFTs) to efficiently process these matrices in diagonal form [30]. These FFT-based methods exploit *equispaced grids* (i.e. cartesian grids or tensorization of the same uniform sampling d times). In FMM formulations based on polynomial interpolation, the use of FFT techniques was shown to be more efficient than low-rank approximations in the low-frequency regime [5], especially for high approximation orders. To our knowledge, directional FFT-based FMMs for the high-frequency regime have not been investigated in the past. Moreover, in such polynomial interpolation-based FMMs, the matrices associated to some FMM operators have to be precomputed, but their number drastically increases in the high-frequency regime (see [24]), which strongly increases the precomputation cost. FFT techniques can also help to reduce such precomputation cost.

1.2 Contributions

In this paper, we aim at efficiently solving BIE N-body problems with oscillatory kernels in the high-frequency regime. We thus present the first directional interpolation-based FMM taking advantage of polynomial interpolation on equispaced grids to allow fast FFT-based evaluations and precomputations of the far-field matrices. Our FMM retain the overall $\mathcal{O}(N log N)$ complexity of directional kernel-independent FMMs in the high-frequency regime, and the usual linear FMM complexity in the low-frequency regime: we thus rely on the FFTs to lower the computation cost in practice in the high-frequency regime. A consistency proof for the polynomial Lagrange interpolation on such grids and on well-separated sets when dealing with asymptotically smooth kernels is provided. This gives a rigorous explanation of the practical convergence of our approximation process despite the well-known stability issue of such polynomial interpolation. We also show how to extend the use of some symmetries in the Fourier domain in order to minimize the number of precomputed matrices. FMMs can rely on different 2^d-tree traversals: we adapt here a specific one (namely the dual tree traversal) to directional FMMs and show its relevance, along with a suitable 2^d-tree data structure, for the non-uniform particle distributions typical of BIE problems. We provide a new, publicly available[2], C++ library implementing our FMM, named as *defmm* (directional equispaced interpolation-based fmm) and highly optimized on one CPU core. We detail the vectorization process of our near-field direct computation using high-level programming, and we show how to efficiently process the numerous and small FFTs required in our method. We also improve the performance of the BLAS-based operators in the multipole and local expansion translations. Finally, we detail a comparison with a state-of-the-art directional polynomial interpolation-based FMM library (namely *dfmm* [24, 25]), exhibiting important performance gains for *defmm* in all the test cases.

The paper is organized as follows. In section 2, we first recall the mathematical bases of the FMMs involved in this paper. We then provide in section 3 a consistency proof of our interpolation

[2] At: `https://github.com/IChollet/defmm`

process on equispaced grids. We detail in section 4 the *defmm* algorithmic design and we show how to extend and exploit the tree symmetries in the Fourier domain. In section 5, we present various HPC optimizations on one CPU core for *defmm*, and finally in section 6 we provide numerical results, including a detailed performance comparison.

2 Presentation of Fast Multipole Methods

We focus in this article on the FMM formulation using polynomial interpolation and its variants [16, 25, 30], which are here briefly recalled.

2.1 Directional FMMs

Suppose that the kernel G can be factorized into

$$G(\mathbf{x},\mathbf{y}) = e^{i\kappa|\mathbf{x}-\mathbf{y}|}K(\mathbf{x},\mathbf{y}), \tag{2}$$

where K is an asymptotically smooth non-oscillatory kernel. Directional approaches rely on the *directional parabolic separation condition* (DPSC) [15], imposing conditions on well-separated subsets of $B \times B$ (i.e. pairs of cells in practice) such that the term G_u in the expression

$$G(\mathbf{x},\mathbf{y}) = e^{i\kappa\langle \mathbf{x},u\rangle} \underbrace{\left(e^{i\kappa\langle \mathbf{x}-\mathbf{y}, \frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|}-u\rangle}K(\mathbf{x},\mathbf{y})\right)}_{=:G_u(\mathbf{x},\mathbf{y})} e^{-i\kappa\langle \mathbf{y},u\rangle} \tag{3}$$

does not oscillate, where $u \in \mathbb{S}^2$ (referred to as a *direction*) is a well-chosen approximation of $(\mathbf{x}-\mathbf{y})/|\mathbf{x}-\mathbf{y}|$ on the unit sphere. Since the only term in equation (3) depending on both $\mathbf{x}$ and $\mathbf{y}$ is G_u, fast summation methods can be derived. The DPSC is verified when $\mathbf{x} \in t$, $\mathbf{y} \in s$, where t, s are two cells such that

$$\kappa w^2 \leq \eta\, dist(t,s) \text{ and } t \subset C(s), \tag{4}$$

with $dist(t,s) := min_{\mathbf{x}\in t,\mathbf{y}\in s}|\mathbf{x}-\mathbf{y}|$ the distance between t and s, w the radius[3] of s (assuming that $w > 1$) and $C(s)$ a cone directed by u with apex in the center of s and aperture $\frac{\mu}{\kappa w}$ [15], η, μ being two strictly positive constants. Such pairs (t,s) are said to be *well-separated* in the high-frequency regime.

Because the cone aperture in the inequality (4) decreases as the cell radius w increases, more wedges are needed to compute the far field for cells close to the 2^d-tree root than for deeper cells. Since the kernel approximations depend on the cone directions, the number of approximations required for each target cell increases from the 2^d-tree leaves to the root. To lower the far field computation cost, the set of directions at each tree level is chosen so that a nested property is verified [7, 15]. This leads to a *direction tree*: each direction at a given 2^d-tree level E is the son of a unique direction at the 2^d-tree level $E+1$.

2.2 Interpolation-based FMMs

The polynomial interpolation techniques for hierarchical methods [8, 16, 20] rely on approximations of G using Lagrange interpolation. We refer to [19] for a description of the multivariate interpolation methods and to the Lagrange interpolation problem. In the following formula, we indicate in gray the terms only appearing when combining the polynomial interpolation techniques for FMM with

[3] i.e. the radius of the smallest ball containing the cell.

the directional approach [24, 25]. Let $\Xi_s = \{\mathbf{y}_1, ..., \mathbf{y}_{\#\Xi_s}\}$ and $\Xi_t = \{\mathbf{x}_1, ..., \mathbf{x}_{\#\Xi_t}\}$ (where $\#\Xi_m$, $m \in \{t, s\}$, denotes the cardinal of Ξ_m) be two interpolation grids in the cells s and t respectively, where the pair (t, s) is well-separated, we have:

$$\begin{aligned} G(\mathbf{x}, \mathbf{y}) &\approx e^{i\kappa\langle \mathbf{x}, u\rangle} \sum_k S_k[t](\mathbf{x}) \sum_l G_u(\mathbf{x}_k, \mathbf{y}_l) S_l[s](\mathbf{y}) e^{-i\kappa\langle \mathbf{y}, u\rangle} \\ &\approx \sum_k \underbrace{\left(e^{i\kappa\langle \mathbf{x}, u\rangle} S_k[t](\mathbf{x}) e^{-i\kappa\langle \mathbf{x}_k, u\rangle}\right)}_{=:S_k^u[t](\mathbf{x})} \sum_l G(\mathbf{x}_k, \mathbf{y}_l) \underbrace{\left(e^{i\kappa\langle \mathbf{y}_l, u\rangle} S_l[s](\mathbf{y}) e^{-i\kappa\langle \mathbf{y}, u\rangle}\right)}_{=:S_l^u[s](\mathbf{y})}, \end{aligned} \tag{5}$$

the polynomials $S_l[s]$ and $S_k[t]$ verifying for any $\mathbf{y} \in s$ and any $\mathbf{x} \in t$

$$S_l[s](\mathbf{y}) \begin{cases} = 1 & \text{if } \mathbf{y} = \mathbf{y}_l \\ = 0 & \text{if } \mathbf{y} \in \Xi_s, \mathbf{y} \neq \mathbf{y}_l, \\ \in \mathbb{R} & \text{otherwise} \end{cases} \qquad S_k[t](\mathbf{x}) \begin{cases} = 1 & \text{if } \mathbf{x} = \mathbf{x}_k \\ = 0 & \text{if } \mathbf{x} \in \Xi_t, \mathbf{x} \neq \mathbf{x}_k. \\ \in \mathbb{R} & \text{otherwise} \end{cases}$$

Suppose that $\mathbf{x} \in t$, t being a target cell. For any source cell s well-separated from t, we define

$$\begin{aligned} p_{t,s}(\mathbf{x}) &:= \sum_{\mathbf{y} \in s \cap Y} G(\mathbf{x}, \mathbf{y}) q(\mathbf{y}) \\ &\approx \sum_k S_k^u[t](\mathbf{x}) \sum_l G(\mathbf{x}_k, \mathbf{y}_l) \sum_{\mathbf{y} \in s \cap Y} S_l^u[s](\mathbf{y}) q(\mathbf{y}), \end{aligned}$$

where the second line is obtained thanks to the approximation (5). This interpolation process can be repeated on $t' \in Sons(t)$ and $s' \in Sons(s)$ recursively, leading in the end to a multilevel algorithm whose operators, named as in [1, 16], are the following:

P2M (particles-to-multipole):	$\mathcal{M}_{s'}^u(\mathbf{y}'_r) := \sum_{\mathbf{y} \in s \cap Y} S_r^u[s'](\mathbf{y}) q(\mathbf{y}),$	$\forall \mathbf{y}'_r \in \Xi_{s'};$
M2M (multipole-to-multipole):	$\mathcal{M}_s^v(\mathbf{y}_l) := \sum_r S_l^v[s](\mathbf{y}'_r) \mathcal{M}_{s'}^u(\mathbf{y}'_r),$	$\forall \mathbf{y}_l \in \Xi_s;$
M2L (multipole-to-local):	$\mathcal{L}_t^v(\mathbf{x}_k) := \sum_l G(\mathbf{x}_k, \mathbf{y}_l) \mathcal{M}_s^v(\mathbf{y}_l),$	$\forall \mathbf{x}_k \in \Xi_t;$
L2L (local-to-local):	$\mathcal{L}_{t'}^u(\mathbf{x}'_h) := \sum_k S_k^v[t](\mathbf{x}'_h) \mathcal{L}_{t'}^v(\mathbf{x}_k),$	$\forall \mathbf{x}'_h \in \Xi_{t'};$
L2P (local-to-particles):	$p_{t,s}(\mathbf{x}) \approx \sum_h S_h^u[t'](\mathbf{x}) \mathcal{L}_{t'}^u(\mathbf{x}'_h),$	$\forall \mathbf{x} \in X \cap t;$

where: $\Xi_{s'} = \{\mathbf{y}'_1, ..., \mathbf{y}'_{\#\Xi_{s'}}\}$ and $\Xi_{t'} = \{\mathbf{x}'_1, ..., \mathbf{x}'_{\#\Xi_{t'}}\}$ are the interpolation grids in s' and t' respectively; v is a son of u in the direction tree such that u and v are the best approximations of $(ctr(t) - ctr(s))/|ctr(t) - ctr(s)|$; $ctr(c)$ denoting the center of the cell c. The direct evaluation of $p_{t,s}$ for all $\mathbf{x} \in X \cap t$ corresponds to the P2P (particles-to-particles) operator between t and s. The result of a P2M or M2M operation is a *multipole* expansion and the result of a M2L or L2L operation is a *local* expansion. The set of cells s such that $p_{t,s}$ is approximated (i.e. not directly evaluated through a P2P operator) is named the *interaction list* of t.

All the FMM operators we defined can be interpreted as matrix-vector products, where the vectors are local or multipole expansions. Hence, we shall for example use in the remainder *M2L matrices* to refer to the matrices involved in the M2L matrix-vector product (and similarly for the other operators).

2.3 Equispaced grids

When dealing with multivariate polynomial interpolations, the most natural approach relies on tensorizations of 1D interpolation grids in a cuboid [8, 16, 20]. Denoting $B_\star = [0,1]^d$ the reference unit box, any cell c writes as

$$c = \boldsymbol{\alpha}_c + \beta_c \cdot B_\star \tag{6}$$

where $\beta_c = \mathrm{diam}(B)/2^L$ with $L = \mathrm{level}(c)$, and $\boldsymbol{\alpha}_c \in \mathbb{R}^d$. We focus here on equispaced grids, which allow to take advantage of the translation invariance of the kernel, i.e. $G((\mathbf{x}+\mathbf{z})-(\mathbf{y}+\mathbf{z})) = G(\mathbf{x}-\mathbf{y})$. The interpolation grid $\Xi_c \subset \mathbb{R}^d$ in cell c is obtained from a reference grid $\Xi_\star \subset \mathbb{R}^d$ of $B_\star$ through (6)

$$\begin{aligned} &\Xi_c = \boldsymbol{\alpha}_c + \beta_c \cdot \Xi_\star \\ &\text{where} \quad \Xi_\star := \mathbb{L} \times \cdots \times \mathbb{L} \\ &\qquad\qquad \mathbb{L} := \{\ell/L,\ \ell = 0, \ldots, L\}. \end{aligned} \tag{7}$$

These transformations can be used to transfer all evaluations of the kernel G on the reference grid $\Xi_\star$. Take a source cell s and a target cell t, and assume that they belong to the same level of their 2^d-tree and thus admit the same size and $\beta_s = \beta_t$. Any $\mathbf{x} \in s$ can be written as $\mathbf{x} = \boldsymbol{\alpha}_s + \beta_s \hat{\mathbf{x}}$ and any $\mathbf{y} \in t$ can be expressed as $\mathbf{y} = \boldsymbol{\alpha}_t + \beta_t \hat{\mathbf{y}}$, so that we have

$$\begin{aligned} &G(\mathbf{x}-\mathbf{y}) = \mathcal{G}_{s,t}(\hat{\mathbf{x}} - \hat{\mathbf{y}}) \\ &\text{with} \quad \mathcal{G}_{s,t}(\mathbf{z}) := G(\boldsymbol{\alpha}_s - \boldsymbol{\alpha}_t + \beta_s \mathbf{z}). \end{aligned} \tag{8}$$

The function $\mathbf{x}, \mathbf{y} \mapsto G(\mathbf{x}-\mathbf{y})$ induces a linear map $G(\Xi_t, \Xi_s) : \mathbb{C}[\Xi_s] \to \mathbb{C}[\Xi_t]$. Similarly $\mathbf{p}, \mathbf{q} \mapsto \mathcal{G}_{s,t}(\mathbf{p}-\mathbf{q})$ induces a linear map $\mathcal{G}_{s,t}(\Xi_\star, \Xi_\star) : \mathbb{C}[\Xi_\star] \to \mathbb{C}[\Xi_\star]$ and, by construction, these two linear maps can be identified to the same matrices under consistent orderings $G(\Xi_t, \Xi_s) = \mathcal{G}_{s,t}(\Xi_\star, \Xi_\star)$. Rewriting the kernel as above, we explicitly take advantage of the fact that $\beta_s = \beta_t$. When examining interactions between s and t, the normalized kernel $\mathcal{G}_{s,t}$ is evaluated over the difference grid

$$\begin{aligned} \Xi_\sharp &:= \{\mathbf{p}-\mathbf{q},\ \mathbf{p}, \mathbf{q} \in \Xi_\star\} \subset \mathbb{R}^d \\ &:= \{\ell/L,\ \ell \in \mathbb{N},\ -L \le \ell \le +L\}^d. \end{aligned} \tag{9}$$

Obviously $\Xi_\star \subset \Xi_\sharp$, and any vector $u \in \mathbb{C}[\Xi_\star]$ can be considered as an element of $\mathbb{C}[\Xi_\sharp]$ by means of extension by zero: define $\chi(u) \in \mathbb{C}[\Xi_\sharp]$ by $\chi(u)(\mathbf{p}) := u(\mathbf{p})$ if $\mathbf{p} \in \Xi_\star$ and $\chi(u)(\mathbf{p}) := 0$ otherwise. The linear map χ simply reduces to a matrix with entries equal to 0 or 1, with at most one non-vanishing entry per row, so applying such a matrix to a vector is computationally harmless. The application of this matrix numerically corresponds to a *zero-padding*.

We have $\mathrm{card}(\Xi_\sharp) = T^d$ with $T := (2L+1)$, i.e. $\Xi_\sharp$ contains T samples in each direction. The optimization we wish to exploit is based on Fast Fourier Transforms (FFTs), this is why we embed the action of (8) in a periodic setting. The kernel $\mathcal{G}_{s,t}$ can be extended by T/L-periodicity, denoting $\mathcal{G}^\sharp_{s,t}$ the unique element of $\mathbb{C}[\Xi_\sharp + \mathbb{Z}^d]$ satisfying

$$\begin{aligned} &\mathcal{G}^\sharp_{s,t}(\mathbf{p} + (T/L)\boldsymbol{\tau}) = \mathcal{G}_{s,t}(\mathbf{p}) \\ &\text{for all} \quad \mathbf{p} \in \Xi_\sharp, \boldsymbol{\tau} \in \mathbb{Z}^d. \end{aligned} \tag{10}$$

In particular $\mathcal{G}^\sharp_{s,t}(\mathbf{p}) = \mathcal{G}_{s,t}(\mathbf{p})$ for $\mathbf{p} \in \Xi_\sharp$. Denote $\mathcal{G}^\sharp_{s,t}(\Xi_\sharp, \Xi_\sharp)$ the matrix associated to the mapping from $\mathbb{C}[\Xi_\sharp] \to \mathbb{C}[\Xi_\sharp]$ associated to $p, q \mapsto \mathcal{G}^\sharp_{s,t}(\mathbf{p}-\mathbf{q})$. This matrix admits a circulant form due to

the periodicity of the corresponding kernel. The notations introduced above lead to a factorization of M2L interactions between cells s and t,

$$G(\Xi_s, \Xi_t) = \chi^\top \cdot \mathcal{G}^\sharp_{s,t}(\Xi_\sharp, \Xi_\sharp) \cdot \chi. \tag{11}$$

As we are considering tensorized interpolation grids, and since $\mathcal{G}^\sharp_{s,t}(\Xi_\sharp, \Xi_\sharp)$ admits circulant form, we naturally make use of a multi-dimensional Discrete Fourier Transform (DFT) to process this matrix. Let us briefly explain how. Given the dual grid $\widehat{\Xi}_\sharp = L \cdot \Xi_\sharp \subset \mathbb{R}^d$, the d-dimensional DFT refers to the linear maps $\mathbb{F} : \mathbb{C}[\Xi_\sharp] \to \mathbb{C}[\widehat{\Xi}_\sharp]$ defined by the formula

$$\mathbb{F}(u)(\boldsymbol{\xi}) := \frac{1}{T^{d/2}} \sum_{\mathbf{x} \in \Xi_\sharp} u(\mathbf{x}) \exp(-2\imath\pi\, \boldsymbol{\xi} \cdot \mathbf{x}) \qquad \boldsymbol{\xi} \in \widehat{\Xi}_\sharp. \tag{12}$$

The inverse DFT is simply the adjoint map defined by the formula $\mathbb{F}^*(\hat{u})(\mathbf{x}) = T^{-d/2} \sum_{\xi \in \widehat{\Xi}_\sharp} \hat{u}(\xi) \exp(+2\imath\pi\, \xi \cdot \mathbf{x})$ for $\mathbf{x} \in \Xi_\sharp$. Due to the circulant form of the matrix $\mathcal{G}^\sharp_{s,t}(\Xi_\sharp, \Xi_\sharp)$, there exists a diagonal matrix $\mathbb{D}_{s,t}$ such that

$$\begin{aligned} &\mathcal{G}^\sharp_{s,t}(\Xi_\sharp, \Xi_\sharp) = \mathbb{F}^* \cdot \mathbb{D}_{s,t} \cdot \mathbb{F} \\ &\text{hence} \quad G(\Xi_s, \Xi_t) = \chi^\top \cdot \mathbb{F}^* \cdot \mathbb{D}_{s,t} \cdot \mathbb{F} \cdot \chi. \end{aligned} \tag{13}$$

The diagonal of $\mathbb{D}_{s,t} = \mathrm{diag}_{\xi \in \widehat{\Xi}_\sharp}(\, \mathbb{F}(\mathcal{G}_{s,t})(\xi)\,)$ contains the Fourier coefficients of the kernel. From the factorized form above, we see that matrix-vector product for the M2L matrix $G(\Xi_s, \Xi_t)$ can be processed efficiently. Indeed the action of χ is inexpensive, and $\mathbb{F}$ can be applied efficiently with $\mathcal{O}(d\, L^d \log L)$ complexity by means of FFTs. Notice that the diagonal of $\mathbb{D}_{s,t}$ (i.e. its non-zero entries) is obtained by DFT of the first column of $\mathcal{G}^\sharp_{s,t}(\Xi_\sharp, \Xi_\sharp)$ and can also be efficiently processed using FFT.

Since the same Fourier basis is used for all M2L matrices, one can apply $\mathbb{F}$ and $\mathbb{F}^*$ (as well as χ and $\chi^\top$) once to each multipole and local expansions respectively *before* and *after* all the M2L evaluations. This reduces M2L matrix evaluations to products with diagonal matrices $\mathbb{D}_{s,t}$ (i.e. Hadamard products), each with a complexity of $(2L-1)^d$ flop. In comparison with the product of a vector by a rank-k matrix approximation of a M2L matrix, requiring $2kL^d$ operations, the product with the diagonalization of the circulant embedding of this matrix is *theoretically* less costly if $k \geq 2^{d-1}$, which is valid except when requesting very low overall accuracies. However, low-rank approaches usually rely on highly optimized implementations (namely BLAS routines) to significantly reduce the computation times of the dense matrix-vector (or matrix-matrix) products (see e.g. [24]). Therefore, theoretical operation counts are insufficient to compare a FFT-based approach with a low-rank one. We will have to compare practical time measurements using implementations of both approaches to determine the best one. One can already notice that, for non-oscillatory kernels, FFT-based approaches offer better or similar performance (depending on the interpolation order) than low-rank compressions for the M2L matrix evaluations [5,10]: this encourages us to investigate the FFT-based approaches for the oscillatory kernels.

In this purpose, we recall first that the interpolation process on equispaced grids is known to be subject to the Runge phenomenon, which may cause the process to diverge, especially for high interpolation orders (see for instance [10]).

3 Consistency of the equispaced interpolation

Despite the asymptotic instability of the Lagrange polynomial interpolation process on equispaced grids, we show in this section that the approximation (5) actually converges on well-separated sets.

We define the *Multipole Acceptance Criterion* (or MAC) $\mathcal{A}$ as a boolean function that takes two cells in argument and returns 1 if and only if these two cells are well-separated. As in [5, 16], we only consider cells at the same 2^d-tree level as input parameters of the MAC. These cells t, s thus have the same radius a, i.e. they are translations of $[-a, a]^d$ in $\mathbb{R}^d$ (see section 2.3). The interpolation grids in t, s can be expressed relatively to the cell centers, so the polynomial interpolation of G on $t \times s$ can be seen as an interpolation of $\tilde{G}$ on $[-a, a]^{2d}$ using $\tilde{G}(\mathbf{x}, \mathbf{y}) := G(ctr(t) + \mathbf{x}, ctr(s) + \mathbf{y})$. Here, $\tilde{G}$ depends on $t \times s$.

We now introduce and prove the following theorem, regarding the interpolation process consistency on our equispaced grids.

Theorem 1. *Suppose that the condition $\mathcal{A}(t, s) = 1$ implies that $\tilde{G}$ is analytic in each variable at any point in $[-a, a]$ with a convergence radius R such that $R > \frac{2a}{e}$. Then the Lagrange interpolation of $\tilde{G}$ on $t \times s$ using L equispaced points in each variable, denoted by $\mathcal{I}_L^{t\times s}[\tilde{G}]$, verifies $\lim_{L\to+\infty} \left|\left|\mathcal{I}_L^{t\times s}[\tilde{G}] - \tilde{G}\right|\right|_{L^\infty([-a,a]^{2d})} = 0$. More specifically, if $radius(s) + radius(t) \leq dist(t, s)$ and if G is asymptotically smooth, the convergence is geometric, i.e. $\exists\, \nu \in (0, 1)$ s.t. $||\mathcal{I}_L^{t\times s}[\tilde{G}] - \tilde{G}||_{L^\infty([-a,a]^{2d})} = \mathcal{O}\left(\nu^L\right)$.*

3.1 Preliminary results

The idea of the proof of Theorem 1 consists in combining 1D convergence estimates with the interpolation error results provided in [26] (restricted to our asymptotically smooth kernel functions). Our convergence proof differs from the one of [16, 25] because we cannot rely on the properties of the Chebyshev grids. We denote by $\mathcal{C}^\infty(\Omega)$ the space of multivariate functions f with bounded derivatives $\partial^\beta f := \partial^{\beta_1} f ... \partial^{\beta_{2d}} f$ on Ω, $\forall \beta \in \mathbb{N}^{2d}$ such that $\max_{k=1,...,2d} \beta_k \leq L$ for any domain $\Omega \subset \mathbb{R}^{2d}$. We denote by $\mathfrak{A} := \{\alpha = (\alpha_1, ..., \alpha_{2d}) \in \{0, 1\}^{2d}, \exists j, \alpha_j = 1\}$. We also use the notation $||f||_\infty := sup_{\mathbf{z}\in[-a,a]^{2d}} |f((\mathbf{z})|$.

Theorem 2. *([26] Thm. 2.1) For $f \in \mathcal{C}^\infty([-a, a]^{2d})$, the product interpolation $\mathcal{I}_L^{[-a,a]^{2d}}[f]$ of f in $[-a, a]^{2d}$ with the same 1D rule with L interpolation nodes in each variable verifies*

$$\left|\left|f - \mathcal{I}_L^{[-a,a]^{2d}}[f]\right|\right|_\infty \leq \sum_{\alpha\in\mathfrak{A}} \omega_L^{\bar{\alpha}} \left|\left|\partial^{\alpha L} f\right|\right|_\infty$$

where $\alpha L = (\alpha_1 L, ..., \alpha_{2d} L)$, $\bar{\alpha} := \sum_{k=1}^{2d} \alpha_k$ and $\omega_L := \frac{1}{L!} \left|\left| \prod_{k=0}^{L-1} (\cdot - x_k) \right|\right|_\infty$, x_k being the k^{th} interpolation point of the 1D rule.

For equispaced grids, the constant ω_L can be bounded using the following lemma.

Lemma 1. *Let $\{x_k := -a + 2ak/(L-1) \mid k \in [\![0, L-1]\!]\} \subset [-a, a]$ an interpolation grid of equispaced points. We have $\omega_L \leq \left(\frac{2a}{L-1}\right)^L / (4L)$.*

Proof. Let $x \in [-a, a]$. We have $\left| \prod_{j=0}^{L-1} (x - x_j) \right| = (2a)^L \left| \prod_{j=0}^{L-1} \left(\frac{x+a}{2a} - \frac{j}{(L-1)} \right) \right|$. Let $y := \left(\frac{x+a}{2a}\right) \in [0, 1]$. This leads to $\left| \prod_{j=0}^{L-1} (x - x_j) \right| \leq \left(\frac{2a}{L-1}\right)^L \left| \prod_{j=0}^{L-1} y(L-1) - j \right|$. Because $y(L-1) \in [0, (L-1)]$,

$\left|\prod_{j=0}^{L-1} y(L-1)-j\right|$ is maximal for $y \in (0, 1/(L-1))$. Using a simple recurrence, one may easily show that $\prod_{j=0}^{L-1}\left|y(L-1)-j\right| \leq \frac{(L-1)!}{4}$, which implies that $\omega_L \leq \frac{1}{L!}\left(\frac{2a}{L-1}\right)^L (L-1)!/4 \leq \left(\frac{2a}{L-1}\right)^L/(4L)$. □

We now want to bound the partial derivatives of the interpolated function. This is the purpose of the following lemma.

Lemma 2. *If f is analytic in all its variables at any point of $[-a,a]$ with a convergence radius $R>0$, we have $||\partial^{\alpha L} f||_\infty \leq \frac{C}{r^{2d}}\left(\frac{L!}{r^L}\right)^{\bar{\alpha}}$ with $0<r<R$, $\bar{\alpha} := \sum_{k=1}^{2d}\alpha_k$, $C \in \mathbb{R}^{*+}$ being a constant independent of L, and α being defined as in Thm. 2.*

Proof. Since f is analytic in all its variables, we can apply the Cauchy integral formula (see [3], Chapter 2, Theorem 6), allowing us to write:

$$f(\mathbf{p}) = \left(\frac{1}{2\pi i}\right)^{2d} \int_{\Gamma_1} ... \int_{\Gamma_{2d}} \frac{f(\mathbf{z})}{(z_1-p_1)...(z_{2d}-p_{2d})} dz_1...dz_{2d},$$

where $\Gamma_j := \{\xi \in \mathbb{C} \mid dist(\xi, [-a,a]) = r\}$. We thus have:

$$\begin{aligned}\left|\left|\partial_{\mathbf{p}}^{\alpha L} f(\mathbf{p})\right|\right|_\infty &\leq (2\pi)^{-2d} \int_{\Gamma_1} ... \int_{\Gamma_{2d}} f(\mathbf{z}) \left|\left|\partial_{\mathbf{p}}^{\alpha L} \frac{1}{(z_1-p_1)...(z_{2d}-p_{2d})}\right|\right|_\infty dz_1...dz_{2d} \\ &\leq \frac{|\Gamma_1|...|\Gamma_{2d}|}{(2\pi)} \left(\sup_{\mathbf{z}\in\Gamma_1\times...\times\Gamma_{2d}} \left|f(\mathbf{z})\right|\right) \left|\left|\partial_{\mathbf{p}}^{\alpha L} \frac{1}{(z_1-p_1)...(z_{2d}-p_{2d})}\right|\right|_\infty.\end{aligned}$$

The term $\left(sup_{\mathbf{z}\in\Gamma_1\times...\times\Gamma_{2d}}|f(\mathbf{z})|\right)$ is bounded thanks to the analyticity of f on a neighborhood of $[-a,a]^{2d}$ encompassing the Γ_j's (with a convergence radius equal to R). Hence, there exists a constant $M(\Gamma_1,...,\Gamma_{2d}) \in \mathbb{R}^{+*}$ such that, defining

$$C(\Gamma_1,...,\Gamma_{2d}) := (2\pi)^{-2d}|\Gamma_1|...|\Gamma_{2d}|M(\Gamma_1,...,\Gamma_{2d}),$$

where $|\Gamma_j|$ denotes the length of the path Γ_j, $j \in [\![1,2d]\!]$, with $\bar{\alpha} := \sum_{k=1}^{2d}\alpha_k$, we have

$$\begin{aligned}\left|\left|\partial^{\alpha L} f(\mathbf{p})\right|\right|_\infty &\leq C(\Gamma_1,...,\Gamma_{2d}) \left|\left|\partial^{\alpha L} \frac{1}{(z_1-p_1)...(z_{2d}-p_{2d})}\right|\right|_\infty \\ (\textit{Since } \alpha_k = 0,1) &\leq C(\Gamma_1,...,\Gamma_{2d}) \prod_{k=1}^{2d} \left(L!^{\alpha_k} \left|\left|(z_k-p_k)\right|\right|_\infty^{-L\alpha_k-1}\right) \\ &\leq C(\Gamma_1,...,\Gamma_{2d}) \underbrace{\prod_{k=1}^{2d} \left(L!^{\alpha_k} r^{-L\alpha_k-1}\right)}_{=\left(\frac{L!}{r^L}\right)^{\bar{\alpha}}/(r^{2d})}.\end{aligned}$$

□

3.2 Proof of the main theorem

We can now prove Thm. 1.

Proof. Following Lem. 1 and Thm. 2, we have,

$$\begin{aligned}\left|\left|\mathcal{I}_L^{t\times s}[\tilde{G}]-\tilde{G}\right|\right|_\infty &\leq \sum_{\alpha\in\mathfrak{A}}\left(\prod_{k=1}^{2d}\omega_L^{\alpha_k}\right)\left|\left|\partial^{\alpha L}\tilde{G}\right|\right|_\infty \\ &\leq \sum_{\alpha\in\mathfrak{A}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{L-1}\right)^L/(4L)\right)^{\alpha_k}\left|\left|\partial^{\alpha L}\tilde{G}\right|\right|_\infty\end{aligned}$$

which becomes, thanks to Lem. 2

$$\left|\left|\mathcal{I}_L^{t\times s}[\tilde{G}]-\tilde{G}\right|\right|_\infty \leq C\sum_{\alpha\in\mathfrak{A}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{L-1}\right)^L\frac{L!}{4r^L L}\right)^{\alpha_k} r^{-1}.$$

Now, by applying Stirling's inequality $L! \leq e^{-(L-1)}L^{L+1/2}$, one obtains

$$\begin{aligned}\left|\left|\mathcal{I}_L^{t\times s}[\tilde{G}]-\tilde{G}\right|\right|_\infty &\leq C\sum_{\alpha\in\mathfrak{A}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{L-1}\right)^L\frac{e^{-(L-1)}L^{L+1/2}}{4r^L L}\right)^{\alpha_k} r^{-1} \\ &\leq C\sum_{\alpha\in\mathfrak{A}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{L-1}\right)^L\frac{eL^{L+1/2}}{4(er)^L L}\right)^{\alpha_k} r^{-1} \\ &\leq C\sum_{\alpha\in\mathfrak{A}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{re}\right)^L\left(\frac{L}{L-1}\right)^L\frac{e}{4\sqrt{L}}\right)^{\alpha_k} r^{-1}.\end{aligned}$$

For $L \geq 2$, we have $\left(\frac{L}{L-1}\right)^L \leq 4$, which allows to write

$$\begin{aligned}\left|\left|\mathcal{I}_L^{t\times s}[\tilde{G}]-\tilde{G}\right|\right|_{L^\infty([-a,a]^{2d})} &\leq C\sum_{\alpha\in\mathfrak{A}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{re}\right)^L\frac{e}{\sqrt{L}}\right)^{\alpha_k} r^{-1} \\ &\leq Cr^{-2d}\sum_{\alpha\in\mathfrak{A}}\underbrace{\prod_{k=1}^{2d}\left(\left(\frac{2a}{re}\right)^L\frac{e}{\sqrt{L}}\right)^{\alpha_k}}_{=\left(\frac{2a}{re}\right)^{\bar{\alpha}L}\left(\frac{e}{\sqrt{L}}\right)^{\bar{\alpha}}}\end{aligned}$$

using $\bar{\alpha} = \sum_{k=1}^{2d}\alpha_k$. The number of terms in the sum over $\mathfrak{A}$ is finite and depends only on the dimension: there is indeed $2^{2d}-1$ terms in this sum since $\alpha = (0,...,0)$ does not verify $||\alpha||_\infty = 1$. This estimate thus tends to zero when L tends to infinity if $\frac{2a}{re} < 1$. Since $r < R$, this is verified if $\frac{2a}{e} < R$. Indeed, $\left(\frac{e}{\sqrt{L}}\right)^{\bar{\alpha}} \leq \left(\frac{e}{\sqrt{2}}\right)^{2d}$ since we assumed that $L \geq 2$. In addition, each $\left(\frac{2a}{re}\right)^{\bar{\alpha}L}$ can be bounded by $\left(\frac{2a}{re}\right)^L$ since $\frac{2a}{re} < 1$. We finally have

$$\left|\left|\mathcal{I}_L^{t\times s}[\tilde{G}]-\tilde{G}\right|\right|_{L^\infty([-a,a]^{2d})} \leq \underbrace{\left(Cr^{-2d}(2^{2d}-1)\left(\frac{e}{\sqrt{2}}\right)^{2d}\right)}_{\text{Does not depend on } L}\underbrace{\left(\frac{2a}{re}\right)^L}_{\underset{L\to+\infty}{\to}0}. \tag{14}$$

Now, let t and s be two cells with radius a at the same 2^d-tree level such that $dist(t,s) \geq 2a = radius(t) + radius(s)$. For asymptotically smooth kernel $G : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$, G is analytic with convergence radius $R = dist(t,s)/2$ (only singular at $\mathbf{x} = \mathbf{y}$). Hence, $R \geq a$. We can thus choose in inequality (14) $r \in \left(\frac{2R}{e}, R\right)$. Such a r exists since $\frac{2}{e} < 1$. We thus have

$$\frac{2}{e}\frac{dist(t,s)}{2} < r < \frac{dist(t,s)}{2} \Rightarrow \underbrace{\frac{radius(t)+radius(s)}{dist(t,s)}}_{=:\theta} > \frac{2a}{re} > \theta\frac{2}{e}.$$

Finally, because $\theta \leq 1$, we conclude from inequality (14) that for any $\nu \in \left(\frac{2}{e}\theta, \theta\right)$

$$\left|\left|\mathcal{I}_L^{t\times s}[\tilde{G}] - \tilde{G}\right|\right|_{L^\infty([-a,a]^{2d})} = \mathcal{O}\left(\nu^L\right). \tag{15}$$

□

This proof has a geometric interpretation. In the inequality (14), the term $\left(\frac{2a}{re}\right)^L$ somehow corresponds to a MAC such as in [14]: a refers to the radius of an interacting cell and r is related to the distance between the interacting cells. The greater this distance, the greater r and the better this estimate. Another information we obtain from the inequality (14) is that the convergence should be geometric in the 1D interpolation order. Indeed, considering $\theta := \frac{radius(t)+radius(s)}{dist(t,s)}$, Estimate (15) somehow indicates that the error of the interpolation process is $\mathcal{O}\left((c\theta)^L\right)$ with c a constant smaller than 1. In practice, our new directional MAC (defined in section 4.3.2 below), as well as the MAC from [16, 24, 25] (referred to as the *strict* MAC[4] in this article), both verify the assumptions of theorem 1: the interpolation process of G therefore converges on such well-separated sets.

In Fig. 2, we provide numerical investigation of the relative error for our equispaced interpolation with respect to the interpolation order in each variable on well-separated sets for two kernels matching the scope of Thm. 1 (namely asymptotically smooth kernels) and for dimensions $d = 1, 2, 3$ (similar results were obtained for higher dimensions but are not reported here since they are less relevant for our application cases). To this end, we considered well separated sets $X \subset [0,1]^d$ and $Y \subset [2,3]^d$ with 30000 randomly sampled particles in each, and randomly chosen charges at each particle in Y. Then, we computed the maximum relative error between a direct N-body problem on these sets and the approximated one, based on an equispaced interpolation on $[0,1]^d \times [2,3]^d$. These results numerically confirm, for any dimension, the expected geometric convergence of the equispaced interpolation applied on well-separated sets. Similar results (not shown here) were also obtained for other configurations of well-separated cells.

We conclude this section by two important remarks. First, the practical efficiency of the interpolation-based FMM relies on the implicit assumption that the constant $C(\Gamma_1, ..., \Gamma_{2d})$ is small. This is the case when using the strict MAC in the low-frequency regime (a justification can be found in [25]) and our new directional MAC (see section 4.3.2) in the high-frequency one. Second, the ill-conditioning of such interpolation on equispaced grids may cause an exponential amplification of floating-point rounding errors at the boundaries of the interpolation domain (see [29, 31]). We thus cannot expect a numerical convergence for any order in practice. Nevertheless, according to our tests (not shown here), we can reach a relative error of 10^{-12} on 3D particle distributions with double-precision arithmetic before facing numerical instabilities. Practical applications usually require much less accurate approximations.

[4]Two cells at the same 2^d-tree level comply with the strict MAC if the distance between them is greater or equal than their side length.

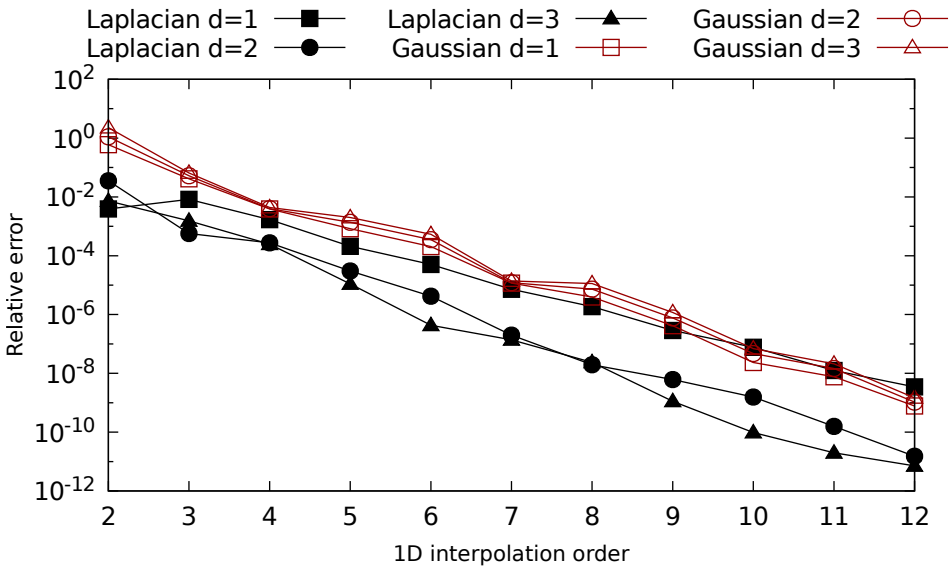


Figure 2: Relative error in maximum norm between matrix-vector product on well-separated cells using direct computation or using single equispaced interpolation with respect to the interpolation order in each variable. d denotes the dimension, *Laplacian* denotes the kernel $|\mathbf{x}-\mathbf{y}|^{-1}$, *Gaussian* denotes the kernel $e^{-|\mathbf{x}-\mathbf{y}|^2}$.

4 *defmm*: a directional equispaced interpolation-based FMM

In this section, we present our new *defmm* library, integrating polynomial interpolations on equispaced grids within a directional kernel-independent FMM.

4.1 Overall algorithm

Like directional FMM algorithms [15, 25], our FMM is divided into four mains steps:

1. a *precomputation step* that involves the 2^d-tree constructions and the precomputation of M2L matrices,
2. an *upward pass* consisting in applying the P2M and M2M operators along the source tree in order to compute the (possibly directional) multipole expansions,
3. a *horizontal pass* consisting in applying the M2L and P2P operators in order to transform (possibly directional) source multipole expansions into (possibly directional) target local expansions on well-separated pairs of cells,
4. a *downward pass* consisting in applying the L2L operators along the target tree in order to compute the leaf (possibly directional) local expansions that can be transformed into far-field contributions by means of L2P operators.

In the following sections, we detail how the efficient handling of equispaced interpolations as well as our algorithmic choices impact these different steps. Hence, in Sect. 4.2 we introduce new operators

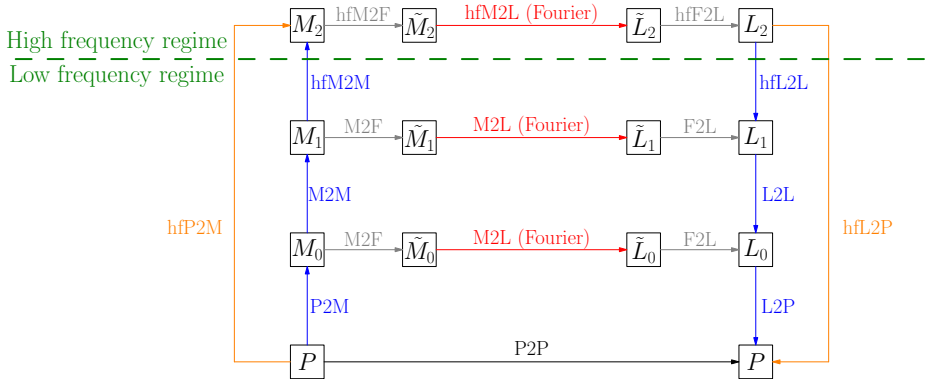


Figure 3: Operators and expansions in *defmm*. `hf` is added in front of the directional operators (used in the high-frequency regime). P: particles; M_i: multipole expansions; L_i: local expansions; $\tilde{M}_i$,$\tilde{L}_i$: multipole and local expansions in the Fourier domain.

that slightly modify the definition of the upward pass (step 2) and of the downward pass (step 4). In Sect. 4.3 we present how to efficiently perform the horizontal pass (step 3), as well as our strategy to minimize the number of precomputed M2L matrices (step 1). Finally, we show in Sect. 4.4 how the use of symmetries can further reduce the precomputation cost and how these techniques combine with the horizontal pass. The new final algorithm is provided in Sect. 4.5.

4.2 Directional FFT-based FMM

The integration of FFT techniques exploiting equispaced grids in a directional polynomial interpolation-based FMM requires operations between various directional expansions to be converted in the Fourier domain.

4.2.1 M2F and F2L operators

As described in section 2, the M2L application on equispaced grids involves three steps: the conversion of the involved multipole expansion (extended by zero-padding) into the Fourier domain, the application of the diagonal M2L operator in this domain, and then the conversion of the resulting (extended) local expansion in the Fourier domain into a local expansion. We thus introduce two extra operators, the M2F (multipole-to-Fourier) and F2L (Fourier-to-local) ones, applying the matrices $\mathbb{F}\chi$ to multipole expansions in the real domain and $\chi^T\mathbb{F}^*$ to local expansions in the Fourier domain. Let T be a M2L matrix, we have (see section 2.3)

$$T = \underbrace{\left(\chi^\top \mathbb{F}^*\right)}_{\text{F2L}} \quad \underbrace{\mathbb{D}_{s,t}}_{\text{Diagonal M2L}} \quad \underbrace{\left(\mathbb{F}\chi\right)}_{\text{M2F}}.$$

In Figure 3, we depict how the different operators are applied in *defmm*. The operators are non-directional (i.e. without the terms in grey in section 2.2) in the low-frequency regime, and directional in the high-frequency one.

4.2.2 Direction generation

In the directional FMM presented in [15] the symmetries within the sets of directions at each tree level are exploited to reduce the number of precomputed M2L matrices. This however constrains the way the directions are generated. On the contrary, using a directional interpolation-based approach the M2L matrices do not depend on the directions (see section 2.2). We thus exploit a similar construction algorithm as presented in [7, 15]: starting from a given regular solid, the set of projected face centers on the unit sphere are chosen as directions at the first high-frequency level. Then, each face is subdivided into 2^{d-1} other faces and the process is repeated recursively in order to obtain the direction tree. Engquist & Ying [15] relies on the d-cube for the direction generation: according to our tests, the d-cube provides indeed the best compromise between the number of directions and the obtained accuracy among the platonic solids. Since, a d-cube has $d!$ faces, there are $d!2^{E(d-1)}$ directions at the E^{th} high-frequency level in *defmm*.

4.3 Algorithmic design

4.3.1 2^d-tree construction

2^d-trees can be built either by setting the maximum number of particles per leaf (denoted as $Ncrit$), or by setting the maximum possible level (denoted as $MaxDepth$). For (highly) non-uniform particle distributions, the $MaxDepth$ strategy leads to (numerous) empty cells, which should not be stored and processed in practice: this requires advanced data structures (see e.g. [13, 22, 27]). The $MaxDepth$ strategy may also generate cells with a greatly varying number of particles [28]. Moreover, a comparison of hierarchical methods in [17] has detailed how the $Ncrit$ strategy better adapts to highly non-uniform distributions than the $MaxDepth$ one. Considering the highly non-uniform distributions used in BIE problems, we therefore choose to rely on the $Ncrit$ strategy in *defmm*. This differs from *dfmm* [24, 25] which relies on the $MaxDepth$ strategy.

Concretely, our 2^d-tree construction algorithm is similar to the *exafmm*[5] one: the particles are first sorted according to the Morton ordering, then each cell is built with a pointer on its first particle. The particles within a cell are thus stored consecutively in memory and followed by the particles of the next cell in the Morton ordering. This ensures that the data is localized for the P2P evaluations. In addition, since the particles of the sons of a given cell are stored consecutively, we can directly apply P2P operators on non-leaf cells. The charges and potentials are stored in two dedicated arrays whose entries correspond to the sorted particles for the same data-locality reason.

We also store all the directional expansions associated to a given cell c in the same array to enhance data locality during the translations of local and multipole expansions: the M2M and L2L evaluations regarding c are indeed performed alltogether (see section 5.2 for the optimizations of these steps).

4.3.2 Dual Tree Traversal

Using the $Ncrit$ criterion however complexifies the interaction list structure (see lists U, V, W and X in e.g. [32]). In directional FMMs, the DPSC for well-separateness (see equation (4)) further complicates the construction of the interaction lists (multiple lists for each target cell with varying sizes and shapes depending on the tree level). We therefore adapt here the Dual Tree Traversal (DTT) [14, 33] to directional FMMs. The DTT is a simple and recursive procedure that simultaneously traverses the target and source trees to evaluate *on-the-fly* the M2L and P2P operators by dynamically testing the MAC on pairs of target and source cells. Hence, the interaction lists are

[5]State-of-the-art parallel C++ (non-directional) FMM library: `https://github.com/exafmm/exafmm`

Algorithm 1 Dual Tree Traversal (DTT) between target cell t and source cell s

```
{A: strict (resp. directional) MAC in low- (resp. high-) frequency regime}
if A(t, s) then
    apply the M2L operator between t and s
    return
else
    if t is a leaf or s is a leaf then
        apply the P2P operator between t and s
        return
    end if
    for t' ∈ Sons(t) do
        for s' ∈ Sons(s) do
            DTT(t', s')
        end for
    end for
end if
```

traversed and evaluated implicitly but never built. This is another difference with *dfmm* which relies on a DTT-like algorithm to *explicitly* build the interaction lists during a precomputation step, and then separately processes these lists for each target cell.

Since FFTs can only efficiently accelerate M2L operations between cells at the same level (because of the relatively small involved circulant embedding), our DTT differs from the original one [14, 33] as follows: (i) when the MAC fails, both cells are split (not the largest one); (ii) a P2P operation is required as soon as one of the two cells is a leaf (not both).

We also aim at one single DTT algorithm for both the low- and high-frequency regimes. This requires the MAC to depend only on the cell radii and distances, and not on directions. Fortunately, this is possible in a directional polynomial interpolation-based FMM, since the M2L matrices do not depend on the directions. Our DTT hence relies in the low-frequency regime on the strict MAC (see section 3.2), which is simple and allows the most far-field interactions for each target cell, and on the following new directional MAC in the high-frequency regime (for wavenumber κ) between cells t, s with radius w at the same tree level:

$$\frac{max\{\kappa w^2, 2w\}}{dist(t,s)} \leq \eta, \tag{16}$$

where $\eta > 0$ ($\eta = 1$ in practice). This new directional MAC is similar to another directional MAC [7], but contrary to this latter our MAC does not depend on the directed wedge condition: we do not consider directions in this MAC but only ratios between cell radii and the cell distance. This means that the DTT performs in the high-frequency regime in a similar way than in the low-frequency one, without considering the directional aspects that are entirely hidden in the M2L applications. We emphasize that the interpolation process using equispaced grids is consistant according to section 3 on cells complying with these two MACs. When this directional MAC is satisfied, one then has to retrieve the best direction to select the directional multipole expansion (from source s) and the directional local expansion (from target t) which are relevant for the M2L operation between t and s. The search of the best direction may have a non-negligible cost, but there is one single direction associated to each M2L matrix. We thus precompute these best directions during the M2L matrix precomputation step (see section 4.3.3).

Algorithm 2 Blank Dual Tree Traversal (BDTT) between target cell t and source cell s

1: **if** t is in the high-frequency regime **then**
2: **if** $\mathcal{A}(t,s)$ **then**
3: $\{\mathcal{D}(Level(t))$ is the level of t in the direction tree: $\}$
4: compute direction $u := \underset{v\in\mathcal{D}(Level(t))}{argmin} \left| v - \frac{ctr(t)-ctr(s)}{|ctr(t)-ctr(s)|} \right|$
5: mark t and s with u
6: **if** the M2L matrix $\mathbb{D}_{s,t}$ corresponding to t and s is not precomputed **then**
7: precompute $\mathbb{D}_{s,t}$
8: **end if**
9: mark $\mathbb{D}_{s,t}$ with u
10: **return**
11: **else**
12: **for** $t' \in Sons(t)$ **do**
13: **for** $s' \in Sons(s)$ **do**
14: BDTT(t', s')
15: **end for**
16: **end for**
17: **end if**
18: **end if**

In the end, we obtain the simple DTT algorithm presented in Algorithm 1 to differentiate in *defmm* the far-field M2L operations from the near-field P2P ones, in both low- and high-frequency regimes.

One may notice that list-based approaches (i.e. without DTT) allow to group or "stack" multiple M2L operations (written as matrix-vector products), into matrix-matrix products (see e.g. [13, 23, 24]). This enables indeed to benefit from the higher efficiency of level-3 BLAS routines, especially for uniform distributions. We refer to this technique as *vector stacking*. Here however, we have M2L operations corresponding to Hadamard products whose grouping cannot lead to more efficient level-3 BLAS operations. Such Hadamard products may be converted to matrix-vector products as shown in [23], but this requires extra zeros for non-uniform particle distributions. Considering the highly non-uniform distributions typical of BIE problems, we believe that the potential gain would be too limited. We hence do not consider vector stacking for our Hadamard products, and we rather rely on the DTT to efficiently process the BIE non-uniform distributions.

4.3.3 Blank passes

On non-uniform distributions only a subset of all possible directional expansions will be used in practice due to the DPSC. Hence, *defmm* determines the actually needed directional expansions (named *effective* expansions) during a precomputation step and only computes and stores these effective expansions to save memory and computations. In this purpose, *defmm* performs first a *blank DTT* (see Algorithm 2) to compute each required M2L matrix and to mark all M2L-interacting cells with the corresponding direction (see line 5 in Algorithm 2). Then, thanks to a *blank downward pass* (see algorithm 3), the required directions are propagated down to the leaves of the 2^d-trees.

Algorithm 3 Blank Downward Pass (BDP) for cell c

```
if c is in the high-frequency regime then
  for each direction u with which c is marked do
    for c' ∈ Sons(c) do
      mark c' with Father(u)
    end for
  end for
  for c' ∈ Sons(c) do
    BDP(c')
  end for
end if
```

4.4 Exploiting symmetries in the Fourier domain

To minimize the number of M2L matrices to be precomputed, in the case of centered expansions in 2^d-trees, it has been observed in the literature that symmetries can be used (see for instance [15,24]). The underlying symmetry group is actually the hyperoctahedral one [12] (i.e. the octahedral group in 3D, corresponding to the symmetry group of the cube), denoted by $\mathfrak{D}_d$ in dimension d, that can be realized as a group of rotation matrices. The use of symmetries strongly reduces the precomputation cost, especially in the high-frequency regime. Hence, we want to also exploit these symmetries in *defmm*. To do so, one has to express these symmetries in the Fourier domain, since the (modified diagonal) M2L matrices are expressed in this domain.

4.4.1 Taking symmetries into account

Let us first detail how, in a given cell c of the (source or target) tree, a permutation matrix over the interpolation grid Ξ_c can be associated to a symmetry. Take any rotation $R : \mathbb{R}^d \to \mathbb{R}^d$ that leaves the unit cube centered at 0 invariant. The set of rotations satisfying this property forms the so-called hyperoctahedral group. The translated grid $\Xi_c - ctr(c)$ is centered at the origin, so $\Xi_c - ctr(c) = R(\Xi_c - ctr(c))$ i.e. $\Xi_c = ctr(c) + R(\Xi_c - ctr(c))$. This transformation, represented in Figure 4, induces a linear map $\mathbb{R}_c : \mathbb{C}[\Xi_c] \to \mathbb{C}[\Xi_c]$ defined by

$$\begin{aligned} &\mathbb{R}_c(u)(\mathbf{p}) := u(ctr(c) + R(\mathbf{p} - ctr(c))) \\ &\forall u \in \mathbb{C}[\Xi_c],\ \forall \mathbf{p} \in \Xi_c. \end{aligned} \tag{17}$$

This matrix $\mathbb{R}_c$ is a permutation of interpolation nodes, and the correspondence $R \leftrightarrow \mathbb{R}_c$ implements a permutation representation of the hyperoctahedral group. This is the group representation that we propose to exploit to minimize the number of precomputed M2L matrices.

Denote $\mathbb{R}_\star : \mathbb{C}[\Xi_\star] \to \mathbb{C}[\Xi_\star]$ the permutation induced on $\Xi_\star$ by means of the correspondence described above. When the same rotation transformation is applied to both source and target cells s and t, the transformed M2L matrix writes

$$\begin{aligned} \mathbb{R}_t^\top G(\Xi_t, \Xi_s)\mathbb{R}_s &= \mathbb{R}_\star^\top \mathcal{G}_{s,t}(\Xi_\star, \Xi_\star)\mathbb{R}_\star \\ &= \mathbb{R}_\star^\top \chi^\top \mathbb{F}^* \mathbb{D}_{s,t} \mathbb{F} \chi \mathbb{R}_\star. \end{aligned} \tag{18}$$

We are going to show that the product above actually writes $\chi^\top \mathbb{F}^* \tilde{\mathbb{D}}_{s,t} \mathbb{F} \chi$ where $\tilde{\mathbb{D}}_{s,t}$ is a diagonal matrix mapping $\mathbb{C}[\widehat{\Xi}_\star] \to \mathbb{C}[\widehat{\Xi}_\star]$ with coefficients obtained by permuting the diagonal of $\mathbb{D}_{s,t}$.

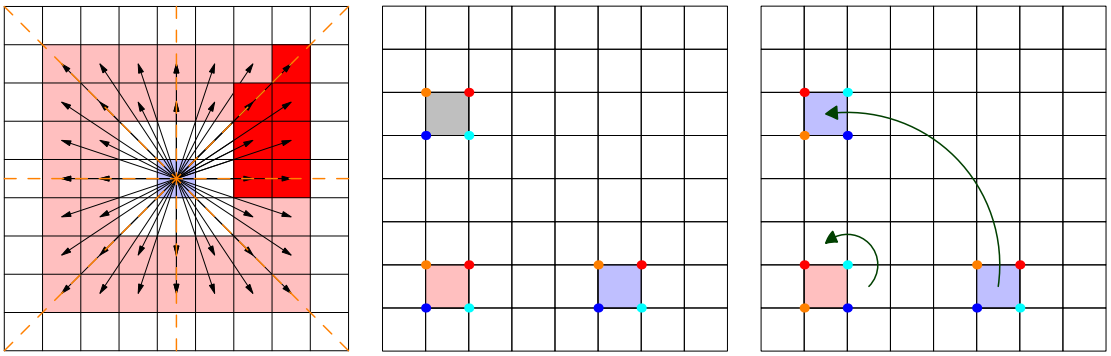


Figure 4: Left: Source cell s in the low-frequency regime (blue) with possible target cells t (red) such that s is in the interaction list of t. The corresponding M2L matrices for cells t in pale red can be all deduced (for instance) from permutations of the dark red ones. Symmetry axes of the square (2-cube) are represented in orange. Middle and right: M2L matrix between the red and blue well-separated cells (middle) as a rotation of the M2L matrix between the red and grey cells (right), that permutes the relative positions of the interpolation nodes (colored dots). Since the entries of the M2L matrices correspond to interpolation nodes, permutations matrices represent this process. The underlying symmetry of $\mathfrak{D}_2$ is the reflection with regard to the line $x = y$.

In concrete computational terms, this means that, whenever two M2L interactions only differ by a rotation, one M2L matrix is deduced from the other by a simple permutation of the diagonal entries of the factor term $\mathbb{D}_{s,t}$, which reduces storage and precomputation cost.

Let $c_\star = (1/2, \dots, 1/2) = ctr(\Xi_\star) \in \mathbb{R}^d$ refer to the center of the normalized interpolation grid. Denote $\Xi_0 = \Xi_\star - ctr(\Xi_\star)$ the normalized interpolation grid translated so as to be centered at 0. Take two vectors $u, v \in \mathbb{C}[\Xi_\star]$. Then we have

$$\begin{aligned} & v^\top \mathbb{R}_\star^\top \mathcal{G}_{s,t}(\Xi_\star, \Xi_\star)\mathbb{R}_\star u \\ &= \sum_{\mathbf{p}\in\Xi_\star} \sum_{\mathbf{q}\in\Xi_\star} \mathcal{G}_{s,t}(\mathbf{p}-\mathbf{q}) u(c_\star + R(\mathbf{p}-c_\star)) v(c_\star + R(\mathbf{q}-c_\star)) \\ &= \sum_{\mathbf{x}\in\Xi_0} \sum_{\mathbf{y}\in\Xi_0} \mathcal{G}_{s,t}(\mathbf{x}-\mathbf{y}) u(c_\star + R(\mathbf{x})) v(c_\star + R(\mathbf{y})) \\ &= \sum_{\mathbf{x}'\in\Xi_\star} \sum_{\mathbf{y}'\in\Xi_\star} \mathcal{G}_{s,t}(R^*(\mathbf{p}-\mathbf{q})) u(\mathbf{p}) v(\mathbf{q}). \end{aligned} \tag{19}$$

Since u, v are arbitrarily chosen in $\mathbb{C}[\Xi_\star]$, this can be rewritten in condensed form by $\mathbb{R}_\star^\top \mathcal{G}_{s,t}(\Xi_\star, \Xi_\star)\mathbb{R}_\star = (\mathcal{G}_{s,t} \circ R^*)(\Xi_\star, \Xi_\star)$. Now there only remains to return to the calculus presented in section 2.3 that shows

$$\begin{aligned} & \mathbb{R}_\star^\top \mathcal{G}_{s,t}(\Xi_\star, \Xi_\star)\mathbb{R}_\star = \chi^\top \mathbb{F}^* \, \mathbb{D}[\mathcal{G}_{s,t} \circ R^*] \, \mathbb{F}\chi \\ & \text{with} \quad \mathbb{D}[\mathcal{G}_{s,t} \circ R^*] = \mathrm{diag}(\, \mathbb{F}(\mathcal{G}_{s,t} \circ R^*)\,). \end{aligned} \tag{20}$$

To summarize, the only difference between $\mathbb{R}_\star^\top \mathcal{G}_{s,t}(\Xi_\star, \Xi_\star)\mathbb{R}_\star$ and $\mathcal{G}_{s,t}(\Xi_\star, \Xi_\star)$ lies in the coefficients of the Fourier symbol in the central diagonal term of their factorized form. Let us examine how the

rotation R^* acts on the Fourier symbol of $\mathcal{G}_{s,t}$. According to (12), and since $R(\Xi_\sharp) = \Xi_\sharp$, we have

$$\begin{aligned}\mathbb{F}(\mathcal{G}_{s,t} \circ R^*)(\boldsymbol{\xi}) &= \frac{1}{T^{d/2}} \sum_{\mathbf{x} \in \Xi_\sharp} \mathcal{G}_{s,t}(R^*\mathbf{x}) \exp(-2\imath\pi\, \boldsymbol{\xi} \cdot \mathbf{x}) \\ &= \frac{1}{T^{d/2}} \sum_{\mathbf{y} \in \Xi_\sharp} \mathcal{G}_{s,t}(\mathbf{y}) \exp(-2\imath\pi\, R^*(\boldsymbol{\xi}) \cdot \mathbf{y}) = \mathbb{F}(\mathcal{G}_{s,t})(R^*\boldsymbol{\xi})\end{aligned} \tag{21}$$

which again summarizes as $\mathbb{F}(\mathcal{G}_{s,t} \circ R^*) = \mathbb{F}(\mathcal{G}_{s,t}) \circ R^*$. In other words, the permutation on the grid $\widehat{\Xi}_\sharp$ that should be applied on the symbol of the periodized M2L operator is the permutation associated to the inverse rotation $R^* = R^{-1}$.

4.4.2 Symmetries in practice

Usually, the multipole and local expansions are permuted respectively before and after the evaluation of a M2L operator (see e.g. [23,24]). Here we can reduce this number of applied permutations to only one thanks to the diagonal form of the M2L matrices in the Fourier domain in *defmm* (see equation (20)). Indeed, the permutations can all be applied to these diagonal M2L matrices, resulting in a permutation of their diagonal entries. This reduces the number of permutations from two to only one.

In practice, the permutations induce memory indirections when performing the Hadamard products that may prevent the compiler auto-vectorization. We hence rely on an OpenMP[6] directive to enforce the compiler vectorization (see [12] for details).

4.5 Final algorithm

We here summarize the main steps of our final algorithm, along with our improvements. Following Sect. 2.2, we denote in grey the terms which are specific to the high-frequency regime.

1. *Precomputation step*
   (a) Build target and source 2^d-trees $\mathcal{T}$ and $\mathcal{S}$
   (b) Apply Blank DTT (Alg. 2) to $\mathcal{T}$ and $\mathcal{S}$ using symmetries (Sect. 4.4)
   (c) Apply Blank Downward Pass (Alg. 3) to $\mathcal{T}$ and $\mathcal{S}$
2. *Directional Upward Pass*: for each cell s [in the high frequency regime] of $\mathcal{S}$
   (a) Compute the [effective] multipole expansions of s (Sect. 4.3.3) from the [effective] multipole expansion[s] of the sons of s using [hf]M2M if s is not a leaf, and [hf]P2M otherwise (Fig. 3)
   (b) Transform the [effective] multipole expansion[s] of c in the Fourier domain using [hf]M2F (Sect. 4.2.1)
3. *Horizontal Pass*
   (a) Apply DTT to $\mathcal{T}$ and $\mathcal{S}$ (Alg. 1) using permutations to handle symmetries when performing the M2L operations (Sect. 4.4.2)
4. *Downward Pass*: for each cell t of $\mathcal{T}$

[6] See: `https://www.openmp.org/`

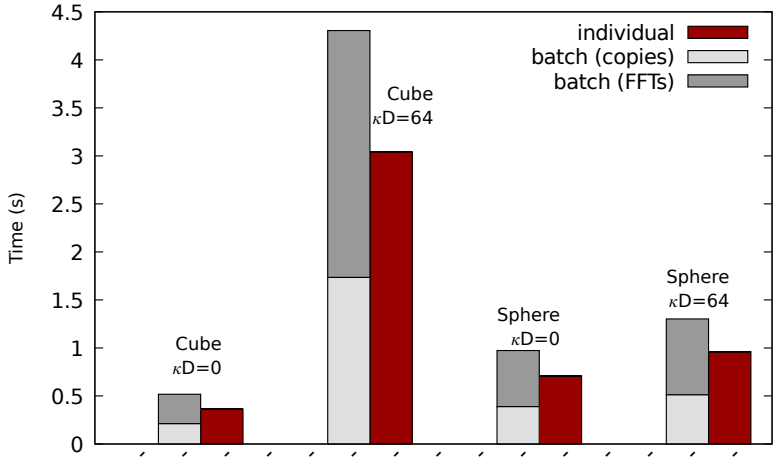


Figure 5: Timings of all FFT applications with 1D interpolation order 4, using the batch and individual methods. Tests performed with 10^7 particles (see section 6 for details).

(a) Transform back the [effective] local expansion[s] of t from the Fourier domain using [hf]F2L (Sect. 4.2.1)

(b) Compute the [effective] local expansion[s] of t from the [effective] local expansion[s] of $Father(t)$ using [hf]L2L (Sect. 4.3.3)

(c) If t is a leaf, perform [hf]L2P on the [effective] local expansion[s] of t

This high-level algorithm may be easily implemented using recursion for the different passes. We refer the interested reader to the public version of our code.

5 Optimizations

We here present algorithmic and programming optimizations regarding high performance computing on one CPU core.

5.1 Processing the FFTs

We rely on the state-of-the-art FFTW library [18] to efficiently perform all at once our numerous FFTs. Our 3D FFTs (considering $d = 3$) are however small: $(2L-1)^d$ elements, with usually $L \leq 7$. This makes the FFTW "plan" creation, required to decide the most efficient FFT implementation, more time consuming than its execution (i.e. the actual FFT computation). Moreover, zero-padding is required for the circulant embedding (see section 4.4) of the expansions stored in the Fourier domain: this implies expansion copies before each FFT, as well as after each reverse FFT.

One could first use the "batch" FFTW feature[7] to efficiently process our numerous small FFTs. This performs all FFTs with one single FFTW plan and can improve performance compared to multiple individual FFT calls. All expansion copies hence have to be performed all together before and after the batch FFTW call. We refer to this method as the *batch* one.

[7]See: `http://www.fftw.org/fftw3_doc/Advanced-Complex-DFTs.html`

Since all our expansions have the same size and the same memory alignment, we can also rely on one single FFTW plan for all our FFTs, and perform individually each FFT along with its expansion copy. This *individual* method benefits from the cache memory for our small-sized expansions, and hence avoids to load data twice from main memory (for FFTs and for copies, which are both memory-bound operations) as done in the batch method. This is shown in figure 5, where the individual method always outperforms (up to 30%) the batch one. We thus use the individual method in *defmm*.

5.2 BLAS-based upward and downward pass operators

Because of the directional aspects of *defmm*, the cost of the upward and downward passes is linearithmic for surface meshes (see [15]). As opposed to FMMs for non-oscillatory kernels, these steps have a significant cost in practice, which justifies their careful optimization. We hence first adapt an optimization suggested in [2] to equispaced grids and to oscillatory kernels (see section 5.2.1), and then improve it for directional FMMs (see sections 5.2.2 and 5.2.3). We validate our optimizations in section 5.2.4.

5.2.1 Tensorized M2M and L2L operators

We here detail a fast evaluation scheme suggested in [2] for the M2M and L2L operators on interpolation grids using tensorized Chebyshev rules in the low-frequency regime. We extend this scheme to equispaced grids which are also tensorized grids. The bijection in definition 1 induces a node indexing in the interpolation grid $\mathbb{G}$ allowing to exploit this tensorized structure.

Definition 1. *Let $L \in \mathbb{N}$. $\mathfrak{I}$ denotes the bijection from $[\![0, L^d - 1]\!]$ to $[\![0, L-1]\!]^d$ such that $\mathfrak{I}^{-1}(\mathbf{I}) := \sum_{k=1}^{d} I_k L^{k-1}$, $\forall\, \mathbf{I} := (\mathfrak{I}_1, ..., \mathfrak{I}_d)$.*

Thanks to this tensorized structure, the matrix representations of the M2M and L2L operators are tensorized matrices. The L2L case being obtained by transposition, we focus on the M2M case. Let $\mathbb{M} \in \mathbb{R}^{L^d \times L^d}$ be the matrix representation of a M2M operation, there exists therefore $M^{(p)} \in \mathbb{R}^{L \times L}$, $p \in [\![1, d]\!]$, such that $\mathbb{M} = \bigotimes_{p=1}^{d} M^{(p)}$. Using definition 1, the following holds

$$\left(\bigotimes_{k=1}^{d} M^{(k)} \right)_{i,j} = \prod_{k=1}^{d} M^{(k)}_{\mathfrak{I}(i)_k, \mathfrak{I}(j)_k}.$$

For any $\mathbf{v} \in \mathbb{C}^{L^d}$, we have

$$\begin{aligned}
(\mathbb{M}\mathbf{v})_i &= \left(\left(\bigotimes_{k=1}^{d} M^{(k)} \right) \mathbf{v} \right)_i = \sum_{j=0}^{L^d-1} \left(\prod_{p=1}^{d} M^{(p)}_{\mathfrak{I}(i)_p, \mathfrak{I}(j)_p} \right) v_j \\
&= \sum_{\mathbf{J} \in [\![0, L-1]\!]^d} \prod_{p=1}^{d} M^{(p)}_{\mathfrak{I}(i)_p, \mathbf{J}_p} v_{\mathfrak{I}^{-1}(\mathbf{J})} = \sum_{\mathbf{J} \in [\![0, L-1]\!]^d} \prod_{p=1}^{d-1} M^{(p)}_{\mathfrak{I}(i)_p, \mathbf{J}_p} \left(M^{(d)}_{\mathfrak{I}(i)_d, \mathbf{J}_d} v_{\mathfrak{I}^{-1}(\mathbf{J})} \right) \\
&= \sum_{\substack{\mathbf{J} \in [\![0, L-1]\!]^d \\ \mathbf{J}_d = 0}} \prod_{p=1}^{d-1} M^{(p)}_{\mathfrak{I}(i)_p, \mathbf{J}_p} \left(\sum_{q=0}^{L-1} M^{(d)}_{\mathfrak{I}(i)_d, q} v_{\mathfrak{I}^{-1}(\mathbf{J} + q\mathbf{e}_d)} \right)
\end{aligned}$$

where the last sum over q matches a matrix-vector product of size $L \times L$. For i varying, this matrix-vector product is performed on L^{d-1} different restrictions of $\mathbf{v}$, involving each time the same matrix. Hence, there exists a permutation $P \in \mathbb{R}^{L^d \times L^d}$ such that

$$(\mathbb{M}\mathbf{v})_i = \sum_{\substack{\mathbf{J} \in [\![0, L-1]\!]^d \\ \mathbf{J}_d = 0}} \prod_{p=1}^{d-1} M^{(p)}_{\mathfrak{I}(i)_p, \mathbf{J}_p} \left(diag(M^{(d)}) P \mathbf{v} \right)_{\mathfrak{I}(i)_d}$$

where $diag(M^{(d)})$ is a block-diagonal matrix with all diagonal bocks equal to $M^{(d)}$. This process can be repeated d times, leading to an overall complexity of $\mathcal{O}\left(dL^{d+1}\right)$ since the permutations are applied in $\mathcal{O}(L^d)$ operations. This compares favorably with the $\mathcal{O}\left(L^{2d}\right)$ complexity of a naive approach. Since the same matrix is used for multiple vectors at each of the d iterations, matrix-vector products can be stacked into matrix-matrix products to benefit from the level-3 BLAS higher efficiency [2]. We will refer to this version as the *tensorized* (or t) method.

The extension of the tensorized method to oscillatory kernels is obtained by noting that the directional M2M matrix $\mathbb{M}(u)$ with direction u can be written

$$\mathbb{M}(u) = D_0(u) \left(\bigotimes_{p=1}^{d} M^{(p)} \right) D_1(u) \tag{22}$$

with two diagonal matrices $D_0(u)$ and $D_1(u)$ composed of complex exponential evaluations (see section 2.2).

5.2.2 Directional stacking

The tensorized method can be further optimized in the high-frequency regime. Starting from equation 22, if we consider for a given cell two directions u and u' with corresponding directional multipole expansions $\mathbf{v}(u)$ and $\mathbf{v}(u')$, denoting by $\odot$ the Hadamard product, the "stacking"

$$\begin{bmatrix} \mathbb{M}(u)\mathbf{v}(u) & \mathbb{M}(u')\mathbf{v}(u') \end{bmatrix}$$

can be expressed as

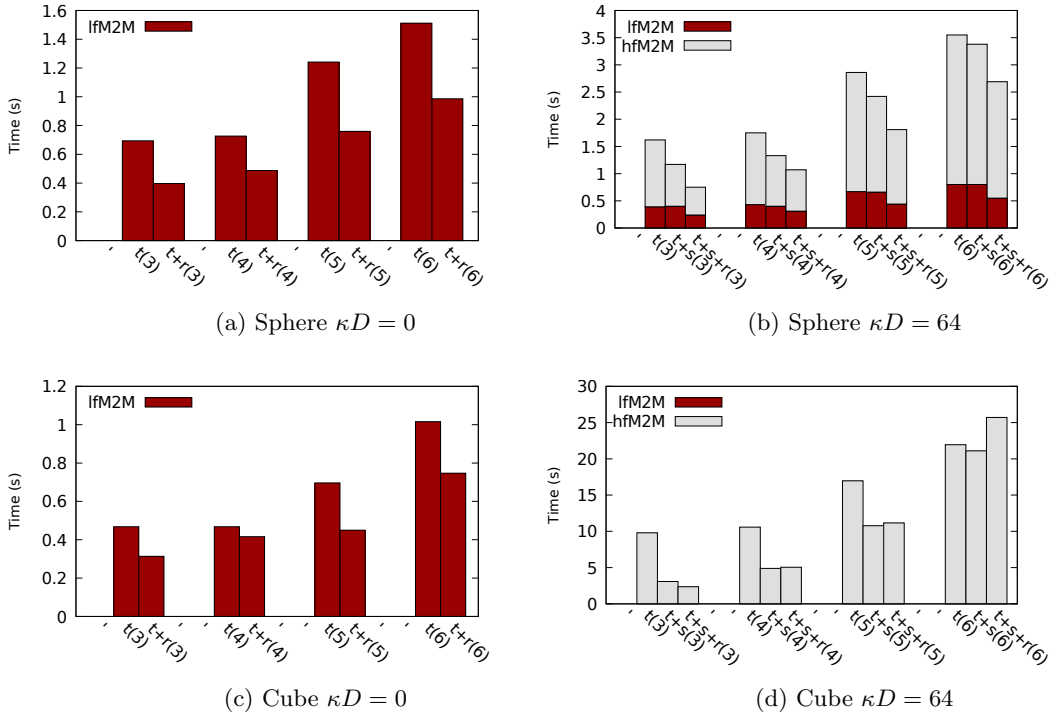
$$\begin{bmatrix} D_0(u) & D_0(u') \end{bmatrix} \odot \left(\left(\bigotimes_{p=1}^{d} M^{(p)} \right) \cdot \left(\begin{bmatrix} D_1(u) & D_1(u') \end{bmatrix} \odot \begin{bmatrix} \mathbf{v}(u) & \mathbf{v}(u') \end{bmatrix} \right) \right).$$

$\otimes_{p=1}^{d} M^{(p)}$ here applies to a matrix instead of a vector, which allows to further benefit from the level-3 BLAS efficiency. Contrary to the tensorized method where L^{d-1} vectors can be stacked, the number of vectors that can be stacked is now L^{d-1} multiplied by the number of effective directional multipole expansions in each cell. This method is referred to as the *tensorized+stacking* method (denoted $t + s$).

5.2.3 Benefiting from real matrix products

One may observe that the matrices $M^{(p)}$ are real since they are composed of evaluations of Lagrange polynomials, but are applied to complex vectors (at least in a directional method). For any $\mathbf{v} \in \mathbb{C}^{L^d}$, we thus have

$$\left(\bigotimes_{p=1}^{d} M^{(p)} \right) \mathbf{v} = \left(\bigotimes_{p=1}^{d} M^{(p)} \right) \mathfrak{Re}\{\mathbf{v}\} + i \left(\bigotimes_{p=1}^{d} M^{(p)} \right) \mathfrak{Im}\{\mathbf{v}\}.$$



(a) Sphere $\kappa D = 0$

(b) Sphere $\kappa D = 64$

(c) Cube $\kappa D = 0$

(d) Cube $\kappa D = 64$

Figure 6: Timings of all M2M evaluations for the tensorized (t), tensorized+stacking ($t + s$) and tensorized+stacking+real ($t + s + r$) methods, for different distributions with 10^7 particles and for different frequency regimes. κD refers to the length of the particle distribution multiplied by the wavenumber κ. 1D interpolation orders are indicated inside parentheses. lfM2M (respectively hfM2M) refers to the low- (resp. high-) frequency M2M operations. See section 6.2 for a detailed test case description.

By deinterleaving the real and imaginary parts in the expansion vectors, one can obtain stacked matrices of real elements. Hence, by discarding the imaginary part of the (real) M2M matrices, we can halve the number of required arithmetic operations. This method is referred to as the *tensorized+stacking+real* method (denoted $t + s + r$). Since there is only one multipole and one local expansion per cell in the low-frequency regime, no stacking of the directional expansions is performed and the $t + s + r$ method is reduced to $t + r$ in this frequency regime.

5.2.4 Performance results

In figure 6 are detailed the M2M timings for these three optimizations. Regarding the sphere with $\kappa D = 64$ (see figure 6b), the new $t + s$ method fastens the M2M evaluations of the t method in the high-frequency regime, while our $t + s + r$ method further reduces the M2M evaluation times of this $t + s$ method in the two frequency regimes. In the end, our $t + s + r$ method outperforms the original t method of [2] by a factor up to 2.16×.

This is also valid for the cube with $\kappa D = 64$ (see figure 6d), when considering low interpolation orders: for $L = 3$ the $t + s + r$ method outperforms the t one by a factor of $\approx 4\times$. For larger

interpolation orders, the $t+s+r$ method is however less efficient than the $t+s$ one, and even leads to a performance loss for $L = 6$. This is due to our small matrix sizes (one matrix being of size $L \times L$) for which the $t+s$ method already provides a high enough number of stacked vectors with the cube distribution: there is thus no benefit with a higher number of stacked vectors (as provided by the $t+s+r$ method). Moreover, the deinterleaving cost is not here offset by the lower number of operations induced by the real matrix products.

In the low-frequency regime (see figures 6a,6b,6c), the deinterleaving still allows faster M2M evaluations than the original t method with performance gains up to 4x. Since we target surface particle distributions for BIE problems, we choose to rely on the $t+s+r$ method in *defmm* for M2M evaluations, as well as for L2L ones (for which the same performance gains have been obtained).

5.3 Vectorization for direct computation

The direct computation involved in the P2P operator consists in two nested loops over the (possibly distinct) target and source particles. As usually performed in FMM codes, we aim at vectorizing this computation to benefit from the AVX2 or AVX-512 units. We target here the vectorization of the outer loop (on target particles), which leads to fewer reduction operations than with an inner loop vectorization. Based on the data locality among particles described in section 4.3.1, our P2P code is close to the one presented in [1]. As in [1], we load separately the real and imaginary parts of the complex numbers and we perform only real (not complex) mathematical operations: this is required for our compiler (Intel C++ compiler) to fully vectorize the code. There are nevertheless some differences between our code and the one presented in [1]:

- Since the inputs of our code are the particle point clouds, we have to be able to numerically handle the practical case of interactions between equally located particles on which the kernel function cannot be evaluated. These singularities (i.e. the interactions between a particle and itself) are resolved directly: if the distance between the particles is too small, we set one operand to 0.0 and we continue the interaction computation, which hence results in 0.0. This minimal test may be processed with masks in the vectorized code, hence leading to a minimal performance penalty.

- In order to vectorize our outer loop, an OpenMP compiler directive is required for the Intel C++ compiler (which targets the inner loop otherwise). It can be noticed that the GNU C++ compiler currently fails to vectorize our code (with or without the OpenMP directive).

- Since the particle positions are not updated from one FMM application to the next in BIE problems, we can rely on a structure-of-array data layout for the potentials and the charges, which favors efficient vector memory accesses.

In the end, using 64-bit floating-point numbers on 512-bit AVX-512 units (hence a maximum speedup of 8×), our vectorized P2P operator offers performance gains up to 7.6× over the original scalar implementation (see [12] for details).

6 Numerical results

We now provide numerical results illustrating the convergence and the performance of *defmm*. All codes run sequentially on a Intel Xeon Gold 6152 CPU with AVX-512 units and 384 GB of RAM.

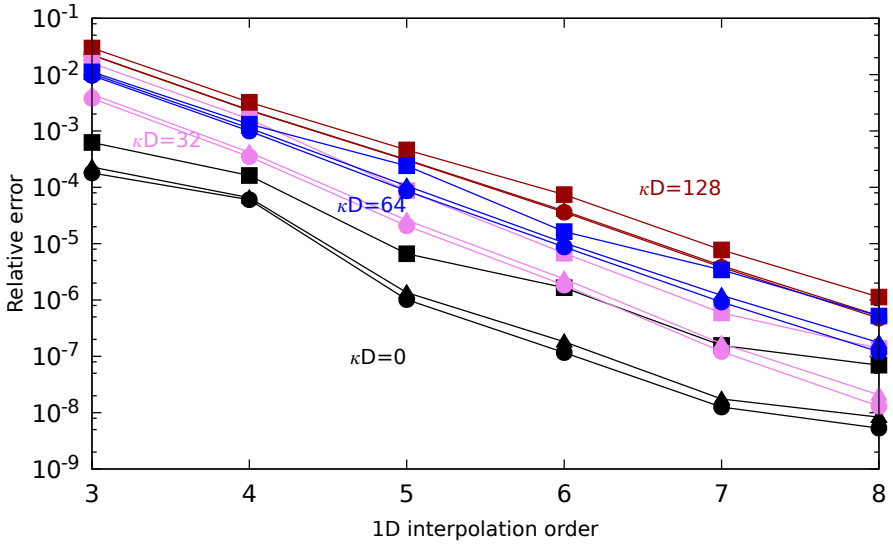


Figure 7: Relative error of the FMM over a 3D uniform cube with 1.25×10^5 particles. Maximum norm: squares; l^1 norm: circles; l^2 norm: triangles.

6.1 Relative error

We first check the overall accuracy of *defmm* on a (volumic) uniform cube test case composed of 125000 particles. We compute the error of a *defmm* approximation $\tilde{p}$ of p (see equation (1)) for a norm $||\cdot||$ as $\frac{||p-\tilde{p}||}{||p||}$. The charges are randomly chosen.

As shown in figure 7, the convergence follows the estimate derived from the interpolation process on equispaced grids in section 3.2, i.e. a geometric convergence in the 1D interpolation order. However, the same interpolation order leads to lower accuracies in the high-frequency regime than in the low-frequency one: this is due to the partitioning of the direction set, which introduces additional approximations in the high-frequency regime compared to the non-directional interpolation in the low-frequency regime. We believe that this is a consequence of the small initial set of directions chosen for performance reasons in the *defmm* implementation (see section 4.2.2) and that increasing the number of possible directions at each tree level should allow to recover better accuracies. Similar results were obtained on a surface sphere distribution.

6.2 Performance comparison with *dfmm*

For this performance comparison, we consider the following test cases, all with 10^7 particles ($\pm 1\%$).

- The *uniform cube* test case (volumic distribution). This classical FMM test case is not typical for BIE problems, but is still interesting due to its high computation cost for directional FMMs (the number of effective expansions being maximum).
- The classical *sphere* test case where the particles are quasi-uniformly scattered on the surface of the unit sphere. This is also considered as a difficult case, with high memory requirements and high computation costs, for the directional methods on boundary particle distributions [15].

- The *refined cube*, where the particles are highly non-uniformly sampled along the surface of the unit cube with a higher concentration around the edges and corners. This mimics distributions obtained with mesh refinement.
- The *ellipse*, which is an elongated distribution favorable to directional FMMs due to the reduced number of directions in the high-frequency regime. Our ellipse distribution presents a higher concentration of particles on the two poles.

defmm is compiled using the Intel C++ compiler. All the BLAS calls use the Intel Math Kernel Library (MKL) and we rely on the FFTW3 library for the FFTs.

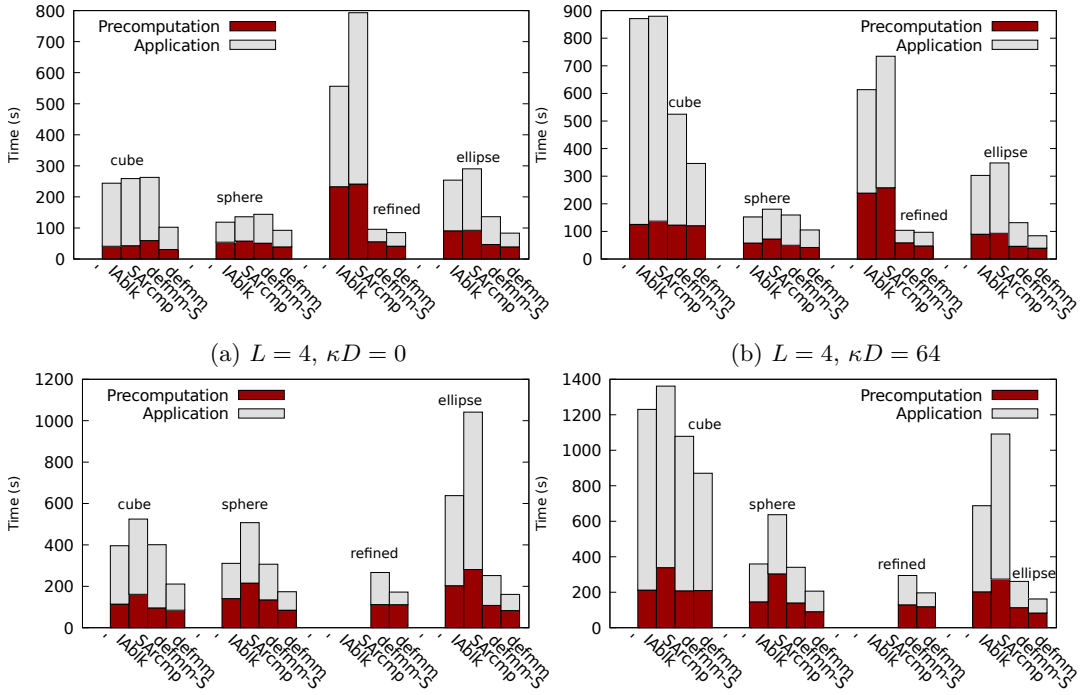
Regarding *dfmm* (only `g++` supported), we consider its two best variants [24], both based on low-rank compressions and on interpolations using tensorized Chebyshev grids. The first variant (*IAblk* – Individual Approximation with symmetries and BLocKing) relies on vector stacking and symmetries to improve performance with level-3 BLAS routines. The second one (*SArcmp* – Single Approximation with ReCoMPression) uses global approximations of the M2L matrices and recompressions to reduce the numerical ranks of these matrices. According to [24], this can lead to faster M2L evaluations than with *IAblk*, but at the cost of longer precomputation times.

Following the methodology used in [25], we fix the threshold in the low-rank approximations of *dfmm* to 10^{-L}. We checked that the accuracy of the Chebyshev interpolation in *dfmm* is then similar to the accuracy of the equispaced interpolation in *defmm*. For each test, the $MaxDepth$ (*dfmm*) and the $Ncrit$ (*defmm*) values (see section 4.3.1) are tuned to minimize the FMM application time. Due to memory constraints, we were not able to run *dfmm* (with *IAblk* or *SArcmp*) on the refined cube for $L = 6$. The comparison results are given in figure 8.

Overall performance Considering the overall times (including precomputation and application times), *defmm* always outperforms *dfmm*, being 1.3× to 6.5× faster than its best variant (*IAblk* here). As detailed below, this is due to the different design of the two codes. Since the P2P operator is not vectorized in *dfmm*, we also present in figure 8 a *defmm* variant, denoted by *defmm-S*, with a scalar (i.e. non-vectorized) P2P implementation. This enables us to more precisely study the impact of the other differences between *defmm* and *dfmm*. *defmm-S* offers similar or better performance than the best *dfmm* variant on the uniform cube and on the sphere, and outperforms *dfmm* by factors 1.8× to 5.9× on more non-uniform distributions (i.e. the refined cube and the ellipse). This shows that regarding directional FMMs our FFT-based approach is competitive with or faster than the low-rank approximations used in *dfmm*.

Sensitivity to the particle distribution Since the distributions have the same number of particles (±1%), comparing two distributions (all other test parameters being identical) illustrates the sensitivity of the method to the particle distribution. Except for the uniform cube in the high-frequency regime (whose higher cost is due to its maximum number of effective directional expansions), the *defmm* performance is few sensitive to the distribution, contrary to the *dfmm* one. This is due to the differences in the tree construction and traversal (see section 4.3): the combination of the $Ncrit$ strategy and of our specific DTT allows to better adapt in *defmm* to the particle distribution than the $MaxDepth$ strategy used in *dfmm*. Besides, the performance gap between *IAblk* and *defmm-S* is minimized for the (uniform) cube and the sphere (quasi-uniform on its surface) distributions. These two distributions benefit indeed most from the BLAS routines and from the vector stacking in *IAblk*.

Sensitivity to the wavenumber Since both codes are based on the same directional approach, their performance is similarly impacted by an increase in the wavenumber. Again, the uniform cube



(a) $L = 4$, $\kappa D = 0$

(b) $L = 4$, $\kappa D = 64$

(c) $L = 6$, $\kappa D = 0$

(d) $L = 6$, $\kappa D = 64$

Figure 8: Timings for one FMM execution of the *dfmm* and *defmm* variants. The two tested variants of *dfmm* are *IAblk* and *SArcmp*. *defmm-S* corresponds to *defmm* without P2P vectorization.

in the high-frequency regime is a special case (maximizing the number of effective expansions), where the *dfmm* features (level-3 BLAS routines, cache management) lower here the performance gap with *defmm* when the wavenumber increases: *dfmm* is however still outperformed by *defmm* in this case.

Sensitivity to the interpolation order When moving from $L = 4$ to $L = 6$, the average performance ratio of *defmm-S* over *SArcmp* moves from 1.58× to 2.40× (for the uniform cube, the sphere and the ellipse test cases, and for any frequency). This increasing performance gain with respect to the interpolation order is due to our FFT-based FMM. With respect to *IAblk*, the performance gain of *defmm-S* moves from 1.42× to 1.56×: here the FFT gain is counterbalanced by the BLAS performance gain (which is greater for larger matrices).

Precomputation costs Thanks to our extension of the symmetry usage to the Fourier domain (see section 4.4), we manage to have on the uniform cube and on the sphere precomputation costs in *defmm* as low as the *dfmm* ones (considering the *IAblk* variant which requires shorter precomputations than *SArcmp*). In addition, in highly non-uniform distributions such as the refined cube and the ellipse, the precomputation costs of *defmm* are drastically lower than the *dfmm* ones, partly thanks to our $Ncrit$-based 2^d-trees and to our blank passes (see section 4.3) which minimize the required precomputations.

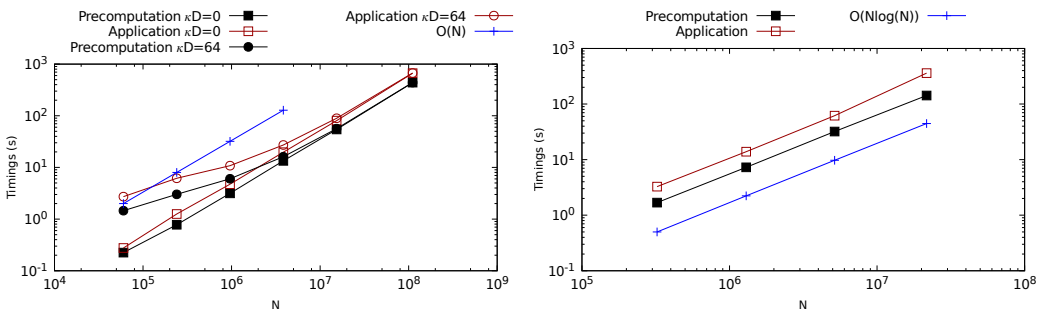


Figure 9: Timings of precomputations and applications of *defmm* on the sphere test case with respect to number of particles at fixed wavenumber κ (left), and at constant number of particles per wavelength (right).

6.3 Specific study of the sphere test case

In this section, we further detail performance results on the sphere test case, regarding various parameters.

6.3.1 Scalings

In Fig. 9, we provide timings results obtained with *defmm* on the sphere with respect to the number of particles in two situations:

1. using a fixed wavenumber, hence only increasing the number of particles without impacting the frequency;

2. using a constant (on average) number of particles per wavelength (i.e. increasing the wavenumber with the number of particles).

In the second situation (see Fig. 9 (right)), both the precomputation and the application timings scale as $\mathcal{O}(N log\, N)$, which is in accordance with the theory. In the first situation (see Fig. 9 (left)), the results indicate that adding particles without increasing the wavenumber results in adding low-frequency cells in our trees. Since our method scales as $\mathcal{O}(N)$ in the low-frequency regime (as confirmed by the $K = 0$ case), we thus asymptotically recover this scaling even for non-zero but fixed wavenumbers (such as $K = 64$). The specific shape of the scaling curves for $K = 64$ is merely due to the lower cost of processing low-frequency cells compared to the high-frequency ones.

6.3.2 Comparison with another directional FMM

Finally, we provide an additional comparison with the kernel-independent directional FMM from [15]. This FMM, based on KIFMM [32] and referred to as *ki-dfmm* here, exploits QR-based randomized low-rank approximations rather than polynomial interpolations to process the far-field contributions. Using *defmm*, we have reproduced in Table 1 the same performance test as in [15], considering their most difficult test case (i.e. the sphere) on the exact same particle distributions. Direct timings were estimated using the same methodology as in [15]. All tests have been executed in sequential, but since there is at least a 10-year gap between the CPU from [15] (published in 2007) and our Intel Xeon Gold 6152 processor (released in 2017), we focus in the comparison on the performance ratios between direct and FMM computation times, rather than on the times in seconds.

$\left(\frac{K}{2\pi}, N, \epsilon\right)$	T_a^{defmm}	T_d	A^{defmm}	$T_a^{ki-dfmm}$	T_d	$A^{ki-dfmm}$
(16,3.22e+5,1e-4)	1.58e+1	7.89e+3	5.00e+2	*9.50e+1*	*1.21e+4*	*1.28e+2*
(32,1.29e+6,1e-4)	7.73e+1	1.28e+5	1.66e+3	*4.28e+2*	*1.95e+5*	*4.55e+2*
(64,5.15e+6,1e-4)	4.01e+2	2.01e+6	5.03e+3	*1.97e+3*	*3.04e+6*	*1.54e+3*
(16,3.22e+5,1e-6)	4.36e+1	7.85e+3	1.80e+2	*2.42e+2*	*1.18e+4*	*4.86e+1*
(32,1.29e+6,1e-6)	2.38e+2	1.28e+5	5.37e+2	*1.21e+3*	*1.87e+5*	*1.54e+2*
(64,5.15e+6,1e-6)	-	-	-	*5.95e+3*	*3.13e+6*	*5.27e+2*
(32,3.22e+5,1e-8)	9.31e+1	7.89e+3	8.47e+2	*5.11e+2*	*1.22e+4*	*2.39e+1*
(32,1.29e+6,1e-8)	-	-	-	*2.62e+3*	*1.96e+5*	*7.51e+1*
(64,5.15e+6,1e-8)	-	-	-	*1.25e+4*	*3.15e+6*	*2.52e+2*

Table 1: Timings (in seconds) for estimated direct computation (T_d) and accelerated (FMM) application (T_a). A denotes the ratio of the direct computation time over the accelerated computation time. Perfomance numbers given in *italic* are extracted from [15].

One can first notice that, due to memory constraints, *defmm* fails to run tests requiring a high precision. Indeed, the memory consumption of our FFT-based approach is far more important than the *ki-dfmm* one, especially on the sphere test case which implies a high number of directional expansions per cell. However, when the *defmm* execution succeeds, *defmm* offers each time a higher performance ratio (over the direct computation) than *ki-dfmm*, which confirms the interest of our approach.

7 Conclusion

In this paper we presented a new approach for the directional interpolation-based FMMs, using FFT techniques thanks to equispaced grids. We provided a consistency proof of the approximation process and showed how to extend the symmetries of the interpolation-based FMM to the Fourier domain. We detailed the algorithmic design of our *defmm* library, as well as its high-performance optimizations on one CPU core. Finally, comparisons with two *state-of-the-art* libraries exhibited the superior performance of our library in all test cases (without memory constraints), the *defmm* performance being also few sensitive to the surface particle distribution. This shows the interest of our FFT-based approach for lowering the FMM cost in the high frequency regime, and for efficiently processing the non-uniform distributions of the BIE problems.

Future works will be dedicated to the *defmm* parallelization. In this purpose, we will be able to lean on the dual tree traversal, which is known to be highly efficient regarding shared-memory parallelism (see e.g. [1, 33]), and on previous work (such as [4]) regarding distributed-memory parallelism. We also plan to integrate *defmm* in an iterative solver in order to solve complex realistic boundary integral equations.

References

[1] M. Abduljabbar, M. A. Farhan, N. Al-Harthi, R. Chen, R. Yokota, H. Bagci, and D. Keyes, *Extreme scale fmm-accelerated boundary integral equation solver for wave scattering*, SIAM Journal on Scientific Computing, 41 (2019), pp. C245–C268.

[2] E. Agullo, B. Bramas, O. Coulaud, E. Darve, M. Messner, and T. Takahashi, *Pipelining the Fast Multipole Method over a Runtime System*, Research Report RR-7981, INRIA, May 2012. https://hal.inria.fr/hal-00703130/file/RR-7981.pdf.

[3] L. V. Ahlfors, *Complex analysis: An introduction to the theory of analytic functions of one complex variable; 3rd ed.*, (1978), pp. xi+331.

[4] A. R. Benson, J. Poulson, K. Tran, B. Engquist, and L. Ying, *A parallel directional fast multipole method*, SIAM J. Sci. Comput., 36 (2014), p. C335–C352.

[5] P. Blanchard, O. Coulaud, and E. Darve, *Fast hierarchical algorithms for generating Gaussian random fields*, Research Report 8811, Inria Bordeaux Sud-Ouest, Dec. 2015.

[6] A. Brandt, *Multilevel computations of integral transforms and particle interactions with oscillatory kernels*, Comput. Phys. Comm., 65 (1991), pp. 24–38.

[7] S. Börm, *Directional H2-matrix compression for high-frequency problems*, Numer. Linear Algebra Appl., 24 (2017).

[8] S. Börm and L. Grasedyck, *Low-Rank Approximation of Integral Operators by Interpolation*, Computing, 72 (2004), pp. 325–332.

[9] C. Cecka and E. Darve, *Fourier-based Fast Multipole Method for the Helmholtz equation*, SIAM J. Sci. Comput., 35 (2013), p. A79–A103.

[10] C. Chen, S. Aubry, T. Oppelstrup, A. Arsenlis, and E. Darve, *Fast algorithms for evaluating the stress field of dislocation lines in anisotropic elastic media*, Modelling and Simulation in Materials Science and Engineering, 26 (2018), p. 045007.

[11] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. F. 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*, J. Comput. Phys., 216 (2006), p. 300–325.

[12] I. Chollet, *Symmetries and Fast Multipole Methods for Oscillatory Kernels*, Ph.D. thesis, Sorbonne Université, Mar. 2021, `https://tel.archives-ouvertes.fr/tel-03203231`.

[13] O. Coulaud, P. Fortin, and J. Roman, *High-performance BLAS formulation of the adaptive Fast Multipole Method*, Mathematical and Computer Modelling, 51 (2010), pp. 177–188.

[14] W. Dehnen, *A hierarchical O(N) force calculation algorithm*, J. Comput. Phys., 179 (2002), p. 27–42.

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

[16] W. Fong and E. Darve, *The black-box fast multipole method*, J. Comput. Phys., 228 (2009), pp. 8712–8725.

[17] P. Fortin, E. Athanassoula, and J.-C. Lambert, *Comparisons of different codes for galactic N-body simulations*, Astronomy and Astrophysics - A&A, 531 (2011), p. A120.

[18] M. Frigo and S. G. Johnson, *The design and implementation of FFTW3*, Proceedings of the IEEE, 93 (2005), pp. 216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[19] M. Gasca and T. Sauer, *Polynomial interpolation in several variables*, Advances in Computational Mathematics, 12 (2000), pp. 377–410.

[20] K. Gibermann, *Multilevel Approximation of Boundary Integral Operators*, Computing, 67 (2001), p. 183–207.

[21] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[22] B. Hariharan, S. Aluru, and B. Shanker, *A Scalable Parallel Fast Multipole Method for Analysis of Scattering from Perfect Electrically Conducting Surfaces*, in SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, 2002, pp. 42–42.

[23] D. Malhotra and G. Biros, *Algorithm 967: A Distributed-Memory Fast Multipole Method for Volume Potentials*, ACM Trans. Math. Software, 43 (2016).

[24] M. Messner, B. Bramas, O. Coulaud, and E. Darve, *Optimized M2L kernels for the Chebyshev interpolation based fast multipole method*, (2012). https://arxiv.org/abs/1210.7292.

[25] M. Messner, M. Schanz, and E. Darve, *Fast directional multilevel summation for oscillatory kernels based on Chebyshev interpolation*, J. Comput. Phys., 231 (2012), p. 1175–1196.

[26] B. Mössner and U. Reif, *Error bounds for polynomial tensor product interpolation*, Computing, 86 (2009), pp. 185–197.

[27] K. Nabors, F. T. Korsmeyer, F. T. Leighton, and J. White, *Preconditioned, Adaptive, Multipole-Accelerated Iterative Methods for Three-Dimensional First-Kind Integral Equations of Potential Theory*, SIAM J. Sci. Comput., 15 (1994), p. 713–735.

[28] V. Pavlov, N. Andonov, and G. Kremenliev, *Porting and Verification of ExaFMM Library in MIC Architecture*, (2014). https://doi.org/10.5281/zenodo.822725.

[29] R. B. Platte, L. N. Trefethen, and A. B. J. Kuijlaars, *Impossibility of fast stable approximation of analytic functions from equispaced samples*, SIAM Review, 53 (2011), pp. 308–318.

[30] D. Schobert and T. Eibert, *A multilevel interpolating fast integral solver with fast Fourier Transform acceleration*, Symposium Digest - 20th URSI International Symposium on Electromagnetic Theory, EMTS 2010, (2010), pp. 520 – 523.

[31] S. Smith, *Lebesgue constants in polynomial interpolation*, Annales Mathematicae et Informaticae, 33 (2006), pp. 109–123.

[32] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196 (2004), p. 591–626.

[33] R. Yokota, *An FMM based on dual tree traversal for many-core architectures*, Journal of Algorithms & Computational Technology, 7 (2012).