Symmetries and Fast Multipole Methods for Oscillatory Kernels

Igor Chollet

Sorbonne Université

École doctorale **Sciences Mathématiques de Paris Centre (ED386)**

Laboratoire de recherche **Institut des Sciences du Calcul et des Données**

Symétries et méthodes multipolaires rapides pour noyaux oscillants

Par **Igor Chollet**

Thèse de doctorat de **Mathématiques appliquées**

Dirigée par Xavier Claeys, Pierre Fortin, Laura Grigori

Présentée et soutenue publiquement le 24 mars 2021

Devant un jury composé de

Rapporteurs	Eric Darrigrand	Maître de conférence à Université de Rennes 1
	Eric Darve	Professeur à Stanford University
Examinateurs	Pascal Frey	Professeur à Sorbonne Université
	Gunnar Martinsson	Professeur à University of Texas
	Raymond Namyst	Professeur à Université de Bordeaux
Directeurs de thèse	Xavier Claeys	Maître de conférence à Sorbonne Université
	Pierre Fortin	Maître de conférence à Université de Lille
	Laura Grigori	Directrice de recherche à Inria Paris

Sorbonne Université

Doctoral school **Sciences Mathématiques de Paris Centre (ED386)**

University department **Institut des Sciences du Calcul et des Données**

Symmetries and Fast Multipole Methods for Oscillatory Kernels

By **Igor Chollet**

Academic field: **Applied Mathematics**

Supervised by Xavier Claeys, Pierre Fortin, Laura Grigori

Defended on March 24^{th} 2021

In the presence of a jury composed of

Referees	Eric Darrigrand	Associate professor at Université de Rennes 1
	Eric Darve	Professor at Stanford University
Examiners	Pascal Frey	Professor at Sorbonne Université
	Gunnar Martinsson	Professor at University of Texas
	Raymond Namyst	Professor at Université de Bordeaux
Supervisors	Xavier Claeys	Associate professor at Sorbonne Université
	Pierre Fortin	Associate professor at Université de Lille
	Laura Grigori	Director of research at Inria Paris

Remerciements

Au moment de me plier à l'exercice des remerciements, c'est moins la peur d'oublier des noms (ce sera forcément le cas) que celle de ne pas suffisamment exprimer ma gratitude envers ceux cités qui m'angoisse.

Mes premiers remerciements vont à Xavier Claeys, Pierre Fortin et Laura Grigori pour m'avoir accordé leur confiance le temps de cette thèse, supporté (et cadré !) ma non-négligeable propention au dispersement et m'avoir offert l'opportunité de me plonger avec eux dans ce passionnant domaine du calcul scientifique. Pour la liberté qu'ils m'ont donnée sur les sujets traités dans ce manuscrit, pour toutes les discussions durant lesquelles ils ont partagé leur expérience, pour les relectures compliquées qu'ils ont eues à subir, mais endurées tout de même, pour leur soutien bienveillant et leurs retours, pour tout cela je leur suis infiniment reconnaissant. J'ai conscience de la chance que j'ai eue de pouvoir compter sur leur disponibilité et leurs conseils. Si j'ai pu garder l'entrain des premiers jours tout le long de cette thèse, c'est en grande partie grâce à eux.

Je tiens également à remercier les rapporteurs, Eric Darrigrand et Eric Darve, pour m'avoir fait l'honneur d'accepter de prendre sur leur temps précieux celui d'étudier ce manuscrit, ainsi que Pascal Frey, Gunnar Martinsson et Raymond Namyst pour avoir accepté d'être membre de mon jury.

Un grand merci à Francis Collino pour avoir partagé son expertise des méthodes multipolaires pour le noyau de Helmholtz ainsi que pour ses suggestions importantes et pertinentes qui ont inspiré une part des développements de ce manuscrit.

Pour m'avoir donné le goût de la recherche et amené sur des terrains scientifiques que je rêve de parcourir à nouveau, je n'oublie pas de remercier Laurent Boudin, Mohab Safey El Din et Nicolas Vauchelet.

Il me faut à présent adresser quelques lignes aux différentes têtes ayant traversé l'excentré 15-16-221, aux membres de l'équipe Inria Alpines dans laquelle j'ai eu la chance de pouvoir évoluer durant ces quelques années ainsi qu'à mes collègues du CEMRACS 2018. D'une façon ou d'une autre, vous avez rendu plus agréable ce travail de thèse. Merci à Pierre pour ces discussions agréables et ces inombrables conseils (promis : je n'oublie pas htool !), à Hugo pour les parties de Star Realms à 20h avant de quitter le bureau, à Frédérique, Giulia, Théo, Philippe, Thomas, Laurent... Quelques mots, enfin, pour Bertrand, pour ces passionnantes discussions autour des littératures de l'imaginaire, qui me manquent indubitablement déjà.

Une pensée toute particulière pour Lise, au sujet de qui les remerciements devraient noircir plus de pages que je ne peux me l'autoriser ici. Abordons simplement ton indéfectible soutien moral, tes remontrances bienvenues. Ce que je te dois dans l'achèvement de ce manuscrit, plus globalement de ce travail, ne se résumera pas en quelques lignes, aussi j'espère tout de même transmettre à travers ce court paragraphe une part de ma gratitude : merci pour tout.

J'ai aussi une pensée nostalgique pour Clémence, Laurent, Mouna et Serge, pour ce quintet qui n'aura malheureusement pas survécu jusqu'à la soutenance : la musique que nous jouions aura été une échappatoire remarquable tout le temps où j'ai pu m'y abandonner. Il aurait été agréable, je m'en rends compte, de plaquer les accords d'un générique de fin lors d'une répétition comme celles dont j'ai et garderai le souvenir. Une pensée encore pour mon ami Bruno, sans qui la guitare et la musique n'auraient jamais été des éléments aussi importants de ma vie ni ces exutoires formidables dans lesquels j'ai aimé me perdre aux pires moments de la rédaction de ce manuscrit.

Aux ex-PIMA et aux nombreuses amitiés nouées lors de ces deux années, dont nombre perdurent aujourd'hui encore.

Une ligne toute entière, évidemment, pour Nicolas, parce que c'est correct.

Une pensée, enfin, pour ma mère et sa patience infinie; une dernière, encore, pour Flore et l'apaisement que tu m'as apporté.

Contents

Chapter 1

Introduction

1.1 Context

The numerical simulation of complex physical phenomena still nowadays challenges the limits of modern computer capabilities. Indeed, many mathematical models describing these phenomena rely on partial differential equations and boundary value problems, whose explicit solutions often have to be numerically approximated. This leads to the numerical inversion of linear systems. Depending on the problem and on the requested solution accuracy, the solving may require massive computations.

In practice, the matrices involved in these linear systems may verify certain properties. When dealing with boundary integral equations which correspond to a boundary integral form of a given boundary value problem, the product by these matrices has a strong relation with the N-body problems. A N-body problem corresponds to the task of computing all pairwise interactions between two sets of particles according to a physical law. Their complexity is quadratic, which quickly becomes prohibitive when the problem size increases. The boundary integral equations being formulated on the boundaries of a domain, the particles coming from the discretization of these equations are distributed on surfaces (see Fig. 1.1). This somehow reduces the dimension of the studied problem, transforming a volumic formulation into a surfacic one. Another important feature of the integral equations is their ability to deal with unbounded domains.

During the mid-80's has emerged a family of methods, said to be hierarchical and aiming at reducing the complexity of solving N-body problems [23, 30, 115]. One of the most popular among them [85] is the Fast Multipole Method (FMM) [115], that paved the way to the evaluation of particular N-body problems with linear complexity, but at the cost of the introduction of a controllable error in the result. The hierarchical aspects of the FMM algorithms are related to the data structures used to represent the particle distributions (i.e. the particle point clouds, referred to as source and target particle distributions). These structures are tree representations of space referred to as 2^d-trees (binary trees in 1D, quadtrees in 2D and octrees in 3D). For a given target particle, these 2^d-trees permit to separate a *near* and a *far* fields, corresponding to a partition of the source particle distribution in two domains. Hence, the interactions between the target particle $\mathbf{x}$ and the source particles in the near field of $\mathbf{x}$ are computed directly and the interactions with the source particles in the far field are approximated.

One of the reasons of the FMM efficiency is its multilevel structure: through various traversals of the 2^d-tree structures (see Fig. 1.2), multilevel approximations of the far field contributions can be obtained and efficiently evaluated. These multilevel representations of the far field contributions exist both for the source and the target particles. By using the links between the different levels, fast reconstructions of the far field influence may be derived.

The effective realization of the 2^d-tree structure leads to different traversal algorithms depending on the construction strategy. These traversals are a main component of the FMM algorithm since

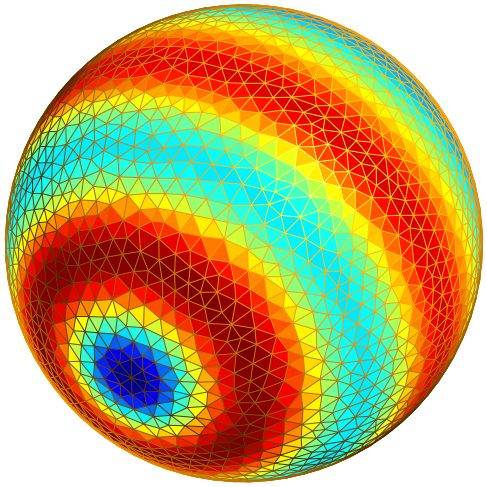


Figure 1.1: Result of a wave scattering simulation using boundary integral equations in an unbounded exterior domain with spherical interior boundaries.

they are directly related to the complexity estimates. There are two main kinds of FMM to consider: the uniform algorithms mainly designed for uniform particle distributions and the adaptive algorithms aiming at dealing with any particle distribution. These two approaches usually rely on different data structures.

Nevertheless, strong difficulties appear when one considers oscillatory problems, such as those appearing in acoustics or electromagnetics. The explicit approximation of oscillatory kernels in the high-frequency regime needs specific developments to be efficiently applied. For the FMM designed to evaluate the Helmholtz kernel in the high-frequency regime (i.e. corresponding to highly oscillatory problems), the far field approximations lead to the explicit diagonalization of large matrices [184]. The multilevel FMM algorithm is obtained in this context through interpolations of the data for each 2^d-tree node. As opposed to the hierarchical methods for non-oscillatory problems, the application costs of these diagonal matrices and of these interpolations increase from the leaves to the root of the 2^d-tree, impacting the complexity of the overall method. There is thus a need for optimizing these steps. In practice, specific integral discretizations over the sphere allow to achieve a fast evaluation of the diagonal matrices while keeping fast evaluations of the data interpolations through Fast Fourier Transform (FFT) applications.

On the other side, many FMM formulations are nowadays designed for any non-oscillatory N-body problems. These methods are said to be *kernel-independent*. The multivariate polynomial interpolation is an example of the techniques that can be used to derive kernel-independent FMM formulations [103]. When such formulation is directly applied to oscillatory problems, their performances deteriorate with the increase of the computational domain size in terms of wavelength. In practical engineering applications, these domains can be several hundreds or thousands wavelength wide. Fortunately, combined with a *directional approach* [94, 174] that consists in modifying the far field shape, compared to the non-directional methods, to take into account the oscillatory behavior of the approximated functions, these kernel-independent methods can be extended to oscillatory problems in the high-frequency regime. An interesting property of these approaches is that they preserve the efficiency of the formulation in the low-frequency regime.

The polynomial interpolation-based FMMs rely on additional fast algorithms for the far field

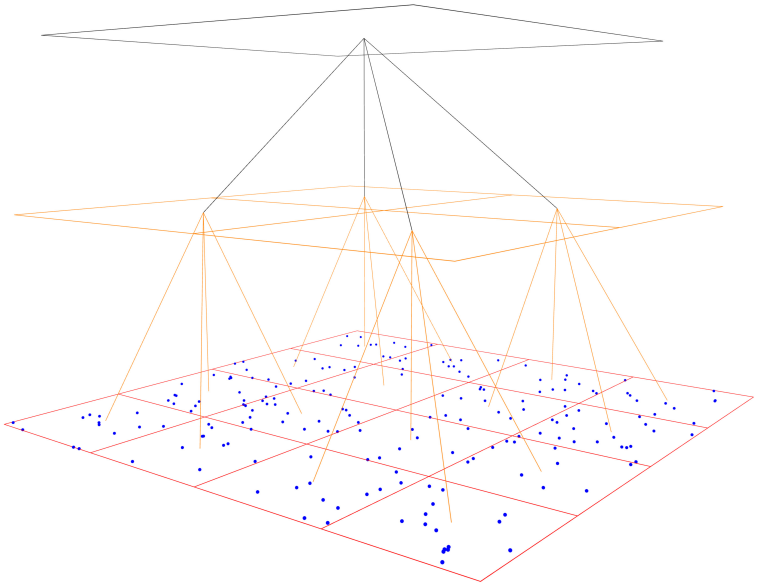


Figure 1.2: Three level quadtree representation of a particle distribution (blue dots).

contribution evaluations. These algorithms are often based on low-rank approximations. However, other options have emerged in the literature, such as FFT-based techniques exploiting the property of cartesian interpolation grids (i.e. composed of equispaced nodes along each spatial axis) and the properties of the interpolated functions. These last techniques were still not investigated in a directional context.

1.2 Present work

The work presented in this manuscript focuses on the hierarchical methods for oscillatory kernels arising in acoustics or electromagnetics with a particular interest for the high-frequency regime. The main objective is to include a hierarchical method able to efficiently deal with oscillatory kernels into a solver for integral equations. There is an important constraint to take into account to produce an efficient code: the particle distributions we deal with are non-uniform, corresponding to point clouds over surfaces. Hence, we want to end up with a method able to handle this (possibly high) non-uniformity. Indeed, the particle distribution may result in a performance bottleneck on meshes with local density variations, as resulting from mesh refinements, if the non-uniformity is neglected. In the same time, in realistic applications, local parts of the particle distributions may prevent the use of hierarchical methods designed for the high-frequency regime only. We thus want to implement an efficient wideband (i.e. able to treat both the low- and high-frequency regimes) method that can efficiently handle highly non-uniform distributions.

We are strongly concerned by the implementation of such method on modern architectures for high performance computing (HPC). Hence, we explored the state-of-the-art optimizations of the hierarchical methods we opted for. There exists numerous methods, with corresponding case-specific optimized implementations. This lead us to the design of a high-level approach for the symmetry-based optimizations exploited in many FMM codes. In a more general way, we aim at presenting general tools to work on the hierarchical methods regardless to the specificities of each. The applications of this study however mainly concerns the FMMs. Indeed, we are mainly interested in these last methods because they offer the best complexities when dealing with oscillatory kernels in the high-frequency regime, compared to other hierarchical methods.

This work explores various FMM formulations, providing various optimizations or extensions in mainly two of them. Most of our ideas are based on the same symmetries (namely, those associated to the hyperoctahedral group) but in different contexts. Each application of these symmetries leads to different algorithms. Hence, we show to what extent the symmetries can be exploited in different FMM formulations.

Last but not least, we present the entire realization of a *C++* FMM library, named *defmm* for Directional Equispaced interpolation-based Fast Multipole Method, discussing the theoretical aspects and the algorithmic design as well as several numerical optimizations for modern HPC architectures. Our library aims at combining several efficient approaches in the literature that have never been associated. Namely, we apply the FFT techniques for a polynomial interpolation-based FMM in a directional context in order to efficiently deal with the wideband aspects of our problems while handling the non-uniform distributions through an adaptive 2^d-tree construction and using relevant tree traversal algorithms. Numerous numerical results on one CPU core illustrate the sequential performances of our library, depending on general criteria such as the computational domain size in terms of wavelength or the uniformity of the particle distribution, including a numerical comparison with a state-of-the-art FMM library.

1.3 Contributions

Several contributions are introduced in this thesis. We give here a quick overview of these.

High-level approach of the hierarchical methods. In this manuscript, we deal with different FMM formulations. The need of handling the general aspects of the FMM and of the other standard hierarchical methods (such as the treecodes or the $\mathcal{H}$-matrices) led us to the design of a mathematical unified framework. We use this framework to express any FMM formulation as an explicit matrix factorization.

Structure of the FMM. We propose a high-level approach regarding the symmetries with FMM formulations on 2^d-tree. This allows to express a set of abstract results on the structure of the FMM factorization matrices regardless of the FMM formulation. Some of these results are already used in practical implementations of different FMM formulations for case specific optimizations, justifying this generalization.

Lebedev rules in *hf-fmm*. As a first realization of the results provided in our high-level description of the structure of the FMM on 2^d-trees, we propose to handle the cubature of the propagating planewave expansion in *hf-fmm* (a kernel-explicit FMM for highly oscillatory problems) using the Lebedev cubature rules. Compared to standard cubature choices, the use of Lebedev rules reduces the computation of far field interactions to a quasi-optimal one and lowers the memory footprint while extending the method to the entire possible set of 2^d-tree symmetries.

However, such application of the Lebedev rules to *hf-fmm* strongly complicates the polynomial interpolation problem over the sphere arising in this context. We propose a new purely algebraic approach to deal with this problem thanks to group-invariant cubature grids. Our method applies to the usual choices of cubature rules, somehow extending the FFT-based fast algorithms for the interpolation over the sphere to a large class of operators. In the same time, this allows to block-diagonalize the interpolation operator when using Lebedev rules, accelerating its application. Our methodology also applies to the other quasi-optimal cubature rules over the sphere, with potential gain depending on their exact structure.

We propose an implementation exploiting the matrix-matrix products to efficiently apply the block-diagonal form of the interpolation operator on the sphere using our method. The scheme we introduce is based on the explicit data redundancies appearing in the factorization thanks to the group representation theory.

Finally, we propose a fast switching strategy allowing to benefit from the varying efficiency of the Lebedev and product cubature rules on the different FMM operators, depending on the tree level.

New directional interpolation-based FMM library. An important part of our work was dedicated to the realization of a FMM for oscillatory kernels. We present a new FMM library (*defmm*) which is:

- Kernel-independent, using polynomial interpolation techniques;
- Wideband, in the sense that both the low- and high-frequency regime can be efficiently treated thanks to a directional approach;
- Accelerated through FFTs for the far field evaluation, using polynomial interpolations on cartesian grids;
- Adaptive, i.e. able to deal with arbitrary particle distributions with a similar efficiency thanks to a new reformulation of a recursive tree traversal (the so called Dual Tree Traversal) in a directional context and to a precomputation step based on blank tree traversals.

Optimizations. Several novel optimizations of our library are proposed to further accelerate the performances. The way of applying the numerous FFTs, of evaluating the near field or the reinterpolation operator (needed in the multilevel interpolation-based FMM scheme) and the expansion storage are discussed. For these optimizations, we tried to exploit at best the architecture of one CPU core (SIMD parallelism, BLAS routines, arithmetic intensity). We also provide another reformulation of the Dual Tree Traversal algorithm exploiting extra FMM operators to better process highly non-uniform distributions.

Consistency proof of interpolation process using equispaced grids on well-separated sets. The polynomial interpolation on grids composed of equispaced nodes suffer from (at least numerical) instabilities. However, such an interpolation on cartesian grids is widely used in hierarchical methods, without stability issues in practice. Since we also base our library on interpolation on cartesian grids, we provide a consistency proof of the interpolation process on these grids under the usual assumptions of the hierarchical methods, hence justifying the practical convergence of the approximations.

Combining FFT techniques and symmetries. Our library exploits the symmetries of the FMM on 2^d-trees. We extend the symmetries already used in the polynomial interpolation-based FMM in the Fourier domain in order to combine them with the FFT techniques using cartesian interpolation grids. This leads to the explicit construction of the associated permutation representations. We propose a new way of exploiting these permutation representations in the far field evaluation, allowing to apply a single permutation instead of two. However, the far field evaluation becomes more difficult to vectorize when using these symmetries, generating Hadamard products with indirections. We thus introduce a new method based on group orbits in order to tackle this issue in a general manner and we propose complete numerical tests on it. The performances lead us to a comparison of the vectorized variants for the Hadamard products using modern compilers.

Comparison of approaches. A careful and complete comparison between our library and a state-of-the-art one, namely *dfmm* [1] is provided. This allows to discuss the different algorithmic choices, i.e. the far field evaluation or the tree construction methods. Since the directional FMMs are quite emerging but still not widely used in the literature, the numerous tests we provide also enable us to discuss the behavior of these methods according to the frequency regime and to the particle distribution.

1.4 Overview

This manuscript is composed of three parts. The first part I is dedicated to the background on hierarchical methods for N-body problems with a special focus on FMM dealing with oscillatory kernels in the high-frequency regime. The second part II of this thesis is concerned by theoretical aspects about the general FMM formulations and by abstract considerations on symmetries. This leads to applications in the particular context of methods for the high-frequency Helmholtz kernel. The third and last part III is concerned by the entire realization of a FMM for arbitrary radial oscillatory kernels.

State of the art. This first part is divided into three chapters. In Chap. 2 are presented the general notions on N-body problems and kernels. We explain how such problems appear when dealing with integral equations. Then, we present the tree structures that are the bases of the hierarchical methods and we describe the main hierarchical algorithms. Among these families, we are particularly interested in the FMM. Indeed, these methods offer efficient multilevel strategies to deal with highly oscillatory kernels, through explicit diagonalization of large matrix blocks for the *kernel-explicit* methods and through directional approaches for the *kernel-independent* approaches. The multilevel strategies usually are the key to reach the linearithmic complexities. Hence, the two other chapters in the state of the art are dedicated to the FMM for the kernels we are considering. In Chap. 3 we present the kernel-explicit method for 3D Helmholtz problems in the high-frequency regime, paying a particular attention to the interpolation problem over the sphere appearing in such a context. The main fast algorithms dealing with this problem are described. In Chap. 4, we present the kernel-independent methods based on polynomial interpolation and an efficient strategy allowing to extend these methods to oscillatory kernels, namely the directional technique. We also describe in Chap. 4 the main acceleration strategies for the evaluation of the far field interactions when using polynomial interpolation.

Symmetries in Fast Multipole Method. This second part is composed of two chapters. In Chap. 5, we are interested in a high-level matrix formulation of the hierarchical algorithms allowing to write any FMM formulation as a matrix factorization. The main other hierarchical methods described in Chap. 2 are also rewritten using the tools we introduced. We also provide a set of results on the symmetries in the FMM formulations based on a particular tree structure: the 2^d-trees. Some of these results are built on a group invariance of particular interest associated to the hyperoctahedral group. Then, in Chap. 6, we extend the kernel-explicit method described in Chap. 3 to this entire group invariance by introducing the Lebedev rules in this context. We describe a block-diagonalization method to tackle the interpolation problem on the sphere with the Lebedev rules. This approach is based on the group representation theory.

A new Fast Multipole Method library for oscillatory kernels. This third part is composed of one single chapter. In Chap. 7, we present a new FMM library (*defmm*), based on FFT techniques and on the directional polynomial interpolation approach, all presented in Chap. 4. These FFT techniques are based on polynomial interpolation of cartesian grids. We give a consistency proof of the approximation method and show how to extend the symmetries studied in Chap. 5 in the Fourier domain in this context. We then propose a group theory based permutation approach to deal with the array indirection problems appearing when numerically exploiting these symmetries in the Fourier domain. We also present several optimizations of our library. A detailed comparison between *defmm* and the other directional polynomial interpolation-based FMM is provided. Hence, in Chap. 7 are discussed the theory, the algorithmic design, the implementation and the validation of our new FMM library.

Part I

State of the art

Chapter 2

N-body problems

Contents

The computation of all pairwise interactions among N-bodies with regard to a specific kernel (describing the interactions between any two bodies) is a computationally intensive task leading to a $\mathcal{O}(N^2)$ operation count for a naive implementation. This prohibitive numerical cost quickly becomes a limit in many applications such as astrophysics, molecular dynamics or solvers for integral equations. The aim of N-body algorithms is to break this complexity.

This chapter is dedicated to the presentation of the N-body problems arising in the context of integral equations. We first define in Section 2.1 what a N-body problem refers to and describe how the integral equation framework leads to this kind of problems. This ends with an overview of the main non-hierarchical methods trying to deal with these N-body problems. Then, in Section 2.2, we present a set of hierarchical methods achieving the same goal with less drawbacks in our application context. As opposed to the non-hierarchical methods, the studied ones are based on a recursive space partitioning and are said to be hierarchical. There exist a lot of hierarchical methods and we are particularly concerned by the Fast Multipole Methods (FMMs) for which we will describe explicit mathematical tools that allow to derive efficient methods in two particular explicit cases.

2.1 Problem formulation

Among the families of numerical methods solving Partial Differential Equations (PDEs), the Boundary Element Methods (BEMs), designed to approximate the solutions of Boundary Integral Equations (BIEs), are widely used in acoustics, electromagnetics, elastodynamics and thermoelastodynamics (see [29] for a large amount of references on the applications). This family of methods reformulates a volumic problem into a surfacic one. Hence, the dimensions of the new problem being lower than the original ones, its numerical treatment generates smaller systems. However, as opposed to the usually sparse matrices generated by the Finite Element Method (FEM) [99], the BEM generates fully populated matrices. This last point can be a performance bottleneck in practice, despite the dimensional reduction. Fortunately, there exists a strong analogy between the product by these dense matrices and the N-body problems, as explained in 2.1.2. Before describing this link, we recall in Section 2.1.1 the definition of a general N-body problem and a set of kernels involved in common BIEs. In Section 2.1.3, we quickly provide a general overview of a set of non-hierarchical methods designed for N-body problems.

2.1.1 The summation problem

Let $d \in \mathbb{N}^*$ be the dimension of the problem we are looking at. Given two point clouds $X, Y \subset \mathbb{R}^d$, $q : Y \to \mathbb{C}$ and a *kernel* $G : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$, the N-body problem between X and Y with the kernel G refers to the problem of computing $p : X \to \mathbb{C}$ defined by:

$$p(\mathbf{x}) := \sum_{\mathbf{y} \in Y} G(\mathbf{x}, \mathbf{y}) q(\mathbf{y}). \tag{2.1}$$

In all this document, the elements of a point cloud X or Y are referred to as **particles**. Notice that this definition of the N-body problem for a kernel with scalar values can be extended to kernels with vector values (namely with values in $\mathbb{C}^d$).

In the BIE context, G corresponds to a Green's kernel. This means that G is the fundamental solution of a PDE. We recall here the expressions of such a G in two of the most common cases.

Kernel	**PDE**	**Expression**
Laplace	$-\Delta G(x) = \delta(x)$	$\begin{cases} \frac{1}{2\pi x} ln\,(x) & in\ 2D \\ \frac{1}{4\pi x} & in\ 3D \end{cases}$
Helmholtz	$-\Delta G(x) - \kappa^2 G(x) = \delta(x)$	$\begin{cases} \frac{i}{4} H_0^{(1)}(\kappa x) & in\ 2D \\ \frac{e^{i\kappa x}}{4\pi x} & in\ 3D \end{cases}$

where δ refers to Dirac's distribution, $H_0^{(1)}$ to the Hankel function of the first kind (see [178] Chapter 10 Paragraph 2) and $\kappa \in \mathbb{C}$ is named the **wavenumber**. The inverse of the wavenumber multiplied by 2π is named the **wavelength**. Only a single argument is needed in these kernels because of their translational and rotational invariances. Such kernels are said to be **radial**. We choose x equal to r defined by:

$$r := |\mathbf{x} - \mathbf{y}|$$

to fit with the notations of Eq. 2.1, $|\mathbf{z}| := \left(\sum_{k=1}^{d} z_k^2 \right)^{1/2}$. As opposed to smooth kernels, such as the Gaussian one given, for a fixed parameter σ, by $G(r) := exp\left(-\frac{1}{2}\left(\frac{r}{\sigma}\right)^2\right)$, the Laplace and Helmholtz kernels are singular at $r = 0$, meaning that their expression tends to $+\infty$ around this singularity.

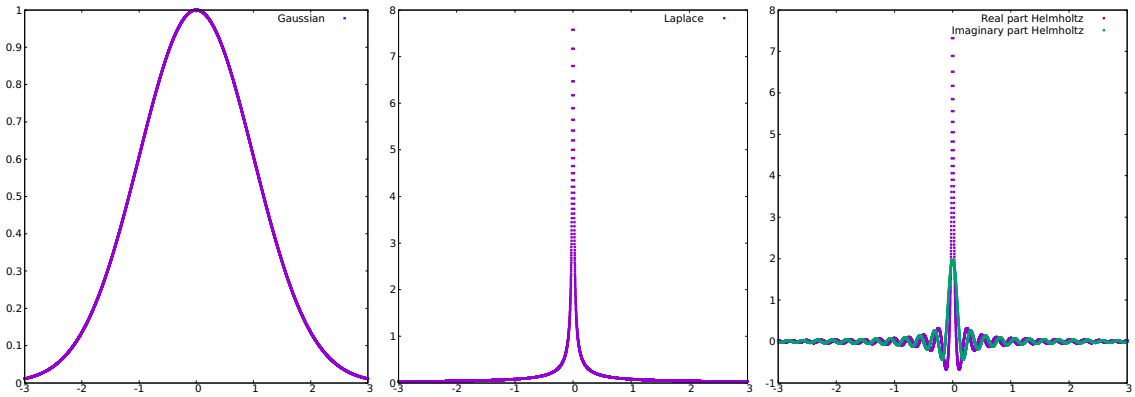


Figure 2.1: From left to right: Gaussian kernel (smooth) with $\sigma = 1$, Laplace kernel (asymptotically smooth and singular), Helmholtz kernel (asymptotically smooth, singular and oscillatory) with $\kappa = 25$.

Because such singular kernels are not defined on the singularity, a common practical choice consists in arbitrarily fixing $G(0) = 0$. Moreover, the Laplace, Helmholtz and Gaussian kernels are all *asymptotically smooth* [46, 47] and this feature allows in practice a fast approximated evaluation of Eq. 2.1.

Definition 2.1.1. *([44] Def. 4.5) Let $G : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. The function G is said to be asymptotically smooth if there exist three non-negative constants $C_{as} > 0$, $\sigma \in \mathbb{N}$ and $c_0 > 0$ such that, $\partial_{\mathbf{p}}^{\nu}$ denoting the ν^{th} directional partial derivative in direction $\mathbf{p}$ (i.e. with $\partial_{\mathbf{p}} f(\mathbf{x}) := lim_{t\to 0} \frac{f(\mathbf{x}+t\mathbf{p})-f(\mathbf{x})}{t}$),*

$$|\partial_{\mathbf{p}}^{\nu} G(\mathbf{x},\mathbf{y})| \leq \begin{cases} C_{as} \frac{(\nu+\sigma-1)!}{(\sigma-1)!} \frac{c_0^{\nu}|\mathbf{p}|^{\nu}}{|\mathbf{x}-\mathbf{y}|^{\sigma+\nu}} & \text{if } \sigma > 0 \\ C_{as}(\nu-1)! \frac{c_0^{\nu}|\mathbf{p}|^{\nu}}{|\mathbf{x}-\mathbf{y}|^{\sigma+\nu}} & \text{if } \sigma = 0 \end{cases}$$

holds for all $\nu \in \mathbb{N}^*$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $\mathbf{x} \neq \mathbf{y}$, and all directions $\mathbf{p} \in \mathbb{R}^d \times \mathbb{R}^d$.

Particular kernels, such as the Helmholtz one because of the complex exponential, have an oscillatory behavior with regard to r. The frequency of the oscillations depends on κ. These oscillations may result in numerical difficulties when trying to deal with. Notice that for a given interval, the Helmholtz kernel stops oscillating if κ is less than the interval size. In addition, the limit case $\kappa = 0$ exactly corresponds to the Laplace kernel. Representations of these kernels are given in Fig. 2.1. We mainly restrict ourselves to the kernels with scalar values, but gradients of these kernels may also be considered, leading to other kernels but with range in $\mathbb{R}^d$.

2.1.2 From integral equations to N-body problems

Suppose that we want to solve the following problem for a given domain Ω and a given differential operator $\mathfrak{L}$, $\Gamma := \partial\Omega$ being the boundaries of Ω (see Fig. 2.2 for a two-dimensional model example):

$$\begin{cases} \mathfrak{L}u = 0 & \text{in } \Omega \\ \gamma_D \cdot u = g & \text{on } \Gamma \\ \text{Other conditions at infinity} & \text{if } \Omega \text{ is an exterior unbounded domain} \end{cases} \tag{2.2}$$

where $\gamma_D \cdot u$ refers to the interior Dirichlet trace of u with respect to Ω and $g : \Gamma \to \mathbb{C}$ refers to a given known function. If $\Omega \subset \mathbb{R}^3$, the standard volumic approaches that rest on a volume discretization of Ω by means of a mesh, such as the Finite Element Method (FEM), can generate too large meshes for practical applications. In addition, in the case of exterior unbounded Ω, these methods are not sufficient and have to be further adapted. However, there exists a family of methods able to deal with this kind of limitations: the BEMs.

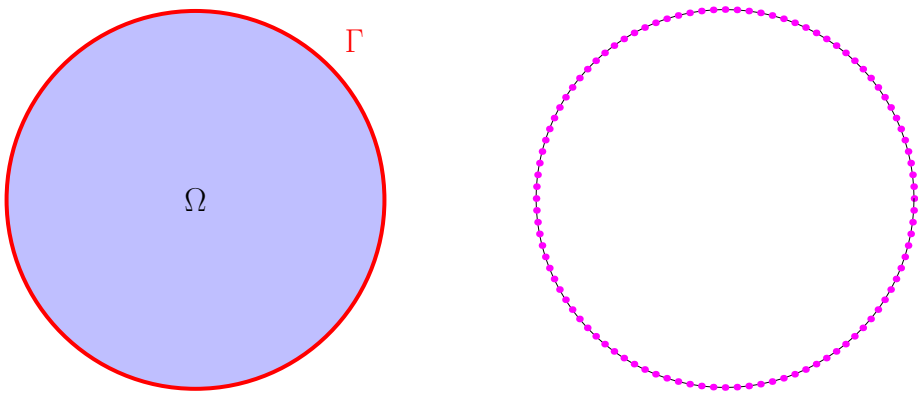


Figure 2.2: Model example of domain Ω (left) and an example of discretization of the boundaries Γ (right).

2.1.2.1 Boundary integral equations

In this paragraph G is assumed to be a Green kernel. An excellent introduction on how N-body problems arise in the numerical solution of BIE is proposed in [162]. First, we have to introduce *layer potentials* and associated *Sobolev spaces* (see [187]). We do not give much details on the potential theory since we only want to show in this paragraph how N-body problems stem from BIEs.

Definition 2.1.2. *Let Ω be a domain in $\mathbb{R}^d$ and $\Gamma := \partial\Omega$ be the boundaries of Ω. The Sobolev trace space $H^{1/2}(\Gamma)$ is defined by*

$$H^{1/2}(\Gamma) := \{v \quad : \quad \Gamma \to \mathbb{C} \quad | \quad ||v||^2_{H^{1/2}(\Gamma)} := \int_{\Gamma\times\Gamma} \frac{|v(\mathbf{x}) - v(\mathbf{y})|^2}{|\mathbf{x}-\mathbf{y}|^3} d\sigma(\mathbf{x},\mathbf{y}) < \infty\}.$$

The dual of $H^{1/2}(\Gamma)$ is the Sobolev space $H^{-1/2}(\Gamma) := H^{1/2}(\Gamma)^*$.

These Sobolev spaces are used to define the *layer potentials*.

Definition 2.1.3. *The Single Layer Potential (SLP) denoted by* **SL** *is defined for any $\mathbf{x} \in \mathbb{R}^d \backslash \partial\Omega$ and any $\phi \in H^{-1/2}(\Gamma)$ by the formula*

$$\mathbf{SL}[\phi](\mathbf{x}) := \int_\Gamma G(\mathbf{x},\mathbf{y})\phi(\mathbf{y}) d\sigma(\mathbf{y})$$

for any bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$.

Definition 2.1.4. *The Double Layer Potential (DLP) denoted by* **DL** *is defined for any $\mathbf{x} \in \mathbb{R}^d \backslash \partial\Omega$ and any $\phi \in H^{1/2}(\Gamma)$ by the formula*

$$\mathbf{DL}[\phi](\mathbf{x}) := \int_\Gamma n(\mathbf{y}) \cdot \nabla_{\mathbf{y}} G(\mathbf{x},\mathbf{y})\phi(\mathbf{y}) d\sigma(\mathbf{y})$$

for any bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$, where $n(\mathbf{y})$ denotes the outward normal vector with regard to Γ at point $\mathbf{y}$.

Then, the Green's representation formula is given by the following theorem.

Theorem 2.1.1. *(Green's Representation Formula) Suppose that ϕ is a solution of Pb. 2.2 defined in Ω. If Γ is smooth, then:*

$$\mathbf{SL}[\gamma_N \cdot \phi](\mathbf{x}) - \mathbf{DL}[\gamma_D \cdot \phi](\mathbf{x}) = \phi(\mathbf{x})\mathbf{1}_\Omega(\mathbf{x})$$

for any $\mathbf{x} \in \mathbb{R}^d$*, where* $\gamma_N \cdot \phi$ *and* $\gamma_D \cdot \phi$ *denote the interior Neumann and Dirichlet traces of* ϕ *respectively and with*

$$\mathbf{1}_\Omega(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega \\ 0 & \text{otherwise} \end{cases}.$$

When considering the interior Dirichlet trace of the representation formula in Thm. 2.1.1, one obtains

$$\gamma_D \cdot \mathbf{SL}[\gamma_N \cdot \phi] - \gamma_D \cdot \mathbf{DL}[\gamma_D \cdot \phi] = \gamma_D \cdot \phi. \tag{2.3}$$

Now, let K be the operator defined by

$$-\gamma_D \cdot \mathbf{DL}[v] = (Id/2 - K)\, v.$$

Proposition 2.1.1. *Let* $v \in H^{1/2}(\Gamma)$*, we have*

$$K[v](\mathbf{x}) = \lim_{\delta \to 0} \int_{\Gamma \setminus B(\mathbf{x},\delta)} n(\mathbf{y}) \cdot \nabla_{\mathbf{y}} G(\mathbf{x},\mathbf{y}) v(\mathbf{y}) d\sigma(\mathbf{y})$$

where $B(\mathbf{x},\delta)$ *is a ball centered in* $\mathbf{x}$ *with radius* δ*, for any* $\mathbf{x} \in \Gamma$*.*

Let V be defined by

$$V := \gamma_D \cdot \mathbf{SL}.$$

Since $\gamma_D \cdot \phi = g$ thanks to Pb. 2.2 and using Eq. 2.3, we obtain the following BIE:

$$\begin{cases} \text{Find } \gamma_N \cdot \phi \in H^{-1/2}(\Gamma) \text{ such that} \\ V(\gamma_N \cdot \phi) = (Id/2 - K)\,(g) \qquad \text{on } \Gamma \end{cases}. \tag{2.4}$$

If the solution of Pb. 2.4 is known, one can construct the solution of Pb. 2.2 by means of the representation formula in Thm. 2.1.1. Since g is known (see Pb. 2.2), the right-hand side in the equlity in Pb. 2.4 is known. Hence, to solve Pb. 2.4, one has to invert V.

2.1.2.2 Discretization and solvers for integral equations

We are concerned by the Boundary Element Method (BEM) [187,195] for the discretization of BIEs. The idea of these method consists in approximating the surface Γ with a mesh $\{\tau_j \mid j \in [\![1,n]\!]\}$, $\cup_{j=1}^n \tau_j \approx \Gamma$, and then to approximate the solution of the equation on each τ_j. We focus on the *Galerkin BEM*, that is on a variational approach to solve Pb. 2.4. Such method leads to a discretization of Pb. 2.4 as a linear system of the form $\mathbf{Av} = \mathbf{f}$ with unknown $\mathbf{v}$ and a dense matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ such that (using piecewise constant approximations)

$$\mathbf{A}_{k,l} := \int_{\tau_k} \phi_k(\mathbf{x}) \int_{\tau_l} G(\mathbf{x},\mathbf{y}) \psi_l(\mathbf{y}) d\sigma(\mathbf{y}) d\sigma(\mathbf{x})$$

for any $k,l = 1,...,n$, where ϕ_k and ψ_l vanish outside τ_k and τ_l respectively and are constant in τ_k and τ_l respectively. Denoting $\mathbf{v} = (v_1,...,v_n)$, we thus have

$$\begin{aligned} (\mathbf{Av})_k &= \sum_{l=1}^n \left(\int_{\tau_k} \phi_k(\mathbf{x}) \int_{\tau_l} G(\mathbf{x},\mathbf{y}) \psi_l(\mathbf{y}) d\sigma(\mathbf{y}) d\sigma(\mathbf{x}) \right) v_l \\ &= \int_{\tau_k} \phi_k(\mathbf{x}) \sum_{l=1}^n \int_{\tau_l} G(\mathbf{x},\mathbf{y}) \psi_l(\mathbf{y}) v_l d\sigma(\mathbf{y}) d\sigma(\mathbf{x}). \end{aligned} \tag{2.5}$$

To evaluate the integrals in Eq. 2.5 in practice, one has to apply *quadrature rules*, i.e. to approximate the intagrals by a weighted sum of nodal evaluations of the integrands. Suppose that each quadrature uses S points. The quadrature of $(\mathbf{Av})_k$ thus writes

$$\sum_{p=1}^S \omega_p(\tau_k) \phi_k(\mathbf{x}_p) \sum_{l=1}^n \sum_{q=1}^S G(\mathbf{x}_p,\mathbf{y}_q) \left(\omega_q(\tau_l) \psi_l(\mathbf{y}_q) v_l\right) \tag{2.6}$$

where $\omega_p(\tau_k)$ (resp. $\omega_q(\tau_l)$) denotes the weight of the p^{th} (resp. q^{th}) quadrature node $\mathbf{x}_p$ (resp. $\mathbf{y}_q$) in τ_k (resp. τ_l). The computation of the red part of Eq. 2.6 corresponds to a N-body problem.

There exist two main families of methods to solve a linear system:

- **Direct methods**, inverting $\mathbf{A}$ explicitly or factorizing $\mathbf{A}$ in such a way that the product by $\mathbf{A}^{-1}$ can be easily evaluated (for instance by means of LU decomposition [81, 113, 120]);
- **Iterative methods**, generating a sequence of approximations of the solution of the system until a *certain convergence* is reached (see for instance [121]).

In the general case, direct methods lead to cubic costs in the dimension of the system that become too expensive for large systems. Modern techniques can provide approximated direct solvers based on hierarchical procedures (for instance in [18–20, 33, 149, 198]) for particular structured matrices. In this thesis, we are concerned by iterative solvers based on Krylov subspaces [121]. Krylov methods construct a sequence $\mathbf{v}_j$ converging to the solution $\mathbf{v}$ of the system. Among the large family of Krylov methods, we are interested in the Generalized Minimum Residual (GMRes) method [185] that is able to deal with the non-symmetric matrices that arise in the BEM context. GMRes is an iterative algorithm constructing such a sequence of $\mathbf{v}_j$ and stopping when a certain convergence criterion is verified. In practice, few iterations may be sufficient to recover an approximated solution with a small error. The cost of each of these iterations is dominated by a product by $\mathbf{A}$. According to Eq. 2.6, a product by such a matrix $\mathbf{A}$ can be somehow interpreted as a N-body problem. Hence, the cost of GMRes applied to a BEM problem is dominated by evaluations of N-body problems. In other words, one can accelerate the resolution using a fast method for N-body problems.

2.1.3 Non-hierarchical methods

We present here a set of examples of methods that do not require a hierarchical space decomposition to solve N-body problems. The general overview given by the following list of non-hierarchical methods for N-body problems is by no means exhaustive.

2.1.3.1 Direct computation

The most naive way for solving a N-body problem consists in computing all the pairwise interactions between particles with regard to the kernel G. This results in a quadratic complexity which quickly becomes prohibitive when trying to deal with large point clouds. When no other method is available, the direct computation still remains an option for numerically solving a given N-body problem. A hardware implementation of this computation has been done for example in [145] in the particular context of gravitational forces.

2.1.3.2 The cut-off method

One of the simplest methods for the approximation of the results of N-body problems involving asymptotically smooth kernels (such as $1/r$) is the *cut-off method* [14]. The idea is simply to neglect the influence of the particles lying outside a ball of fixed radius around each target particle. In matrix terms, this consists in sparsifying a dense matrix by setting an important amount of entries to zero. This can also be seen as a direct method where a large part of the computation is omitted. This can however induce an important error (see [92, 225]).

2.1.3.3 Particle-Mesh methods

For the Poisson's equation, the *Particle-Mesh* (PM) method (and its extensions to non-uniform particle distributions) uses Fast Fourier Transforms (FFTs) combined with interpolation on regular grids in order to solve the related N-body problems [137]. Thanks to the complexity of the FFTs, this class of methods reduces the quadratic complexity of the treated N-body problems to a

linearithmic one. The main issue of this method is its efficiency that deteriorates on non-uniform distributions [218]. There exists an extension to the PM method designed to tackle this issue, namely the *Particle-Particle/Particle-Mesh* (P3M) method [137]. Basically, this consists in combining a direct method for the near interactions and the PM method for the far interactions. This enhancement also increases the precision on high particle density areas.

2.1.3.4 Ewald's summation

Periodic systems can benefit from another method: the Ewald's summation [101,106]. This method also uses a different treatment of the near and far interactions. The computations are performed using Fourier techniques, assuming that the system is infinitely periodic.

2.1.3.5 Precomputed FFT

A kernel independent non-hierarchical method based on interpolation on grids composed of equispaced points and FFT, named precomputed FFT, exploits the Toeplitz structures coming from the discretization of radial kernels on such grids [177,215,216]. This leads to an overall complexity between $\mathcal{O}(N\,logN)$ and $\mathcal{O}(N^{1.5}\,logN)$, depending on the particle distribution (see [215]).

2.1.3.6 Sparse Cardinal Sine Decomposition

The Sparse Cardinal Sine Decomposition (SCSD) [15, 16] approximates a radial Green kernel G with a sum of cardinal sines (first line of Eq. 2.7). Then, the use of the integral expression of such functions on the unit sphere (second line of Eq. 2.7) allows to obtain an approximation of this expression by means of cubatures (third line of Eq. 2.7) and, by means of Non-Uniform Fast Fourier Transform (NUFFT) [32, 90, 91, 160], a fast algorithm can be derived. This approximation is not valid in the whole computation domain, so a correction for near interactions has to be applied in order to recover the final result. The approximations read:

$$\begin{aligned} G(\mathbf{x},\mathbf{y}) &\approx \sum_p \alpha_p sinc(\eta_p|\mathbf{x}-\mathbf{y}|) \\ &= C\sum_p \alpha_p \int_{\mathbb{S}^{d-1}} e^{i\eta_p\langle \mathbf{x}-\mathbf{y},\lambda\rangle} d\lambda \\ &\approx C\sum_p \alpha_p \sum_q \omega_q e^{i\eta_p\langle \mathbf{x},\lambda_q\rangle} e^{-i\eta_p\langle \mathbf{y},\lambda_q\rangle} \end{aligned} \tag{2.7}$$

with C being a given constant, $\alpha_p \in \mathbb{C}$, $\omega_q \in \mathbb{R}$ the cubature weights and λ_q the cubature nodes. The complexity of such a method is related to the number K of needed cubature points to achieve a given accuracy. Thanks to the NUFFT, this results in a linearithmic complexity with respect to K if the correction step can be evaluated in at most the same cost. Notice that $K \neq N$ in general. A similar method is proposed in [27] for two-dimensional kernels verifying the radial property. A particular interest of this method is its ease of implementation. However, because the transformation takes into account the whole computational domain at once, this method does not adapt to the particle distribution. In consequence, the efficiency with regard to the number of particles strongly depends on their distribution and highly non-uniform distributions cannot be treated efficiently this way.

2.2 Hierarchical methods

Hierarchical methods for N-body problems refer to a family of algorithms based on hierarchical decompositions of the particle spaces, represented as trees. Originally deriving from an algorithm proposed in [23] in astrophysics, the hierarchical methods are based on two main ideas:

- The domain encompassing the particle sets can be represented by a tree whose nodes are labeled by restrictions of this domain;
- The far interactions can be approximated efficiently by easily evaluable expansions and only the near interactions have to be computed directly to reach a reasonable accuracy.

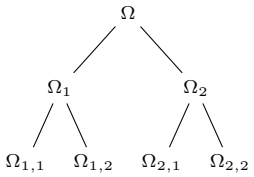
In Sect. 2.2.1 we present the tree structures used in hierarchical methods and in Sect. 2.2.2 we define the near and far fields that represent, for each particle, the amount of particle interactions that will be approximated with respect to this particle. These two ingredients are then used in Sect. 2.2.3 to present the first hierarchical method considered in this manuscript: the treecodes. In the scope of this section, the treecodes are used as a simple illustration of the concepts introduced in the paragraphs below. Their particularities will be further used in Chap. 5 where a generalized approach for hierarchical method will be presented. Then, we describe the Fast Multipole Methods (FMMs) in Sect. 2.2.4, providing explicit formulas that are used to derive fast algorithms for Laplace and Helmholtz kernels. In Sect. 2.2.5 are finally presented the algebraic counterpart of the FMM and its main generalizations.

2.2.1 Tree structures

The hierarchical aspects we mentioned in the introduction of Sect. 2.2 are related to a recursive space decomposition. Let Ω be a domain in $\mathbb{R}^d$ and $\Omega_1, \Omega_2 \subseteq \Omega$ be such that $\{\Omega_1, \Omega_2\}$ forms a partition of Ω. Each Ω_i, $i = 1, 2$ can be further partitioned, for instance in two subdomains we denote by $\Omega_{i,1}$ and $\Omega_{i,2}$. Hence we obtain $\underset{i,j=1,2}{\cup} \Omega_{i,j} = \Omega_1 \cup \left(\underset{j=1,2}{\cup} \Omega_{2,j} \right) = \left(\underset{j=1,2}{\cup} \Omega_{1,j} \right) \cup \Omega_2 = \Omega$ which provide other partitions of Ω. There exists a set of links between these partitions: starting from any partition of Ω, another partition is obtained by replacing any of its members by a partition of it. This leads to a structure that can be easily represented.

2.2.1.1 Tree representation of space

The toy example we described is very simple to represent. Suppose that an element of a partition is "connected" to the domain (or subdomain) being partitioned. This leads to the following representation



which corresponds to a tree whose nodes are labeled by a domain or a subdomain. This graph structure is what we call a tree representation of Ω. Hence, the *root* of this tree is the entire original domain and any subdomain that is not further partitioned is named a *leaf*. Each node of such a tree is named a *cell* and each cell c' that is directly connected (through only one single edge) to another one c that is closer to the root is said to be a *son of* c, c being the father of c'. We denote by $Sons(c)$ the set of sons of c and by $Father(c')$ the cell c. The *ancestors* of a cell c' is the set of cells $Ancestors(c')$ closer to the root than c' but containing c'. If c is an ancestor of c' then c' is said to be a *descendant* of c. There exist different manners of deriving such a tree starting from a given domain of $\mathbb{R}^d$ and different kinds of trees, depending on the way this domain is recursively decomposed. We present in the following paragraphs the main realizations of hierarchical tree representations of a domain in hierarchical methods. For any cell c, the center of c, denoted by $ctr(c)$, refers to the center of the smallest ball encompassing c and the radius of c, $rad(c)$ refers to the radius of this ball. The root of a tree representation T of a domain refers to this domain itself

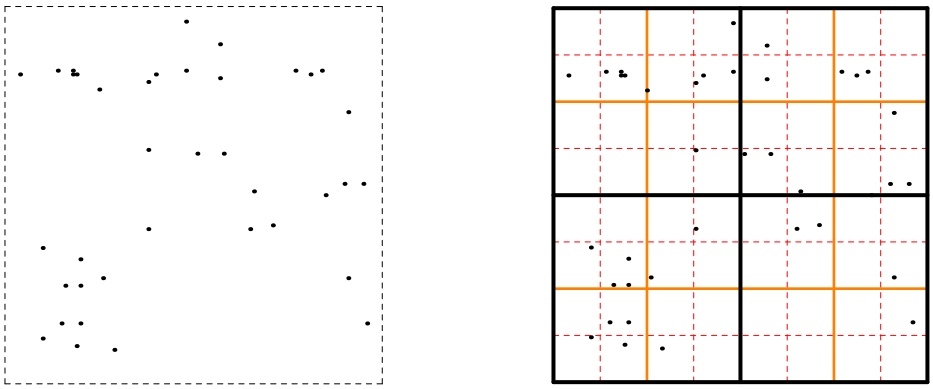


Figure 2.3: Particle distribution in 2D (left) with bounding box (dashed lines left) and regular hierarchical quadtree splitting (right) on the same point cloud. Cells of the first level (black), second level (orange) and third level (dashed+red) are illustrated.

and is denoted $Root(T)$. The levels of a tree are indexed from the root to the leaves, starting by 0. The *depth* of a tree refers to the greatest index. The total number of levels in the tree is thus equal to the depth of this tree plus one.

2.2.1.2 Splitting space

First, because we deal with point clouds (particle sets, i.e. finite sets of $\mathbb{R}^d$, $d = 1, 2, 3$), the domain that has to be considered is any closed set encompassing these point clouds. In practice, a cuboid or a d-cube[1] is usually chosen [23, 30] and referred to as a *bounding box*, as illustrated on Fig. 2.3. In the original method of Appel [23], particular binary trees (named *kd*-trees [39]) are used to represent a two-dimensional domain. This tree is constructed by recursively applying bisections according to each axis[2], dividing each time a particle set into two other ones. Another approach has been adopted for the Barnes & Hut algorithm presented in [30], where 2^d-trees space representations[3] are considered. The particles are bounded by the boundaries of a d-cube in dimension d (that is a square in 2D as in Fig. 2.3 and a cube in 3D as in Fig. 2.5). Then, this d-cube is decomposed into (up to) 2^d other equally-sized d-cubes. This process is repeated recursively until a *stopping criterion* is validated. As opposed to the *kd*-trees, the 2^d-trees operate the decomposition of a cube in an arbitrary way without considering the particle sets. In general, the stopping criterion can basically be based on

1. a maximal depth. This criterion is referred to as $MaxDepth$.

2. a maximal number of particles per leaf, that is a threshold above which the cell is divided into other ones. This criterion is referred to as $Ncrit$.

A space decomposition using a tree with the $MaxDepth$ criterion is illustrated in Fig 2.3 and the related perfect tree[4] is depicted in Fig 2.4.

On its own, the space decomposition is not sufficient to obtain a fast scheme: this still has to be combined with an approximation strategy.

[1] A d-cube is a cuboid in dimension d with equally sized sides.
[2] Separations hyperplane which is parallel to $d-1$ axes.
[3] A *quadtree* in 2D or an *octree* in 3D
[4] A 2^d-tree is said to be perfect if all its non-leaf nodes have 2^d sons and if all the leaves appear at the same tree level.

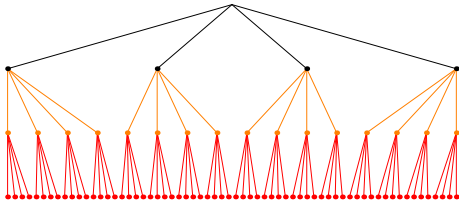


Figure 2.4: Perfect quadtree representation of the space decomposition of Fig 2.3 with a maximal depth equal to 3 and with the storage of all cells. The same colors as in Fig 2.3 are used for each level.

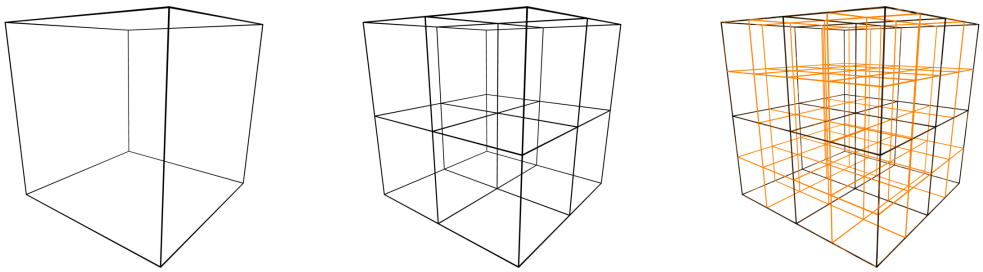


Figure 2.5: Three level octree space splitting.

2.2.2 Near and far fields

Like the Ewald summation mentioned in Section 2.1.3.4, the idea of the hierarchical methods is to divide the computation of Eq. 2.1 for each target particle $\mathbf{x}$ in two sums:

1. A *near field* part $\mathcal{N}(\mathbf{x})$, corresponding to the close particle interactions whose contributions are evaluated directly (i.e. exactly). In practice, this near field corresponds to a set of source particles that are geometrically "close" to the target particle $\mathbf{x}$.

2. A *far field* part $\mathcal{F}(\mathbf{x})$, composed of the complementary of the near field part. The contribution of the particles in the far field is approximated.

In other words, Eq. 2.1 gives:

$$\sum_{\mathbf{y}\in Y} G(\mathbf{x},\mathbf{y})q(\mathbf{y}) = \left(\sum_{\mathbf{y}\in\mathcal{N}(\mathbf{x})} G(\mathbf{x},\mathbf{y})q(\mathbf{y})\right) + \left(\sum_{\mathbf{y}\in\mathcal{F}(\mathbf{x})} G(\mathbf{x},\mathbf{y})q(\mathbf{y})\right)$$

and the second term of the right-hand side of this equation is supposed to be efficiently approximated using a fast method. This fast evaluation of the far field is one of the keys of the efficiency of hierarchical methods. The near and far fields are always defined according to a target, so there are as many near and far fields to be defined as the number of target objects (i.e. particles at this stage of the presentation). The hierarchical space decomposition comes into play in this definition: this allows to efficiently assemble $\mathcal{N}(\mathbf{x})$ and $\mathcal{F}(\mathbf{x})$ for any target $\mathbf{x}$ in terms of unions of cells.

2.2.3 Treecodes

Originally, the idea of the treecodes can be found in the method of Appel [23] and one of the most famous algorithms is presented in [30]. These methods are based on a tree decomposition of the target point cloud and these trees are generally constructed using the $Ncrit$ criterion. Basically, three tools have to be defined to present a treecode method for a given kernel:

1. An acceptance criterion that defines the near and far fields;

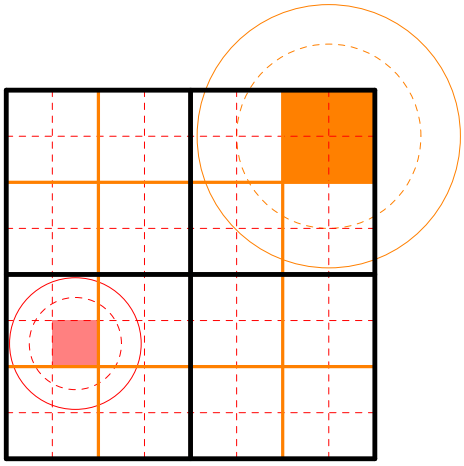


Figure 2.6: Exclusion areas of the red and the orange cells for $\theta = 0.7$ (continuous lines) and $\theta = 1$ (dashed lines). We assumed here that the center of mass of the particles in these cells coincide with the center of these cells.

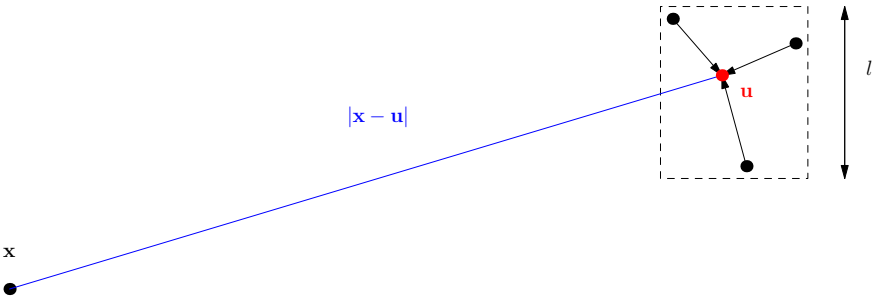


Figure 2.7: Schematic representation of the Barnes & Hut acceptance criterion

2. A way of approximating the kernel on the far field;

3. An appropriate tree traversal allowing for each particle to efficiently determine the near and far field parts.

2.2.3.1 Acceptance Criterion

In the Appel's algorithm, an explicit approximation of the kernel is used for a target particle $\mathbf{x}$ and a cell c, with an error controlled by the condition

$$\frac{rad(c)}{|\mathbf{x} - ctr(c)|} \leq \theta \tag{2.8}$$

for a constant $\theta \in \mathbb{R}^+$ (in practice $\theta \in [1/2, 1]$). Such a c is in the far field of $\mathbf{x}$. For any target particle $\mathbf{x}$, the leaf cells that do not respect the ratio in Eq. 2.8 are in the near field of $\mathbf{x}$. The smaller θ, the larger the near field. The influence of θ on the near field is depicted on Fig. 2.6. In the Barnes & Hut algorithm [30], the radius of the cell is replaced by the length l of the cell (see Fig. 2.7). The center of the cells is chosen to be the center of mass of the particles in this cell.

These two criteria are referred to as **acceptance criteria** (AC) and are defined using purely geometric elements. In other words, the AC does not consider the particles in a cell but the cell itself. When the AC is verified for a given particle and a given cell, this *particle-cell* pair can be approximated in the treecode algorithm. Such a pair $(\mathbf{x}, c)$ is said to be **well-separated** and c lies in the far field of $\mathbf{x}$ if none of the ancestors of c is well-separated from $\mathbf{x}$. In practical implementations, an AC is a boolean function that returns *true* if and only if the tested *particle-cell* pair is well-separated [211].

2.2.3.2 Algorithms

The next step in order to derive a fast method consists in *aggregating* the contributions of the particles lying in the same cell. In the Barnes & Hut algorithm, the charges of the particles of a cell are simply added. This procedure is denoted **Aggregate**($\mathbf{c}$) for any cell c. The result of such a procedure operating on a non-leaf cell c can be based on the results of $Aggregate(c')$ for each $c' \in Sons(c)$, which reduces the aggregation cost. An algorithm for this aggregation process is denoted by *Upward Pass* and a formulation is given in Alg. 1.

The computation of a *particle-cell* interaction on a well-separated *particle-cell* pair $(\mathbf{x}, c)$ is realized by a procedure denoted **Approx_Interact**($\mathbf{x}, \mathbf{c}$). The direct computation of the contribution to the N-body problem of two particles $\mathbf{x}$ and $\mathbf{y}$ is computed using the procedure **Direct_Interact**($\mathbf{x}, \mathbf{y}$). Hence, in Alg. 2 is described the tree traversal allowing for any particle to evaluate the near and far field contributions. Finally, a "naive" global treecode algorithm is summarized in Alg. 3. We recall that the target point cloud is denoted X and the source one Y.

Algorithm 1 UpwardPass

```
// Input: c ∈ 𝒮 with 𝒮 a tree representation of Y
// Output: ∅
procedure UpawardPass(c)
    if c is not a leaf then
        for c' ∈ Sons(c) do
            UpwardPass(c')
        end for
    end if
    Aggregate(c)
end procedure
```

Algorithm 2 STT (Single Tree Traversal)

```
// Input: x ∈ X, c ∈ 𝒮 with 𝒮 a tree representation of Y
// Output: ∅
procedure STT(x, c)
    if AC(x, c) then
        // The approximation can be used
        Approx_Interact(x, c)
    else
        if c is a leaf then
            // Direct computation.
            // The intersection c ∩ Y refers to the particles of Y included in c
            for y ∈ (c ∩ Y) do
                Direct_Interact(x, y)
            end for
        else
            // Explore the deeper tree levels
            for c' ∈ Sons(c) do
                STT(x, c')
            end for
        end if
    end if
end procedure
```

Algorithm 3 Treecode

```
// Input: X, 𝒮 a tree representation of Y
// Output: ∅
procedure TREECODE(X, 𝒮)
    // Aggregate approximations
    UPWARDPASS(Root(𝒮))
    // Traverse source tree for each target particle
    for x ∈ X do
        STT(x, Root(𝒮))
    end for
end procedure
```

Remark 2.2.1. *The method given in Alg. 3 does not minimize the number of tree traversals. Instead of this algorithm considering individually each target particle, groups of clustered target particles can be considered [31, 88]. In this case the distance involved in the AC (that is the denominator in Eq. 2.8) becomes the distance between the boundaries of the smallest cuboid encompassing this group of particles and the center of mass of the source cell.*

2.2.3.3 Complexity

The popularity of treecodes stems from their ease of implementation and their complexity. We consider in this paragraph the complexity of Algs. 1, 2 and 3. Let N be the size in terms of particles of the two point clouds X and Y. The complexity of Barnes & Hut algorithm is of order $\mathcal{O}(NlogN)$ (i.e. *linearithmic* complexity). The intuitive idea for obtaining this complexity is based on the fact that the number of levels in the tree representation of the source point cloud that need to be traversed in the STT (Alg. 2) is equal (in mean) to $\mathcal{O}(logN)$. Hence, the average cost of a STT is $\mathcal{O}(logN)$. Because there is a number of STT in Alg. 3 equal to the number of target particles, that is N, we obtain the mentioned $\mathcal{O}(NlogN)$. The complexity of the Upward Pass is also $\mathcal{O}(NlogN)$ because the same argument in the number of levels can be applied: there are $\mathcal{O}(logN)$ levels so the number of cells is $\mathcal{O}(NlogN)$ and the Upward Pass (which is assimilated to a depth-first search algorithm) traverses each cell once, leading to a $\mathcal{O}(NlogN)$ complexity. Hence, the overall complexity is $\mathcal{O}(NlogN)$. See also [100] for the complexity analysis of the Appel's algorithm.

2.2.3.4 Extensions

The Appel and Barnes & Hut algorithms were designed for astrophysical applications, but there exist many other kinds of treecodes, designed for other kernels (see for instance [24, 102]). Quite recently, kernel-independent treecodes [146, 168, 205, 209] were introduced and allow to deal with a single code with arbitrary (non-oscillatory) kernels.

Remark 2.2.2. *To our knowledge, there is no treecode with $\mathcal{O}(Nlog(N))$ complexity for the Helmholtz kernel in the high-frequency regime.*

2.2.4 Fast Multipole Methods

In [115], Greengard & Rokhlin describe an algorithm named the Fast Multipole Method (FMM) that paves the way for a new kind of hierarchical methods, able to deal with a N-body problem in a linear time. The FMM is one of the ten algorithms cited in [85]. The algorithm in [115] treats two-dimensional uniform distributions (uniform FMM). In [55] is proposed a version of the FMM adapted to non-uniform distributions (adaptive FMM). Three-dimensional extensions of these methods are introduced in [118]. The adaptive three-dimensional case is considered in [65].

Since its introduction for the Poisson equation, the FMM has been extended to many other kernels such as the Lennard-Jones potential [82, 92], the Helmholtz kernel [60, 70, 97, 114] and Maxwell equations [75], the Yukawa potential [117, 140, 223], the Stokes equations [201, 207], the elastodynamics equations [58, 59], the Gaussian kernel [119, 194, 208], *etc...* Because of its efficiency and its ability to cover a large spectrum of applications, the FMM has become a very popular algorithm, declined in many ways. We restrict our presentation in the following paragraphs to the algorithm itself, putting aside its inclusion in other methods.

2.2.4.1 Uniform algorithm

At the most abstract level, the idea of the FMM is to introduce a second type of approximation in addition to the far field approximation of the treecodes, combined with a tree representation of the target point cloud in addition to the source tree representation already used in treecodes. The first of these elements takes the form of *local approximations* that are only valid (up to a certain error) in the cells in which they are defined. Hence, the *particle-cell* interactions of the treecodes are replaced by *cell-cell* interactions that aim at transforming the far field approximations into local ones. Because the particles are not anymore directly involved in this process between approximations, an additional computational gain is obtained. Finally, the FMM algorithm operates a last tree traversal on the target tree to transform the local expansions into contributions to the final sum. In other words, the results of the *cell-cell* interactions are aggregated into local approximations that are transformed into contributions at the particle level only at the end of the whole process. We describe here the uniform algorithm first presented in [115].

On the use of two trees. Because the FMM considers a tree representation of both the target and source point clouds, it is convenient to adopt the following strategy: defining a cubic box B encompassing the two point clouds, two 2^d-trees, $\mathscr{S}$ and $\mathscr{T}$, with $Root(\mathscr{S}) = Root(\mathscr{T})$, can be constructed, $\mathscr{S}$ being a tree representation of the source point cloud Y and $\mathscr{T}$ one of the target point cloud X. Hence, the positions of the cells of the two trees coincide if they are non-empty, but they play different roles. A consequence is that they can be both represented with the same tree. For instance, if we choose $X = Y$, the cells represented on Fig. 2.3 correspond to target and source cells, but one has to keep in mind that two trees exist.

Expansion types. In the FMM, as opposed to treecodes, the contribution of each *cell-cell* interaction involving the same target cell is converted into and added to an approximation valid in this cell. The FMM uses user chosen approximation orders. Hence, the far field approximations are vectors with a size dictated by this order. The term **multipole expansion** is used to refer to such a far field approximation. The behavior of the influence of the far field of a cell on it is a function that can also be approximated by an appropriate expansion named **local expansion**. The *cell-cell* interactions consist in transforming the multipole expansions into local ones. The amount of work needed to compute the contribution of the far field of a target cell is no more related to the number of particles that are included in it but only on its geometric characteristics and on the approximation method. These transformation processes between expansions are depicted on Fig. 2.8. To each source (resp. target) cell is attached a unique multipole (resp. local) expansion, so that we can speak about *the* multipole (resp. local) expansion of a cell.

Once all the necessary *cell-cell* interactions are computed, the local expansions in each target cell contains the information about the far field of this cell. In the FMM algorithm, an additional step is needed to recover the contribution to the final sum. This is done by adding the information contained in the local expansions of the ancestors of a cell to its own local expansion and, if the target cell is a leaf, by transforming this information into an effective contribution of the far field of each particle in this cell. If the cost of a *cell-cell* interaction is approximately constant at each tree level, it is straightforward that the lower the 2^d-tree level of an interacting *cell-cell* pair, the better the computational efficiency for the indirectly involved target particles.

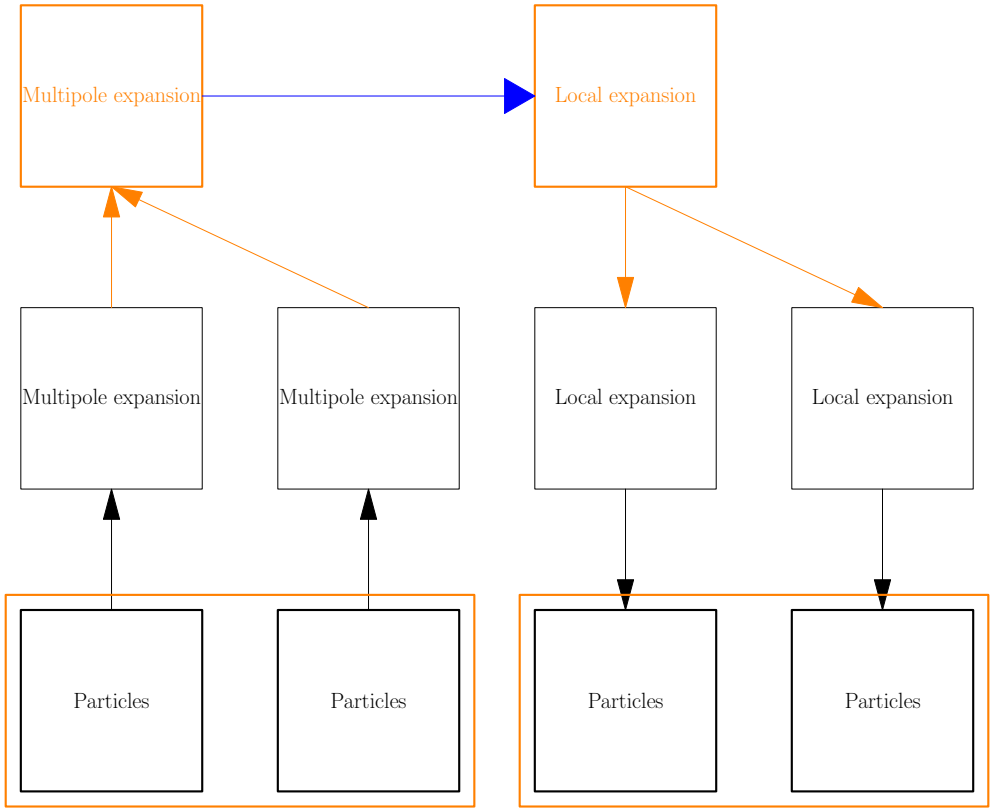


Figure 2.8: Links between expansions. Multipole expansions are assembled at the leaf level (black arrows from particles to multipole expansions) and aggregated to multipole expansions in the upper levels (orange arrows from multipole expansions to multipole expansions). Multipole expansions are then transformed into local ones (blue arrow). Local expansions are then transformed into another local expansions in lower cells (top-down orange arrows) and finally converted into effective contributions at the particle level (top-down black arrows).

However, each of these approximations has a restricted convergence domain. Their manipulation has to be done according to a criterion preserving a global controlled error. The definition of this criterion is quite different than the AC of treecodes since the two arguments of it are both cells.

Multipole Acceptance Criterion. Instead of Acceptance Criterion, the term **Multipole Acceptance Criterion** (MAC) is used in the context of the FMM. The MAC can be seen as a boolean function testing two cells. If a pair of target and source cells is **well-separated**, in the sense that the conversion of a multipole expansion associated to the source cell into a local one associated to the target cell can be safely computed, the MAC returns *true*. If this is not the case, the MAC returns *false*. In the original FMM algorithm (uniform case), this MAC is implemented as a simple criterion: two cells are supposed to be well-separated if and only if their distance, defined as the minimal distance between a point of the first cell and a point of the second one, is strictly positive[5]. This means that two cells at the same level are well-separated if and only if there exists a cell between them. A two-dimensional example is given in Fig. 2.9. This criterion is refered to as the *strict MAC*.

This MAC is used under the assumption that the *cell-cell* interactions only involve pairs of cells belonging to the same level of the 2^d-trees. Two well-separated cells do not necessarily lead to a

[5]Still considering that 2^d-tree representations sharing the same root are used for both X and Y

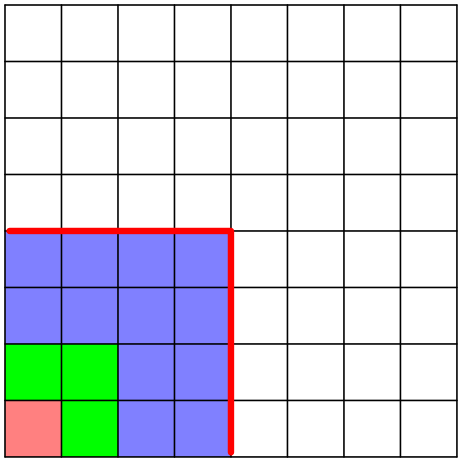


Figure 2.9: Target cell t (red) and its well-separated source cells (blue and white). Near field of t (green). t only interacts at this level with blue cells. Beyond the red separation line, *cell-cell* interactions involving the space part containing t are realized at upper levels of the trees. This situation corresponds to the strict MAC used in the uniform algorithm of the FMM.

cell-cell interaction since a pair of their ancestors may already be well-separated.

Interaction lists. The set of source cells interacting with a target one is a crucial concept in the FMM that needs a proper definition.

Definition 2.2.1. *Let t be a target cell. The set of source cells interacting with t is composed of the source cells s that are well-separated from t and such that there is no pair $(a, b) \in Ancestors(t) \times Ancestors(s)$ such that a and b are well separated. This set is named the* ***interaction list*** *of t and is denoted by $\Lambda(t)$. In a mathematical form, this reads:*

$$\Lambda(t) := \{s \in \mathscr{S} \mid MAC(s,t) = true \text{ and } \forall a \in Ancestors(t),\ \forall b \in Ancestors(s),\ MAC(a,b) = false\}.$$

Remark 2.2.3. *An analogous definition when using the strict MAC (that can be found in [115]) consists in starting from the definition of a* ***neighbor*** *of a cell. Let c_1 be a cell. A cell c_2 is in the* ***neighborhood*** *of c_1, denoted $Neigh(c_1)$, if and only if $MAC(c_1, c_2) = false$. Hence we have*

$$\Lambda(t) = \{s \in \mathscr{S} \mid MAC(s,t) = true \text{ and } Father(s) \in Neigh(Father(t))\}.$$

We easily see (referring to Fig. 2.9) that for any target cell t, the strict MAC of the uniform algorithm bounds the cardinal of $\Lambda(t)$, $\#\Lambda(t)$, with regard to the dimension d in the following way:

$$\#\Lambda(t) \leq 6^d - 3^d.$$

The way the interaction lists are determined is one of the critical steps of the FMM. In the method of Greengard & Rokhlin [115], these interaction lists are implicitly known because the 2^d-trees are supposed to be perfect, considering the relative positions between the cells at a same given level. No additional method is needed to recover these interaction lists in the uniform algorithm. Notice that in this original method, the $MaxDepth$ criterion is used to construct the 2^d-trees.

Operators. Before describing the algorithms of the uniform FMM, we quickly provide the general terminology for the FMM expansion transformations. This is summarized in a set of six (linear) operators that are listed below. The expressions of these operators are specific to each kernel. Explicit formulas for the Laplace and Helmholtz kernels are given in 2.2.4.3.

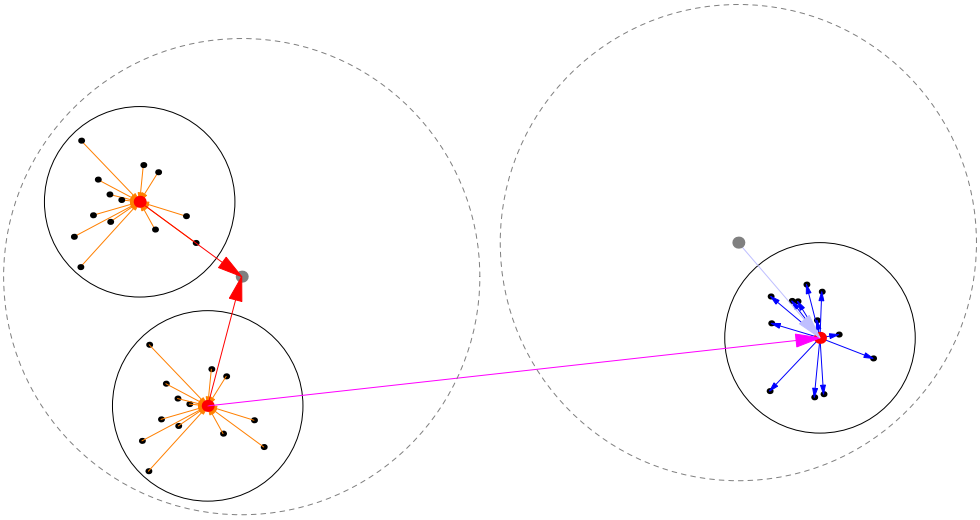


Figure 2.10: Schematic representation of FMM operators: **P2M** (orange), **M2M** (red), **M2L** (magenta), **L2L** (pale blue), **L2P** (blue). Circles represent cells and dashed circles represent the fathers of the cells they encompass. There are two kinds of incoming contributions to each target cell t: the **L2L** operator one (from its father) and the **M2L** operator ones (from well-separated cells in the interaction list of t).

1. **P2M** (**P**articles-to-**M**ultipole): takes a source cell s in argument and assemble the mutlipole expansion in s from the charges of the particles in s.

2. **M2M** (**M**ultipole-to-**M**ultipole): takes a source cell s and $s' \in Sons(s)$ in argument and adds to the multipole expansion in s the contribution of the multipole expansions in s'. This can be understood in the same way as the aggregation routine (see Section 2.2.3.2) of the treecodes.

3. **M2L** (**M**ultipole-to-**L**ocal): takes a source cell s and a target cell t in argument and transforms the multipole expansion in s into a local expansion in t, then adds the results to the local expansion of t. The application of a **M2L** operator corresponds to a *cell-cell* interaction.

4. **L2L** (**L**ocal-to-**L**ocal): takes a target cell t' and $t := Father(t')$ in argument and converts the local expansion in t into a local expansions in t', then adds the result to the local expansion in t'.

5. **L2P** (**L**ocal-to-**P**articles): takes a target cell t in argument and transforms the local expansion in t into an effective contribution of the far field of t to each particles in t.

6. **P2P** (**P**articles-to-**P**articles): takes a target cell t and a source cell s in argument and computes all the *particle-particle* interactions between the particles of t and those of s. This computation is done directly (i.e. using direct kernel evaluations).

A schematic representation of the action of these operators is provided in Fig. 2.10. In the uniform algorithm, the operators **P2M**, **L2P** and **P2P** are only applied at the deepest level. One may notice that there is no *cell-cell* interaction at the two highest levels (the root and its sons) since no pair of cells at these levels are well-separated. Using all these definitions, we are in position to provide the algorithms corresponding to the FMM in its uniform version.

Algorithms. The FMM is usually presented through two algorithms: an upward pass during which the multipole expansions are computed (i.e. the **P2M**s and **M2M**s operators are applied)

and a downward pass where the *cell-cell* interactions are performed (i.e. the **M2L**s are applied) and the local expansions transformed into local ones in lower cells or into effective contributions at the particle level (i.e. the **L2L**s and **L2P**s are applied). The upward pass is described in Alg. 4 and the traditional downward pass in Alg. 5, where we denote by lvl_{max} the user chosen depth of the 2^d-trees, $T_{|l}$ the restriction to the cells at level l of any tree T and $Leaves(T)$ the set of leaves of T.

Algorithm 4 Upward Pass

```
1: // Input: 𝒮 the 2^d-tree representation of Y
2: // Output: ∅
3: procedure UpwardPass(𝒮)
4:     Assemble the multipole expansions at the leaf level
5:     for s ∈ Leaves(𝒮) do
6:         P2M(s)
7:     end for
8:     // From the leaf level to the root...
9:     for l = lvl_max − 1, ..., 0 do
10:        // ...assemble the multipole expansions
11:        for s ∈ 𝒮_|l do
12:            for s' ∈ Sons(s) do
13:                M2M(s, s')
14:            end for
15:        end for
16:    end for
17: end procedure
```

Algorithm 5 Downward Pass

```
// Input: 𝒯 the 2^d-tree representation of X, 𝒮 the one of Y
// Output: ∅
procedure DownwardPass(𝒯, 𝒮)
    // At each level, from the root to the leaves...
    for l = 2, ..., lvl_max − 1 do
        // for each cell on this level...
        for t ∈ 𝒯_|l do
            // ...evaluate the interaction list...
            for s ∈ Λ(t) do
                M2L(t, s)
            end for
            // ...and transmit the information to the sons.
            for t' ∈ Sons(t) do
                L2L(t', t)
            end for
        end for
    end for
    // For each leaf cell...
    for t ∈ 𝒯_|lvl_max do
        // ...compute far field using the local expansion...
        L2P(t)
        // ...and perform the direct computation for the near field.
        for s ∈ 𝒩(t) do
            P2P(t, s)
        end for
    end for
end procedure
```

Complexity. The complexities of Algs. 4 and 5 depend on the costs of the **P2M**, **M2M**, **M2L**, **L2L** and **L2P** operators (the cost of the **P2P** operator being only dictated here by the particle distribution). In the original FMM for the Poisson's equation, the complexity of each of these operators is only linked to the approximation order which is considered to be $\mathcal{O}(1)$ (the size of a multipole/local expansion is supposed to be a constant). In addition, in this method, the cost of an operator is the same for each level. To achieve the linear complexity, one has to assume that the number of particles N divided by the number $C(N)$ of cells for the deepest level is such that

$$\exists a \leq b \in \mathbb{R}^{+*}, \quad \frac{N}{C(N)} \in [a, b] \tag{2.9}$$

where a and b are constants independent of N. Because the size of the interaction list is bounded (by a constant), the application of the **M2L** operators involving the same target cell costs $\mathcal{O}(1)$. The assumption 2.9 ensures that the application of all the **P2P** operators is $\mathcal{O}(N)$. The complexity of each **P2M** and **L2P** operator is $\mathcal{O}(1)$ using the same assumption. Finally, all other operators are applied once per non-leaf cell. The total number of cells is also $\mathcal{O}(N)$ thanks to the assumption 2.9 which allows to conclude that the overall FMM costs $\mathcal{O}(N)$.

An important remark concerns the choice of the method. The complexities we gave are valid only if the operators have a cost $\mathcal{O}(1)$ at each level. This is not the case in every FMM formulations. For instance, the FMMs designed for the Helmholtz kernel in the high-frequency regime involve operators whose cost depend on the level in a way that impacts the global complexity. This will be further discussed in Chap. 3.

The assumption 2.9 is somehow related to a uniform particle distribution. This version is

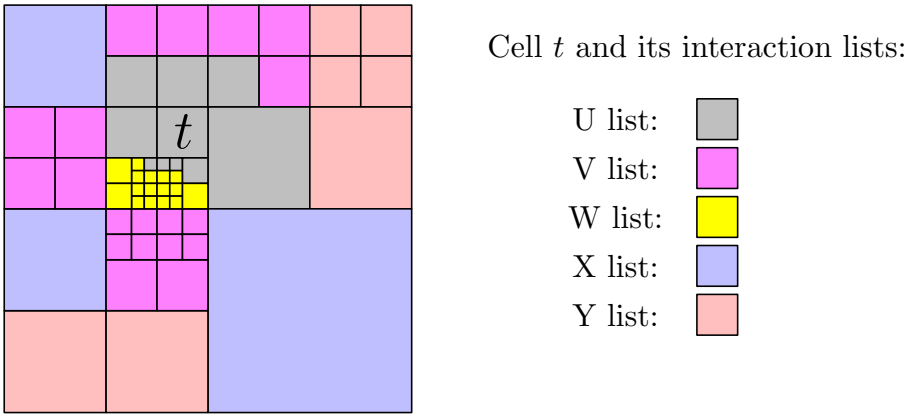


Figure 2.11: Adaptive FMM interaction lists, following [55].

obviously not well suited to deal with highly non-uniform distributions. Hence, there exists a popular variant that tackles this issue.

2.2.4.2 Adaptive algorithm

The adaptive algorithm of the FMM was originally proposed in [55] and then enhanced in [65], based on the same ideas than in the algorithm presented in [83]. In such an adaptive FMM, the 2^d-trees are constructed using the $Ncrit$ criterion (see Sect. 2.2.1.2): the leaves are no more determined by their level in the trees but by the number of particles they contain. The resulting 2^d-trees are not perfect. The idea is to match the cases in which the assumption 2.9 is not verified, which arises in many applications, such as in the BEM context. The main difference with the uniform algorithm is that the interaction lists will differ because of the non-perfectness of the trees. The adaptive FMM still reaches the $\mathcal{O}(N)$ complexity, assuming that the operators all cost $\mathcal{O}(1)$ to apply[6], when dealing with average non-uniform distributions. The global error of the FMM depends on many parameters: the distribution, the expansion types, the formulation of the operators... In practice, the error estimates are given relatively to a particular FMM formulation. However, we may mention that in [93] is given a quite general error analysis.

UVWXY-lists approach. Because the leaf cells may now appear at any level, the simple interaction list and the neighborhood of a cell (involving only cells at the same level) are not relevant anymore. In [55] are introduced five different lists that are depicted in the two-dimensional case on Fig. 2.11, allowing to distinguish all required interactions. We briefly describe these lists.

- The U-list, only defined for leaf cells, contains the set of all direct neighbors of t (see Fig. 2.11). This list corresponds to the set of source cells s such that the interaction between t and s is computed directly, using the **P2P** operator.

- The V-list, which is composed of the set of source cells s at the same level of t well-separated from t but with $MAC(Father(t), Father(s)) = false$ and such that the interaction between t and s is computed using the **M2L** operator. This corresponds to the definition we gave of $\Lambda(t)$.

- The W-list, denoted by $W(t)$ and defined for the leaf cells only, contains the set of source cells s such that the distance between s and t verifies $dist(t, s) \geq w(s)$, where $w(s)$ denotes the side length of s. This means that the multipole expansion in s can be converted into a

[6] The FMM for high-frequency Helmholtz kernel does not veriry this assumption.

contribution to the paticles in t, which is done by using an additional operator named **M2P** (**M**ultipole-to-**P**articles).

- The X-list, never explicitly built when the target and source point clouds are the same, is composed of the source cells s such that $t \in W(s)$. Hence, the contribution of the particles of s can be added to the local expansion in t, which is done, again by a new operator named **P2L** (**P**articles-to-**L**ocal).

- The Y-list, never explicitly built, contains the source cells s such that $\exists c \in Ancestors(t)$ with $MAC(c, s) = true$ and the *cell-cell* interaction between c and s has already been computed using a **M2L** operator. This means that the influence of s on the ancestors of t is contained in the local expansions of c and will be transmitted to t by applying the **L2L** operator.

Remark 2.2.4. *The application cost of a **M2P** or **P2L** is related to the FMM expansion expressions. In practice, the application of a **P2P** operator can be less costly than a **M2P** or a **P2L** if the number of involved particles is smaller than a threshold. In practical implementation, this distinction may take the form of a simple test deciding what is the best operator to apply in term of complexity. This optimization is described in [65].*

The upward and downward passes presented in the uniform case in Section 2.2.4.1 can be written under the adaptive FMM context using a recursive form that takes into account the non-perfect trees. The construction of the U, V and W lists may have a cost: depending on the distribution, some of these lists may be unbounded.

Dual Tree Traversal. Interaction lists different from the one of the uniform algorithm can be found for instance in [43, 53, 147]. In the adaptive FMM algorithm, the way the distance between well-separated cells is controlled is obtained in the same manner than for the treecodes (see Section 2.2.3.1), introducing an appropriate ratio. We here give the criterion provided in [92], for a parameter $\theta \in \mathbb{R}^{+*}$ and two cells t and s:

$$\frac{rad(t) + rad(s)}{|ctr(t) - ctr(s)|} \leq \theta. \tag{2.10}$$

This criterion is referred to as the *adaptive MAC*. If a pair (t, s) verifies 2.10, then this pair is well-separated.

The construction of the interaction list from a given MAC can be achieved in a recursive way by traversing the source and target trees. This is the purpose of the Dual Tree Traversal (DTT) algorithm [79, 212, 220]. We provide in Alg. 6 the DTT algorithm, where the routine **Explore** returns $(t, Sons(s))$ if $rad(t) > rad(s)$ and $(Sons(t), s)$ otherwise. Notice that the interaction lists are not constructed anymore using the DTT.

Algorithm 6 DTT (Dual Tree Traversal)

```
1: // Input: Target cell t ∈ 𝒯, source cell s ∈ 𝒮
2: // Output: ∅
3: procedure DTT(t, s)
4:     if MAC(t, s) then
5:         M2L(t, s)
6:     else
7:         if isLeaf(t) or isLeaf(s) then
8:             P2P(t, s)
9:         else
10:            T, S = Explore(t, s)
11:            for t' ∈ T do
12:                for s' ∈ S do
13:                    DTT(t', s')
14:                end for
15:            end for
16:        end if
17:    end if
18: end procedure
```

Remark 2.2.5. *The **M2P** and **P2L** operators can be included in the DTT with only small modifications: tests are added after the MAC application deciding if such an operator can be efficiently applied.*

The main advantages of the Dual Tree Traversal are its simplicity and its ability to deal with variable or various MACs combined with trees built using the $Ncrit$ criterion without strong algorithmic modifications.

Alternatives. When dealing with non-uniform distributions, the 2^d-trees generated using the $MaxDepth$ criterion may have cells containing no particle. The number of empty cells can be important enough in practical applications to impact the complexity if they are stored and/or treated by the algorithm. However, they should not involve computations since they have no effect on the other cells. 2^d-trees based on $MaxDepth$ avoiding the computations in empty cells are considered in [176, 183]. The notion of *adaptive cell*, as a cell that has more than a single non-empty son, is used to evaluate only the needed cells: a non-adaptive cell can be replaced by its son. A data structure well suited for both uniform and non-uniform distributions using this adaptive version has been introduced in [73], named *octree with indirections*. Another method based on a particular structure named *compressed octree* has been proposed in [17, 133, 134, 190].

One may notice that for non-uniform distributions, the approach based on the $MaxDepth$ criterion has the disadvantage of generating cells with a greatly varying number of particles, which may deteriorate the performances of hierarchical algorithms such as the Fast Multipole Method (this argument is expressed, for instance, in [179]). A comparison between different hierarchical codes (treecodes and FMMs) using $MaxDepth$ and $Ncrit$ on uniform and non-uniform distributions has been done in [104], illustrating the behavior of the different approaches and suggesting that the $Ncrit$ criterion is better suited than $MaxDepth$ for non-uniform distribution since it naturally adapts to the particle distribution.

There exists a third approach considering binary trees instead of 2^d-trees and realizing another kind of adaptivity with respect to the distribution. This family of methods is named *balanced tree FMM* [49, 93, 138] and realizes an adaptive FMM algorithm based on balanced trees: non-leaf nodes are decomposed according to the local particle distribution using well chosen median planes, keeping constant the number of tree levels.

2.2.4.3 Mathematical foundations of kernel explicit multipole and local expansions

In this section, we present two three-dimensional realizations of the Fast Multipole Method: one for the Laplace kernel and one for the Helmholtz kernel (focusing on the high-frequency regime), both based on explicit expansions of kernels using special functions in three dimensions. Originally, the FMM was designed for Poisson's equation using explicit formula only valid in this particular case (a specific kernel and a given dimension), and the early FMM versions for other kernels were also based on explicit expansions of these particular kernels. Hence, these methods are referred to as *kernel-explicit* methods. We are mainly concerned here by the exact formulas, the discretization of the Helmholtz method leading to technical difficulties, which will be discussed in Chapter 3. To be more precise, we present in the following how the continuous **P2M**, **M2M**, **M2L**, **L2L** and **L2P** operators are defined. The presentation of these two methods are motivated by the following points:

- They give concrete instances of the definitions of the FMM operators.
- They lead to the same global algorithm but using quite different tools, which shows the desire of a unified framework (see Chapter 5).
- These two formulations are discretized in different manners, which explains how the kernel-explicit method implementations are specific to a particular case.
- The Laplace case illustrates the techniques used in a large panel of FMMs for non-oscillatory kernels (including the low-frequency Helmholtz kernel) and Helmholtz one illustrates the difficulties appearing for the highly oscillatory kernels (see Chapter 3).

We may emphasize that we present the FMM operators in these two cases in a non-standard way: the theorems leading to their definitions are used in the same manner than in the literature but we define them as *mathematical* operators, acting on application sets. This formulation actually prepares to the general formulation we propose in Chapter 5.

Operators for the 3D Laplace kernel. First, following [97]

$$\frac{1}{|\mathbf{x}-\mathbf{y}|} = \sum_{n=0}^{+\infty} \frac{r_<^n}{r_>^{n+1}} P_n\left(\left\langle \frac{\mathbf{x}}{|\mathbf{x}|}, \frac{\mathbf{y}}{|\mathbf{y}|} \right\rangle\right)$$

where $\mathbf{x}$ and $\mathbf{y}$ are expressed in spherical coordinates $(r_\mathbf{x}, \theta_\mathbf{x}, \phi_\mathbf{x})$ and $(r_\mathbf{y}, \theta_\mathbf{y}, \phi_\mathbf{y})$ respectively, $r_> := max\{r_\mathbf{x}, r_\mathbf{y}\}$, $r_< := min\{r_\mathbf{x}, r_\mathbf{y}\}$ and P_n denotes the Legendre polynomial of order n (see [178] Chap. 14 Paragraph 7).

We may then mention the classical spherical addition theorem (see [118]) which expresses such evaluations of Legendre polynomials in terms of spherical harmonics (see [178] Chap. 14 Paragraph 30(i)).

Theorem 2.2.1. *(Spherical harmonics addition theorem [178] Chap. 14 Paragraph 30(iii)) Denoting by Y_l^m the spherical harmonic of order m and degree l, the following holds:*

$$P_l(\langle \nu, \xi \rangle) = \frac{4\pi}{2l+1} \sum_{m=-l}^{l} Y_l^m(\theta_\eta, \phi_\eta) Y_l^{m*}(\theta_\xi, \phi_\xi)$$

$\forall\, \eta, \xi \in \mathbb{R}^3$ *with* $|\eta| = |\xi| = 1$.

Now, we introduce the following quantities (see [97])

$$\begin{cases} O_l^m(r,\theta,\phi) & := \frac{(-1)^l i^{|m|}}{A_l^m r^{l+1}} Y_l^m(\theta,\phi) \\ \\ I_l^m(r,\theta,\phi) & := i^{-|m|} A_l^m r^l Y_l^m(\theta,\phi) \end{cases}$$

$\forall (l,m) \in \mathbb{N} \times \mathbb{Z}, 0 \leq |m| \leq l$, with $A_l^m := \frac{(-1)^l}{\sqrt{(l-m)!(l+m)!}}$.

Then, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$, the following theorem expresses the Laplace kernel as a series of separable expressions.

Theorem 2.2.2. *(Classical translation theorem [97]) If $|\mathbf{x}| > |\mathbf{y}|$, then*

$$\frac{1}{|\mathbf{x}-\mathbf{y}|} = \sum_{m=0}^{+\infty} \sum_{l=-m}^{m} (-1)^m O_m^l(\mathbf{x}) I_m^{-l}(\mathbf{y}). \tag{2.11}$$

We denote by $\mathbb{N}_Z$ the restriction of $\mathbb{N} \times \mathbb{Z}$ such that

$$\mathbb{N}_Z := \{(m,l) \in \mathbb{N} \times \mathbb{Z},\ 0 \leq |l| \leq m\}. \tag{2.12}$$

The multipole and local expansions will be *applications* from $\mathbb{N}_Z \to \mathbb{C}$.

Remark 2.2.6. *In practice, the series*

$$\sum_{m=0}^{p} \sum_{l=-m}^{m} (-1)^m O_m^l(\mathbf{x}) I_m^{-l}(\mathbf{y})$$

pointwise converges exponentially fast in p to $\frac{1}{|\mathbf{x}-\mathbf{y}|}$ so the discretization is done only by truncating Eq. 2.11. Notice that the convergence is exponential on well-separated sets (with regard to the adaptive MAC given in Eq. 2.10), implying that only a small number of terms are needed for a reasonable accuracy. Hence, one only has to replace $\mathbb{N}_Z$ by the induced restriction (i.e. $\{(m,l) \in \mathbb{N}_Z, m \leq p\}$) in the definitions of the 3D Laplace FMM operators given in the following to obtain the discrete version, used in practical algorithms.

Two other theorems allow to define the classical operators of the FMM in the 3D Laplace kernel case.

Theorem 2.2.3. *(Outer-to-outer and Outer-to-Inner Laplace translation theorem [97]). Assuming $|\mathbf{x}| > |\mathbf{y}|$, the following holds*

$$O_m^l(\mathbf{x}-\mathbf{y}) = \sum_{j=0}^{+\infty} \sum_{k=-j}^{j} (-1)^j O_{m+j}^{l+k}(\mathbf{x}) I_j^{-k}(\mathbf{y})$$

$\forall (m,l) \in \mathbb{N}_Z$.

Theorem 2.2.4. *(Inner-to-Inner Laplace translation theorem [116])*

$$I_m^l(\mathbf{x}-\mathbf{y}) = \sum_{j=0}^{m} \sum_{k=-j}^{j} (-1)^j I_{m-j}^{l-k}(\mathbf{x}) I_j^k(\mathbf{y})$$

$\forall (m,l) \in \mathbb{N}_Z$.

Following Thm. 2.2.2, we have

$$\begin{aligned}\sum_{\mathbf{y} \in Y} \frac{q(\mathbf{y})}{|\mathbf{x}-\mathbf{y}|} &= \sum_{\mathbf{y} \in Y} \sum_{m=0}^{+\infty} \sum_{l=-m}^{m} (-1)^m O_m^l(\mathbf{x}) I_m^{-l}(\mathbf{y}) q(\mathbf{y}) \\ &= \sum_{m=0}^{+\infty} \sum_{l=-m}^{m} (-1)^m O_m^l(\mathbf{x}) \sum_{\mathbf{y} \in Y} I_m^{-l}(\mathbf{y}) q(\mathbf{y}).\end{aligned}$$

Because the convergence of the *truncated* multipole expansions deteriorates near the singularity, this sum cannot be directly applied: the MAC is needed to control its convergence. However, following the FMM algorithm, this formula may be applied on restrictions of Y, namely on the particles of a leaf cell. We list in the following the different FMM operators for the 3D Laplace kernel, that are obtained using Thms. 2.2.3 and 2.2.4.

Definition 2.2.2. *(3D Laplace **P2M**) Let s be a cell and $q : Y \cap s \to \mathbb{C}$, where $Y \cap s$ is the (finite) set of particles contained in s. The **P2M** operator in s is the application $\boldsymbol{P2M}[s] : (Y \cap s \to \mathbb{C}) \to (\mathbb{N}_Z \to \mathbb{C})$ such that $\forall (j,k) \in \mathbb{N}_Z$*

$$(\boldsymbol{P2M}[s]\ q)\,(j,k) := (-1)^j \sum_{\mathbf{y} \in Y \cap s} I_j^k(\mathbf{y} - ctr(s)) q(\mathbf{y}).$$

The result of the application of a **P2M** operator is a multipole expansion. The multipole expansions are aggregated together to form multipole expansions in all the non-leaf cells of the source tree by using the **M2M** operator.

Definition 2.2.3. *(3D Laplace **M2M**) Let s be a cell and $q : \mathbb{N}_Z \to \mathbb{C}$. The **M2M** operator from s' to s is the application $\boldsymbol{M2M}[s,s'] : (\mathbb{N}_Z \to \mathbb{C}) \to (\mathbb{N}_Z \to \mathbb{C})$ such that $\forall (j,k) \in \mathbb{N}_Z$*

$$(\boldsymbol{M2M}[s,s'] \cdot q)\,(j,k) := \sum_{m=0}^{j} \sum_{\substack{l=-m \\ |k-l| \leq j-m}}^{m} I_{j-m}^{k-l}(ctr(s) - ctr(s')) q(l,m).$$

The multipole expansion in s is then equal to

$$\sum_{s' \in Sons(s)} \mathbf{M2M}[s,s'] \cdot \mathcal{M}[s']$$

where $\mathcal{M}[s']$ is *the* multipole expansion in s'.

Definition 2.2.4. *(3D Laplace **M2L**) Let t and s be two well-separated cells and $q : \mathbb{N}_Z \to \mathbb{C}$. The **M2L** operator from s to t is the application $\boldsymbol{M2L}[t,s] : (\mathbb{N}_Z \to \mathbb{C}) \to (\mathbb{N}_Z \to \mathbb{C})$ such that*

$$(\boldsymbol{M2L}[t,s] \cdot q)\,(j,k) := \sum_{m=0}^{+\infty} \sum_{l=-m}^{m} O_{j+m}^{-k-l}(ctr(t) - ctr(s)) q(m,l).$$

The result of such an application of a **M2L** operator is a local expansion in t.

Definition 2.2.5. *(3D Laplace **L2L**) Let t be a cell and $p : \mathbb{N}_Z \to \mathbb{C}$. The **L2L** operator from s to t is the application $\boldsymbol{L2L}[t',t] : (\mathbb{N}_Z \to \mathbb{C}) \to (\mathbb{N}_Z \to \mathbb{C})$ such that*

$$(\boldsymbol{L2L}[t',t] \cdot p)\,(j,k) := \sum_{m=j}^{+\infty} \sum_{\substack{l=-m \\ |l-k| \leq m-j}}^{m} I_{m-j}^{l-k}(ctr(t') - ctr(t)) p(m,l).$$

Definition 2.2.6. *(3D Laplace **L2P**) Let t be a leaf cell and $p : \mathbb{N}_Z \to \mathbb{C}$. The **L2P** operator in t is the application $\boldsymbol{L2P}[t] : (\mathbb{N}_Z \to \mathbb{C}) \to (X \cap t \to \mathbb{C})$ such that $\forall \mathbf{x} \in t \cap X$*

$$(\boldsymbol{L2P}[t] \cdot p)\,(\mathbf{x}) := \sum_{j=0}^{+\infty} \sum_{k=-j}^{j} I_j^k(\mathbf{x} - ctr(t)) p(k,j). \tag{2.13}$$

Our notation choice is justified by the "chain" appearing when one wants to look at the influence of the far field generated by $s' \in Sons(s)$ on $t' \in Sons(t)$ such that $MAC(t,s) = true$. Indeed, this influence is approximated in the FMM algorithm by

$$\mathbf{L2P}[t'] \cdot \mathbf{L2L}[t',t] \cdot \mathbf{M2L}[t,s] \cdot \mathbf{M2M}[s,s'] \cdot \mathbf{P2M}[s'] \cdot q_{|s'}$$

where $q_{|s'}$ denotes the vector of charges of the source particles in s'.

Operators for the 3D Helmholtz kernel. The same form of operators can be obtained when the Helmholtz kernel in the high frequency regime is considered, but based on different mathematical tools. The results of this paragraph can be found in [70] and the formulas on special functions can be found in [7]. The FMM described in a continuous way in this paragraph is known, after discretization, as *hf-fmm*.

Theorem 2.2.5. *(Gegenbauer's addition theorem [70]) Let j_l be the spherical Bessel function of order l ([178] Chap. 10 paragraph 47(ii)), P_l be the Legendre polynomial of order l and $h_l^{(1)}$ be the first kind spherical Hankel function of order l ([178] Chap. 10 paragraph 47(ii)). The following holds:*

$$\frac{e^{i\kappa|\mathbf{t}+\mathbf{z}|}}{|\mathbf{t}+\mathbf{z}|} = i\kappa \sum_{l=0}^{+\infty} (-1)^l (2l+1) j_l(\kappa|\mathbf{z}|) h_l^{(1)}(\kappa|\mathbf{t}|) P_l\left(\left\langle \frac{\mathbf{t}}{|\mathbf{t}|}, \frac{\mathbf{z}}{|\mathbf{z}|} \right\rangle\right).$$

To develop a multipole method, $\mathbf{t}$ has to be understood as the difference between the center of interacting source and target cells and $\mathbf{z}$ as the difference between the coordinates of the source and target particles given relatively to the center of the cell in which they lie. To be more precise, considering $\mathbf{x} \in X$ and $\mathbf{y} \in Y$, X and Y being the target and source point clouds respectively, and two cells s and t such that $\mathbf{x} \in s$ and $\mathbf{y} \in t$:

$$\begin{aligned} \mathbf{x} - \mathbf{y} &= [(\mathbf{x} - ctr(t)) - (\mathbf{y} - ctr(s))] + [(ctr(t) - ctr(s))] \\ &=: \mathbf{z} + \mathbf{t} \end{aligned}$$

with $ctr(s)$ being the center of s and $ctr(t)$ the center of t. We need another result to derive the continuous algorithm for the Helmholtz kernel.

Theorem 2.2.6. *(Propagating plane wave expansion [70])*

$$4\pi i^l j_l(\kappa|\mathbf{z}|) P_l\left(\langle \mathbf{t}/|\mathbf{t}|, \mathbf{z}/|\mathbf{z}| \rangle\right) = \int_{\mathbb{S}^2} e^{i\kappa\langle \mathbf{z}, \lambda\rangle} P_l(\langle \mathbf{t}/|\mathbf{t}|, \lambda\rangle) d\lambda$$

$\mathbb{S}^2 := \{\mathbf{x} \in \mathbb{R}^3 \mid |\mathbf{x}| = 1\}$ *denoting the unit sphere.*

By combining Thms. 2.2.5 and 2.2.6, we obtain (see [184])

$$\frac{e^{i\kappa|\mathbf{t}+\mathbf{z}|}}{|\mathbf{t}+\mathbf{z}|} = \lim_{n\to\infty} \frac{i\kappa}{4\pi} \int_{\mathbb{S}^2} \mathcal{T}_n(\mathbf{t}, \lambda) e^{i\kappa\langle \mathbf{z}, \lambda\rangle} d\lambda \tag{2.14}$$

where we used

$$\mathcal{T}_n(\mathbf{t}, \lambda) := \sum_{l=0}^{n} (2l+1) h_l^{(1)}(\kappa|\mathbf{t}|) P_l(\langle \mathbf{t}/|\mathbf{t}|, \lambda\rangle). \tag{2.15}$$

The complex exponential can then be decomposed, giving for any $\epsilon > 0$ the existence of $n \in \mathbb{N}$ such that for any cell s:

$$\begin{aligned} \sum_{\mathbf{y}\in s\cap Y} G(\mathbf{x}, \mathbf{y}) q(\mathbf{y}) &= \sum_{\mathbf{y}\in s\cap Y} \left(\frac{i\kappa}{4\pi} \int_{\mathbb{S}^2} \mathcal{T}_n(\mathbf{t}, \lambda) e^{i\kappa\langle \hat{\mathbf{x}} - \hat{\mathbf{y}}, \lambda\rangle} d\lambda \right) q(\mathbf{y}) + \mathcal{O}(\epsilon) \\ &= \frac{i\kappa}{4\pi} \int_{\mathbb{S}^2} \mathcal{T}_n(\mathbf{t}, \lambda) e^{i\kappa\langle \hat{\mathbf{x}}, \lambda\rangle} \sum_{\mathbf{y}\in s\cap Y} e^{-i\kappa\langle \hat{\mathbf{y}}, \lambda\rangle} q(\mathbf{y}) d\lambda + \mathcal{O}(\epsilon) \end{aligned}$$

which draws the shape of the associated FMM algorithm. The next definitions provide expressions for the *hf-fmm* operators.

Definition 2.2.7. *(hf-fmm **P2M**) Let s be a cell, $Y \cap s$ be the (finite) set of particles in s and $q : Y \cap s \to \mathbb{C}$. The **P2M** operator in s is defined by $\boldsymbol{P2M}[s] : (Y \cap s \to \mathbb{C}) \to (\mathbb{S}^2 \to \mathbb{C})$ such that $\forall \lambda \in \mathbb{S}^2$*

$$\left(\boldsymbol{P2M}[s] \cdot q\right)(\lambda) := \sum_{\mathbf{y}\in s\cap Y} e^{-i\kappa\langle \hat{\mathbf{y}}, \lambda\rangle} q(\mathbf{y}).$$

Definition 2.2.8. *(hf-fmm **M2M**) Let s be a cell and $q : \mathbb{S}^2 \to \mathbb{C}$. The **M2M** operator from $s' \in Sons(s)$ to s is defined by $\boldsymbol{M2M}[s, s'] : \left(\mathbb{S}^2 \to \mathbb{C}\right) \to \left(\mathbb{S}^2 \to \mathbb{C}\right)$ such that $\forall \lambda \in \mathbb{S}^2$*

$$\left(\boldsymbol{M2M}[s, s'] \cdot q\right)(\lambda) := e^{-i\kappa\langle ctr(s) - ctr(s'), \lambda\rangle} q(\lambda).$$

Definition 2.2.9. *(hf-fmm **M2L**) Let t and s be two well-separated cells and $q : \mathbb{S}^2 \to \mathbb{C}$. The **M2L** operator from s to t is defined by $\boldsymbol{M2L}[t, s] : \left(\mathbb{S}^2 \to \mathbb{C}\right) \to \left(\mathbb{S}^2 \to \mathbb{C}\right)$ such that $\forall\ \lambda \in \mathbb{S}^2$*

$$\left(\boldsymbol{M2L}[t, s] \cdot q\right)(\lambda) := \mathcal{T}_n(ctr(t) - ctr(s), \lambda) q(\lambda).$$

Definition 2.2.10. *(hf-fmm **L2L**) Let t be a cell, $t' \in Sons(t)$ and $p : \mathbb{S}^2 \to \mathbb{C}$. The **L2L** operator from s to t is defined by $\boldsymbol{L2L}[t', t] : \left(\mathbb{S}^2 \to \mathbb{C}\right) \to \left(\mathbb{S}^2 \to \mathbb{C}\right)$ such that $\forall\ \lambda \in \mathbb{S}^2$*

$$\left(\boldsymbol{L2L}[t', t] \cdot p\right)(\lambda) := e^{i\kappa\langle ctr(t') - ctr(t), \lambda\rangle} p(\lambda).$$

Definition 2.2.11. *(hf-fmm **L2P**) Let t be a leaf cell and $p : \mathbb{S}^2 \to \mathbb{C}$. The **L2P** operator in t is defined by $\boldsymbol{L2P}[t] : \left(\mathbb{S}^2 \to \mathbb{C}\right) \to (X \cap t \to \mathbb{C})$ such that $\forall\ \mathbf{x} \in X \cap t$*

$$\left(\boldsymbol{L2P}[t] \cdot p\right)(\mathbf{x}) := \int_{\mathbb{S}^2} e^{i\kappa\langle \mathbf{x} - ctr(t), \lambda\rangle} p(\lambda) d\lambda. \tag{2.16}$$

As for the Laplace case, the influence of the far field generated by $s' \in Sons(s)$ on $t' \in Sons(t)$ such that $MAC(t, s) = true$ is approximated in *hf-fmm* using the following "chain":

$$\mathbf{L2P}[t'] \cdot \mathbf{L2L}[t', t] \cdot \mathbf{M2L}[t, s] \cdot \mathbf{M2M}[s, s'] \cdot \mathbf{P2M}[s'] \cdot q_{|s'}.$$

After discretization of the integral, the expressions of the **M2M** and **L2L** operators have to be reformulated, which will be explained in detail in Chapter 3. This is a consequence of the varying cubature grids used among the tree levels, inducing an interpolation step on $\mathbb{S}^2$. *hf-fmm* is a special case of FMM in which the number of needed terms in the expansions grows with the size of the interacting cells to preserve a reasonable accuracy, although the diagonal expression of **M2L** operators [184] (see Def. 2.2.9) allows to evaluate these cell interactions efficiently. The overall complexity of *hf-fmm* is at least $\mathcal{O}(NlogN)$ on surfacic particle distributions [56] because of the increasing cost of the FMM operators from the leaves to the root of the 2^d-trees.

Low-frequency FMMs. The discretization and the numerical evaluation of special functions involved in Eq. 2.14 make this formula impossible to use for the low-frequency regime [135], that is when the product $C\kappa$ is less than a critical threshold, where C denotes the size of the computational domain. In many FMMs, this threshold is chosen so that there is no oscillation in the Helmholtz kernel in the low-frequency regime, that is $C\kappa \leq 1$ (see [174]). The classical approaches use *plane-wave expansions* in the low-frequency regime, adding an evanescent wave to the plane wave expansion of the kernel [114]. These methods are referred to as *lf-fmm* and have been studied in [78, 206].

Wideband FMM. The expansions used in *hf-fmm* becoming unstable in the low frequency regime while the use of the expansions of *lf-fmm* in the high-frequency regime is suboptimal (see [66]), the need of combining *lf-fmm* and *hf-fmm* is of important concerns. Indeed, the different cells of a given octree may lie in different regimes. The cells verifying $w\kappa \leq 1$, with w the size of the cell, are said to be in the low-frequency regime and those not verifying this inequality are said to be in the high-frequency regime. To handle both low and high frequency regimes, a standard approach ([60,66]) consists in adding a conversion between low-frequency and high-frequency expansions once the boundary between the two regimes is reached during the tree traversals. Another wideband FMM for the Helmholtz kernel with complex wavenumber is presented in [68].

2.2.5 Algebraic hierarchical methods

Algebraic generalizations of hierarchical algorithms have been widely studied and has led to the definition of a set of formats: panel-clustering methods [128, 129], $\mathcal{H}$-matrices [35, 123, 127], $\mathcal{H}^2$-matrices [124, 127], hierarchical semi-separable matrices [214]... The basic idea is to represent a N-body problem in a (dense) matrix form where particular blocks of the matrix can be approximated by low-rank factorizations. See also [28] for a presentation of hierarchical formats.

2.2.5.1 Matrix form

The starting point is to consider Eq. 2.1 as a matrix-vector product with a matrix $A \in \mathbb{C}^{M \times N}$ and the vector $\mathbf{q} \in \mathbb{C}^N$, where $M := \#X$ is the cardinal of X and $N := \#Y$ is the cardinal of Y. Indeed, denoting by $\mathbf{x}_k$ the k^{th} element of X and by $\mathbf{y}_l$ the l^{th} element of Y, we can define

$$A_{k,l} := G(\mathbf{x}_k, \mathbf{y}_l)$$

and

$$\mathbf{q}_l := q(\mathbf{y}_l)$$

such that Eq. 2.1 becomes equivalent to the computation of $\mathbf{p} \in \mathbb{C}^M$ such that

$$\mathbf{p} = A\mathbf{q}. \tag{2.17}$$

This formulation does not directly lead to a computational gain and the storage of the entire A requires $\mathcal{O}(MN)$ memory instead of $\mathcal{O}(N + M)$ for a direct computation (because only the source and target particles can be stored since the kernel G can be evaluated *on-the-fly*). However, the matrix form of the N-body problem allows to consider the acceleration methods from linear algebra viewpoint, which provides an interesting link between well-separated sets and low-rank approximations.

2.2.5.2 Low-rank matrices

Suppose that there exists $r \in \mathbb{N}^*$ such that

$$A = \sum_{n=1}^{r} u_n v_n^T$$

with $u_n \in \mathbb{C}^M$, $v_n \in \mathbb{C}^N$. We thus have

$$\begin{aligned} A\mathbf{q} &= \left(\sum_{n=1}^{r} u_n v_n^T \right) \mathbf{q} \\ &= \sum_{n=1}^{r} u_n \left(v_n^T \mathbf{q} \right) \end{aligned}$$

which costs $\mathcal{O}(r(M + N))$ operations to compute. If $r << min(M, N)$, this formulation results in a reduction of the cost of the matrix-vector product.

For arbitrary point clouds X and Y, the matrix A does not admit such a decomposition. However, the restrictions of A to well-separated groups of particles (i.e. groups of particles lying in pairs of cells that are well-separated with regard to a well chosen MAC, by analogy with the FMM) may be approximated using low-rank factorizations. These low-rank factorizations can be derived from the following theorem.

Theorem 2.2.7. *([125] Lemma C.5) Let $B \in \mathbb{C}^{M \times N}$. There exist unitary matrices $U \in \mathbb{C}^{M \times M}$ and $V \in \mathbb{C}^{N \times N}$ such that*

$$B = U \Sigma V \tag{2.18}$$

where $\Sigma \in (\mathbb{R}^+)^{M\times N}$ *such that* $\Sigma_{k,l} \neq 0 \Leftrightarrow k = l$, $\sigma_k := \Sigma_{k,k}$ *and* $\sigma_k \geq \sigma_{k+1}\ \forall k = 1, ..., min(M, N)-1$. *The* σ_k*'s named singular values of* B. *The factorization of Eq. 2.18 is referred to as the Singular Value Decomposition (SVD) of* B.

Because of the decaying of the singular values with regard to their index, a natural idea to obtain a low-rank approximation of an arbitrary matrix consists in replacing the singular values by zeros up to a given index. Such truncation is named a truncated SVD. The approximation truncation threshold can be estimated by means of the Eckart-Young-Mirsky theorem.

Theorem 2.2.8. *(Eckart-Young-Mirsky theorem [113] Th. 2.5.3) Let* $B^{(r)}$ *be the truncated SVD of* B *keeping the first* r *singular values of* B. *The following hold:*

$$\min_{C\in\mathbb{C}^{M\times N}\,|\,rank(C)=r} ||B - C||_2 = ||B - B^{(r)}||_2 = \sigma_{r+1},$$

$$\min_{C\in\mathbb{C}^{M\times N}\,|\,rank(C)=r} ||B - C||_F = ||B - B^{(r)}||_F = \sqrt{\sum_{j=r+1}^{min(M,N)} \sigma_j^2},$$

where $||\cdot||_2$ *denotes the spectral norm and* $||\cdot||_F$ *denotes the Frobenius norm.*

The SVD can be explicitly computed numerically in order to produce optimal low-rank approximations, but there exists several alternatives in practice, less costly in terms of numerical resources such as Rank-Revealing QR [62] or randomized techniques [130]. Popular methods in the BEM context are also able to compute in a fast manner the low-rank approximations: these are detailed in Chap. 4.

2.2.5.3 Hierarchical matrices

The hierarchical matrix format ($\mathcal{H}$-matrix) only consists in applying low-rank approximations of blocks of A that correspond to well-separated sets. This results in a $\mathcal{O}(r(NlogN + MlogM))$ algorithm for both application and storage, where r denotes the maximal rank of the low-rank approximations.

The $\mathcal{H}$-matrices usually rely on tree decompositions of the point clouds and a traversal of these trees to determine the blocks of the global matrix that need to be compressed. An optimization of the $\mathcal{H}$-matrix format is obtained by taking into account the links between the cells of the target and source trees. This last format is named $\mathcal{H}^2$-matrices and can be interpreted as an algebraic generalization of the FMM. In [48], a comparison between $\mathcal{H}$-matrices and FMM is described in the particular case of two Helmholtz problems (in low and high frequency regimes). Precomputation timings are better using the FMM and the memory consumption is in favor of the FMM. Hence, because the results show better application timings for the $\mathcal{H}$-matrices than the FMM, the number of needed matrix-vector products to obtain overall better timings with the $\mathcal{H}$-matrices are given: at least a few hundred applications.

Remark 2.2.7. *In its purely algebraic form, there is no constraint in the type of tree representation of the computational domain in the* $\mathcal{H}$*-matrix format. Hence, this tree does not have to be a* 2^d*-tree and can be, in practice, a binary tree (see for instance the code htool [2, 169]).*

Remark 2.2.8. *The rank of the blocks corresponding to cells in the high-frequency regime for oscillatory kernels increases with the frequency. Hence, the rank of these blocks tends to be proportional to their sizes (see [66]).*

Chapter 3

Kernel-explicit Fast Multipole Methods for highly oscillatory problems

Contents

In this chapter, we present the discretization of the Gegenbauer formula (Th. 2.2.5) exploited in the kernel-explicit FMM for the Helmholtz kernel in the high-frequency regime (that is the method *hf-fmm*). This leads to a high level discussion on the cubature problem on the sphere and to a presentation of the product rules used in practice in the *hf-fmm* context. We also present the difficulties of the treatment of the polynomial interpolation over the sphere in the multilevel algorithm appearing as a consequence of these cubatures over the sphere in Sect. 3.1. A set of fast algorithms designed to tackle the complexity of this interpolation is described in Sect. 3.2.

3.1 Discretization

To numerically deal with the kernel expansion in Eq. 2.14

$$\frac{e^{i\kappa|\mathbf{t}+\mathbf{z}|}}{|\mathbf{t}+\mathbf{z}|} = \lim_{n\to\infty} \frac{i\kappa}{4\pi} \int_{\mathbb{S}^2} \mathcal{T}_n(\mathbf{t},\lambda) e^{i\kappa\langle \mathbf{z},\lambda\rangle} d\lambda,$$

two different approximations are needed. First, as we already discussed in Sect. 2.2.4.3, one has to fix an integer $L \in \mathbb{N}^*$ and to remove the limit, keeping $\mathcal{T}_L$ only. In addition, the integral over the unit sphere cannot be directly computed and needs to be approximated through cubatures (that are quadratures in more than one dimension). This leads to an approximation of restrictions of the N-body problem with Helmholtz kernel in Eq. 2.1 of the form

$$\sum_{\mathbf{y}\in s\cap Y} \frac{e^{i\kappa|\mathbf{x}-\mathbf{y}|}}{|\mathbf{x}-\mathbf{y}|} q(\mathbf{y}) \approx \frac{i\kappa}{4\pi} \sum_{p=1}^{Q_L} \omega_p e^{i\kappa\langle \hat{\mathbf{x}},\lambda_p\rangle} \mathcal{T}_L(\mathbf{t},\lambda_p) \sum_{\mathbf{y}\in s\cap Y} e^{-i\kappa\langle \hat{\mathbf{y}},\lambda_p\rangle} q(\mathbf{y}) \tag{3.1}$$

for any $\mathbf{x} \in t$ where t, s are well-separated cells with regard to the strict MAC of Sect. 2.2.4.1, using $\hat{\mathbf{x}} := \mathbf{x} - ctr(t)$, $\hat{\mathbf{y}} := \mathbf{y} - ctr(s)$ and where the set $\{(\omega_1,\lambda_1),...,(\omega_{Q_L},\lambda_{Q_L})\} \subset \mathbb{R}\times\mathbb{S}^2$ refers to a cubature rule over the unit sphere. The definition of cubature, along with examples of usual cubature rules, are provided in Sect. 3.1.1. The discretization of the integral in Eq. 3.1 affects the definition of the *hf-fmm* operators. In Sect. 3.1.2, we present how the cubature rule is chosen according to the requested accuracy for the discretized **M2L** operator and in Sect. 3.1.3, we discuss the impact of the cubature on the **M2M**/**L2L** operators.

3.1.1 Cubature on the sphere

The design of accurate cubature rules with a minimal number of cubature nodes is still nowadays a very hard task for complex domains. A survey of the general cubature problem is provided in [71] and the principal methods on the sphere are compared in [37]. We first recall how this cubature problem writes on the sphere.

3.1.1.1 Spherical harmonics

The set of polynomials on the sphere corresponds to the set of spherical harmonics. A cubature rule of order L and size Q_L on the sphere refers to a finite set of $Q_L \in \mathbb{N}^*$ pairs formed by a real scalar (weight) and a point on the sphere (cubature node), i.e. a set $\{(\omega_q,\lambda_q) \in \mathbb{R}\times\mathbb{S}^2 \,|\, q \in [\![1,Q_L]\!]\}$ such that

$$\forall l \in [\![0,L]\!], m \in [\![-l,l]\!], \int_{\mathbb{S}^2} Y_l^m(\lambda) d\lambda = \sum_{q=1}^{Q_L} \omega_q Y_l^m(\lambda_q).$$

In other terms, such rule integrates exactly all the spherical harmonics with orders and degrees up to L by using Q_L points. Giving this Q_L number of points in a cubature rule and the location of the cubature nodes, one may find the weights by solving a Vandermonde-like system of the form

$$\begin{bmatrix} Y_0^0(\lambda_1) & \dots & Y_0^0(\lambda_{Q_L}) \\ Y_1^{-1}(\lambda_1) & \dots & Y_1^{-1}(\lambda_{Q_L}) \\ Y_1^0(\lambda_1) & \dots & Y_1^0(\lambda_{Q_L}) \\ Y_1^1(\lambda_1) & \dots & Y_1^1(\lambda_{Q_L}) \\ \vdots & \vdots & \vdots \\ Y_L^L(\lambda_1) & \dots & Y_L^L(\lambda_{Q_L}) \end{bmatrix} \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_Q \end{bmatrix} = \begin{bmatrix} \int_{\mathbb{S}_2} Y_0^0(\lambda) d\lambda \\ \int_{\mathbb{S}_2} Y_1^{-1}(\lambda) d\lambda \\ \int_{\mathbb{S}_2} Y_1^0(\lambda) d\lambda \\ \int_{\mathbb{S}_2} Y_1^1(\lambda) d\lambda \\ \vdots \\ \int_{\mathbb{S}_2} Y_L^L(\lambda) d\lambda \end{bmatrix} = \begin{bmatrix} \sqrt{4\pi} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

which is well-known to be highly ill-conditioned, the second equality being obtained by using $\int_{\mathbb{S}^2} Y_l^m(\lambda) Y_{l'}^{m'}(\lambda) d\lambda = \delta_{m,m'}\delta_{l,l'}$ and $Y_0^0 \equiv \frac{1}{\sqrt{4\pi}}$. Finding a cubature rule by solving a Vandermonde's system assumes that the cubature nodes are already known, which is often not the case.

In practice, the problem of designing a cubature rule can be simplified, relying on the properties of the integration domain.

3.1.1.2 Product of uniform rules

The most naive way to obtain a cubature on the sphere is to consider products of one-dimensional quadrature in both spherical angles (see for instance [26]). Indeed, the analytic expression of the spherical harmonics gives:

$$\begin{aligned}\int_{\mathbb{S}^2} Y_l^m(\lambda)d\lambda &= \int_{\theta\in[-\frac{\pi}{2},\frac{\pi}{2}]}\int_{\phi\in[0,2\pi]}\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(cos(\theta))e^{im\phi}d\theta d\phi \\ &= \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}\int_{\theta\in[-\frac{\pi}{2},\frac{\pi}{2}]}P_l^m(cos(\theta))d\theta\int_{\phi\in[0,2\pi]}e^{im\phi}d\phi\end{aligned} \tag{3.2}$$

where P_l^m denotes the associated Legendre function. Then, by using

$$\begin{cases}\phi_j & := \frac{2\pi}{2L+1}j \quad \forall\, 0\leq j\leq 2L \\ \omega_{\phi_j} & := \frac{2\pi}{2L+1}\end{cases}$$

we obtain [197]

$$\sum_{0\leq j\leq 2L}\omega_{\phi_j}e^{im\phi_j} = \begin{cases}2\pi \ \textit{iff}\ m=0; \\ 0 \ \textit{otherwise.}\end{cases}$$

Such pairs (ω_{ϕ_j},ϕ_j) define an appropriate quadrature rule over the azimuthal axis, i.e. a quadrature of the red integral in Eq. 3.2. The blue integral can also be evaluated exactly for $m=0$ (which is the only term that is not removed by a product with the quadrature of the red integral in Eq. 3.2) by using a rule $\{(\omega_{\theta_k},\theta_k)\}_k$ with $2L+1$ uniformly sampled points (uniform rule). The weights of this rule have to be found by solving, once again, a Vandermonde-like system:

$$\begin{bmatrix}P_0(cos(\theta_1)) & \dots & P_0(cos(\theta_{2L+1})) \\ \vdots & \ddots & \vdots \\ P_{2L}(cos(\theta_1)) & \dots & P_{2L}(cos(\theta_{2L+1}))\end{bmatrix}\begin{bmatrix}\omega_{\theta_1}\\ \vdots \\ \omega_{\theta_{2L+1}}\end{bmatrix} = \begin{bmatrix}2\\0\\ \vdots \\0\end{bmatrix}. \tag{3.3}$$

The resulting product rule is obtained by combining these two uniform rules, giving

$$\begin{cases}\{(\theta_k,\phi_j) \ \mid \ k,j\in[\![1,2L+1]\!]\} \ \textit{(cubature nodes)} \\ \{\omega_{\theta_k}\omega_{\phi_j} \ \mid \ k,j\in[\![1,2L+1]\!]\} \ \textit{(cubature weights)}\end{cases}.$$

This generates a cubature rule that integrates exactly the spherical harmonics up to the order $2L$ with size $(2L+1)^2$.

3.1.1.3 Gauss-Legendre product rules

There exists a better way of integrating the Legendre polynomials up to a certain order than a uniform sampling. Following [203], one may replace the uniform sampling of the polar axis in the product uniform rules of Sect. 3.1.1.2 by a Gauss-Legendre formula with $L+1$ nodes. This generates a cubature rule on the sphere with size $(L+1)(2L+1)$ and that integrates exactly the spherical harmonics of order up to $2L$. This cubature rule will be refer to as the *product Gauss-Legendre* cubature rule. This last rule is used in many versions of *hf-fmm* [60, 66, 77, 184, 197].

3.1.2 M2L operator

After having applied the discretization summarized in Eq. 3.1, a *hf-fmm* **M2L** evaluation (see Sect. 2.2.4.3) consists in computing $\mathcal{T}_L(\mathbf{t},\lambda)q(\lambda)$ for any $(\omega,\lambda)\in\mathfrak{Q}\subset\mathbb{R}\times\mathbb{S}^2$, $\mathfrak{Q}$ being a cubature rule on the sphere, $\mathbf{t}$ being the difference between the interacting cell centers and q being a (discretized) mutlipole expansion (i.e. an application from $\mathfrak{Q}$ to $\mathbb{C}$). Let L be the truncation order of the limit in Eq. 3.1. This **M2L** operator is often said to be diagonal by analogy with its matrix expression:

$$\begin{bmatrix}\mathcal{T}_L(\mathbf{t},\lambda_1) & & \\ & \ddots & \\ & & \mathcal{T}_L(\mathbf{t},\lambda_{Q_L})\end{bmatrix}\begin{bmatrix}q(\lambda_1)\\ \vdots \\ q(\lambda_{Q_L})\end{bmatrix}.$$

The complexity in terms of multiplications/additions of the application of a single **M2L** matrix, assuming that the entries of the diagonal matrix are already known, is exactly equal to the number of cubature nodes, that is equal to Q_L. As a direct consequence, by minimizing the number of cubature nodes, one also minimizes the application cost of the **M2L** operators.

In practice, the truncation order L is chosen accordingly to the radii of the interacting cells (that are supposed to be equal since *hf-fmm* uses the strict MAC, see Sect. 2.2.4.1) in order to preserve a global accuracy for the overall method. This order L thus varies between the tree levels (meaning that the cubature grids also do). In particular, L has to be large enough to ensure the convergence of the Gegenbauer series but small enough to avoid the divergence of $\mathcal{T}_L$ at infinity (see [66]). A sufficient criterion for the convergence of the Gegenbauer series [56] consists in choosing $L>\kappa w$, where w is the diameter of the interacting cells. The choice of the parameter L has been studied in [54, 67, 70, 77]. A usual option [56, 69, 135] is given by

$$L\approx\kappa w+(1.8d_0)^{2/3}(\kappa w)^{1/3}$$

d_0 denoting the number of requested digits of accuracy. This heuristic choice for L illustrates the dependence with the levels, since the diameter of the cells are fixed by their tree level.

3.1.3 M2M/L2L operators

Because the cubature grids change at each level of the octrees, one needs to define a way of interpolating the data associated to a set of cubature nodes to another set. Indeed, the *hf-fmm* **M2M**/**L2L** operators in Sect. 2.2.4.3 were diagonal. However, this diagonal form only exists because the input and output sets (i.e. the entire unit sphere) were the same. Because these sets are different after the discretization of the integral on the unit sphere[1] as in Eq. 3.1, this diagonal form does not hold any more.

The (discretized) **M2M**/**L2L** operators are decomposed into two steps: a translation (Sect. 3.1.3.1) and an interpolation over the sphere (Sect. 3.1.3.2).

3.1.3.1 Translation

The first step of the **M2M** and **L2L** operators consists in decentering the cubature grid. Indeed, a cubature grid is attached to a given cell, centered in the center of this cell. When moving the data from a cell to another one, a shifting has to be operated (the positions of $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are given relatively to the centers of interacting cells in Eq. 3.1). This can be done by applying a diagonal matrix product with complex exponentials to the multipole/local expansion. Let $\{(\omega_q,\lambda_q)\mid q\in[\![1,Q_L]\!]\}$ be a cubature grid associated to the cell c. Let $c'\in Sons(c)$ and q be a multipole expansion associated to c. The translation from c' to c in the **M2M** operator can be written as

$$\begin{bmatrix}e^{-i\kappa\langle ctr(c')-ctr(c),\lambda_1\rangle} & & \\ & \ddots & \\ & & e^{-i\kappa\langle ctr(c')-ctr(c),\lambda_{Q_L}\rangle}\end{bmatrix}\begin{bmatrix}q(\lambda_1)\\ \vdots \\ q(\lambda_{Q_L})\end{bmatrix}. \tag{3.4}$$

[1] Since the cubature rule depends on the tree level.

An equivalent formula is obtained for the **L2L** operator by reversing the roles of c' and c, i.e. by taking the conjugates of the entries of the matrix in Eq. 3.4. This shifting is a diagonal operation.

3.1.3.2 Interpolation between cubature grids

Once decentered, the expansions defined on a cubature grid of the incoming level have to be interpolated into expansions in the cubature grid of the new level (since the cubature nodes are not the same and the cubature rules have different size). This can be performed using Lagrange-like interpolation over the sphere (see [10]). We describe in the following how this interpolation is derived for band-limited functions in $L^2(\mathbb{S}^2)$ [197].

Let $f \in L^2(\mathbb{S}^2)$ be a band limited function with bandwidth L. We have

$$f(\lambda) = \sum_{l=0}^{L} \sum_{m=-l}^{l} A_{l,m} Y_l^m(\lambda)$$

where

$$A_{l,m} := \langle f, Y_l^m \rangle_{L^2(\mathbb{S}^2)}$$

where $\langle \cdot, \cdot \rangle_{L^2(\mathbb{S}^2)}$ refers to the scalar product in $L^2(\mathbb{S}^2)$.

Let $\mathfrak{Q}$ be a cubature rule on the sphere integrating exactly the integral in $A_{l,m}$. We obtain

$$\begin{aligned} f(\lambda) &= \sum_{l=0}^{L} \sum_{m=-l}^{l} \left(\sum_{(\omega_q, \lambda_q) \in \mathfrak{Q}} \omega_q f(\lambda_q) Y_l^{m*}(\lambda_q) \right) Y_l^m(\lambda) \\ &= \sum_{(\omega_q, \lambda_q) \in \mathfrak{Q}} \omega_q \left(\sum_{l=0}^{L} \sum_{m=-l}^{l} Y_l^m(\lambda) Y_l^{m*}(\lambda_q) \right) f(\lambda_q) \\ &= \sum_{(\omega_q, \lambda_q) \in \mathfrak{Q}} \omega_q \left(\sum_{l=0}^{L} \frac{2l+1}{4\pi} P_l(\langle \lambda_q, \lambda \rangle) \right) f(\lambda_q) \end{aligned}$$

and the red term will be refered to as the *interpolation operator on the sphere*, denoted by $\mathfrak{I}_{\mathbb{S}^2} : \mathbb{S}^2 \times \mathbb{S}^2 \to \mathbb{R}$, that is

$$\mathfrak{I}_{\mathbb{S}^2}(\mu, \lambda) = \sum_{l=0}^{L} \frac{2l+1}{4\pi} P_l(\langle \mu, \lambda \rangle).$$

The value of f on $\mu \in \mathbb{S}^2$ is given by

$$f(\mu) = \sum_{(\omega_q, \lambda_q) \in \mathfrak{Q}} \mathfrak{I}_{\mathbb{S}^2}(\mu, \lambda_q) \left(\omega_q f(\lambda_q) \right). \tag{3.5}$$

This allows to interpolate the values of any expansion on a cubature grid to another cubature grid by taking as values for μ all the nodes of the new cubature grid. The expression in Eq. 3.5 for all μ can trivially be expressed a matrix-vector product. The involved matrix is a dense one and its dimension is equal to the product of the number of nodes in the two involved cubature grids. Hence, this matrix-vector product is costly for large cubature rules, that is for high truncation orders L in the computation of $\mathcal{T}_L$. To be more precise, this cost increases from the leaves to the roots of the 2^d-trees because of the choice of L as discussed in Sect. 3.1.2.

3.2 Fast algorithms for the polynomial interpolation over the sphere

In practice, the naive application of the **M2M**/**L2L** operators as dense matrices becomes too prohibitive for the highest levels of the octrees (i.e. close to the root). To tackle this issue, many

approaches were proposed in the literature. We describe in the following paragraphs how these products act on the global complexity of *hf-fmm* (Sect. 3.2.1) and the main methods, to our knowledge, aiming at accelerating these products (Sect. 3.2.2).

3.2.1 Impact on the total complexity

The crucial point is that the cost of the **M2M**/**L2L** operations has, in *hf-fmm*, to be balanced with the cost of the **M2L** operations because of the varying ratio between **M2L** and **M2M**/**L2L** numbers at each level of the octrees and because the cost of these operators increases differently from the leaves to the root. If no optimization technique is used, the cost of a **M2M**/**L2L** operation at a given octree level associated to a truncation order L for the Gegenbauer series is $\mathcal{O}(L^4)$ and the one of a **M2L** operation at the same level is $\mathcal{O}(L^2)$. Hence, as mentioned for instance in [56], a naive application of such **M2M**/**L2L** operators leads to a global complexity of $\mathcal{O}(N^{\frac{2}{3}})$, that can be reduced to $\mathcal{O}(Nlog^2N)$ or $\mathcal{O}(NlogN)$ based on choices of fast interpolation algorithms on the sphere [56], where N denotes the number of source and target particles and where the particle distributions are supposed to be surfacic.

Near the deepest octree levels, the cubature grids usually only involve a low number of points (because of the small diameter of the involved cells and assuming that the leaves have a side length of approximately one wavelength). Only a single **M2M**/**L2L** operation is performed per cell, compared to up to 189 **M2L** per cell [76]. Hence, this **M2M**/**L2L** cost can be negligible at these deepest levels. On the opposite, as the level number diminishes, the number of involved cells also does, so there also are less **M2L** operations to apply at the highest levels. The (naive) **M2M**/**L2L** application cost may dominate the FMM evaluation cost on these levels. Fortunately, there exists a set of algorithms designed for the acceleration of the **M2M**/**L2L** operators.

3.2.2 Fast schemes

In the following paragraphs are listed fast methods to perform the interpolation over the sphere needed in the **M2M**/**L2L** operators (see Sect. 3.1.3.2). We denote by L the truncation order of the Gegenbauer formula as in Eq. 3.1.

3.2.2.1 Threshold method

A method that does not preserve the exactness of the interpolation but leads to an overall linearithmic complexity for *hf-fmm* (evaluating approximated interpolations over the sphere in $\mathcal{O}\left(L^2\right)$ flops) is presented in [75, 76]. The idea is to sparsify the **M2M**/**L2L** matrices by discarding all the entries that have a "small" module, according to a user chosen threshold. This somehow corresponds to a non-local operator approximation with a local one. Notice that the error induced by the sparsification has to be taken into account in the overall error estimate when using such a method.

3.2.2.2 Local Lagrange interpolation

Another local method can be obtained by using localized Lagrange interpolation on the sphere [98]. This technique also reduces the interpolation complexity to a quasi-linear one: $\mathcal{O}(L^2p^2)$, where p is the number of interpolation nodes in the neighborhood of a node in each of the spherical directions and is chosen to be small in practice. Once again, this method adds an additional approximation to the error induced by the truncation of the Gegenbauer series that has to be taken into account in the global accuracy estimate.

3.2.2.3 Jakob-Chien-Alpert's algorithm

This paragraph is dedicated to the fast algorithm proposed in [144]. Let us suppose that all the considered cubature grids are product grids (in spherical coordinates), that is product uniform (see

Sect. 3.1.1.2) or Gauss-Legendre (see Sect. 3.1.1.3) rules. The main point is that $\mathfrak{I}_{\mathbb{S}^2}$ can be written in term of spherical harmonics thanks to the spherical harmonic addition theorem ([178] Sect. 14.30(iii) Eq. 14.30.9):

$$\mathfrak{I}_{\mathbb{S}^2}(\mu,\lambda) = \sum_{l=0}^{L}\sum_{m=-l}^{l} Y_l^m(\mu)Y_l^{m*}(\lambda). \tag{3.6}$$

Denoting by (θ_u,ψ_u) the spherical angles of $u \in \{\mu,\lambda\}$, this form allows the exploitation of the product structure of the cubature grids. Let $Q_l^m(x) := \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(x)$, with P_l^m the associated Legendre polynomial indexed by (l,m) (see [7] Chap. 8). Let $q(\theta_\lambda,\psi_\lambda) := \omega f(\lambda)$ for any cubature node λ with associated weight ω in the cubature rule $\mathfrak{Q}$ (see Eq. 3.5), where $(\theta_\lambda,\psi_\lambda)$ are the spherical angles of λ expressed in spherical coordinates. Thanks to Eq. 3.6, we have:

$$\begin{aligned}
\sum_{(\omega,\lambda)\in\mathfrak{Q}} \mathfrak{I}_{\mathbb{S}^2}(\mu,\lambda)\omega f(\lambda) &= \sum_{\theta_\lambda}\sum_{\psi_\lambda}\sum_{l=0}^{L}\sum_{m=-l}^{l} e^{im\psi_\mu}Q_l^m(cos\theta_\mu)Q_l^m(cos\theta_\lambda)e^{-im\psi_\lambda}q(\theta_\lambda,\psi_\lambda) \\
&= \sum_{m=-L}^{L} e^{im\psi_\mu}\sum_{\theta_\lambda}\sum_{|m|\le l\le L} Q_l^m(cos\theta_\mu)Q_l^m(cos\theta_\lambda)\left(\sum_{\psi_\lambda} e^{-im\psi_\lambda}q(\theta_\lambda,\psi_\lambda)\right) \\
&= \sum_{m=-L}^{L} e^{im\psi_\mu}\sum_{\theta_\lambda} C_m(\theta_\mu,\theta_\lambda)\left(\sum_{\psi_\lambda} e^{-im\psi_\lambda}q(\theta_\lambda,\psi_\lambda)\right)
\end{aligned} \tag{3.7}$$

with $C_m(\theta_\mu,\theta_\lambda) := \sqrt{\frac{(L+1)^2-m^2}{4(L+1)^2-1}}\frac{Q_{L+1}^m(cos\theta_\lambda)Q_L^m(cos\theta_\mu)-Q_{L+1}^m(cos\theta_\mu)Q_L^m(cos\theta_\lambda)}{cos\theta_\lambda-cos\theta_\mu}$ obtained with the well-known Christoffel-Darboux formula (see [178] Sect. 18.2(v)). The algorithm is divided into three steps:

- for each m and each θ_λ, compute $F_m(\theta_\lambda) := \sum_{\psi_\lambda} e^{-im\psi_\lambda}q(\theta_\lambda,\psi_\lambda)$ using FFT,
- for each m, compute $G_m(\theta_\mu) := \sum_{\theta_\lambda} C_m(\theta_\mu,\theta_\lambda)F_m(\theta_\lambda)$,
- for each (θ_μ,ψ_μ), compute $\sum_{m=-L}^{L} e^{im\psi_\mu}G_m(\theta_\mu)$ using FFT.

The second step can be interpreted as dense matrix-vector products. This method is referred to as the *Jakob-Chien-Alpert's* algorithm. It is possible to accelerate this step using 1D FMMs [144], allowing an asymptotic cost reduction. This algorithm has a complexity of $\mathcal{O}(L^3)$ flops without 1D FMM, and a complexity of $\mathcal{O}(L^2 logL)$ with this optimization (asymptotically). One can notice that the use of 1D FMMs also adds a new error to the global *hf-fmm* scheme which has to be taken into account in the global accuracy estimate.

3.2.2.4 Fast Spherical Transforms

Combining Eq. 3.6 and Eq. 3.5, the interpolation on the sphere becomes

$$\begin{aligned}
f(\mu) &= \sum_{(\omega_q,\lambda_q)\in\mathfrak{Q}} \mathfrak{I}_{\mathbb{S}^2}(\mu,\lambda_q)\left(\omega_q f(\lambda_q)\right) \\
&= \sum_{l=0}^{L}\sum_{m=-l}^{l} Y_l^m(\mu)\sum_{(\omega_q,\lambda_q)\in\mathfrak{Q}} Y_l^{m*}(\lambda_q)\left(\omega_q f(\lambda_q)\right)
\end{aligned}$$

for any μ in the target cubature grid with a source cubature grid $\mathfrak{Q}$. This can be evaluated efficiently using the Fast Spherical Transforms (FST) [69]. This reduces the complexity of the interpolation on the sphere to $\mathcal{O}(L^2 log\ L)$. However, as mentioned in [56], the accuracy and stability of such a method still has to be studied. The overall complexity of *hf-fmm* using FST is $\mathcal{O}(N)$ for volumes and $\mathcal{O}(Nlog^2N)$ for surfaces.

3.2.2.5 Two-dimensional FFTs

In [56], using a tensorized Fourier basis in both spherical angles on trigonometric polynomial representation of $\mathcal{T}_L$, product uniform rules allow to use FFT in both spherical angles (instead of only one using the Alpert's algorithm). This leads to $\mathcal{O}(L^2 log\ L)$ interpolations on the sphere with numerical stability. The drawback of this method is that the number of involved cubature nodes is large compared to the Jakob-chien-Alpert's algorithm with Gauss-Legendre rules. Filtering techniques are used to reduce this number of points. The main advantage of this approach is that the interpolation on the sphere is fast (with complexity $\mathcal{O}\left(L^2 logL\right)$) and exact.

3.2.2.6 Ahrens-Beylkin's method

In [10] is presented a method dealing with cubature grids invariant under the action of the icosahedral group. As in the Jakob-chien-Alpert's algorithm, the Christoffel-Darboux formula is used after an application of the spherical harmonics addition's theorem to discard the sum over the spherical harmonics degrees. Unfortunately, because of the non-product structure of the considered grids, the factorization does not lead to the same asymptotic complexity than the Jakob-chien-Alpert's algorithm using 1D FMMs and requires $\mathcal{O}\left(L^3\right)$ operations. This algorithm is not an exact interpolation since 1D FMMs are used. One may notice that the article [10] also proposes, for the same grids and with the same complexity, an algorithm using Unequally Spaced Fast Fourier Transforms, based on an expression of the interpolation kernel on the sphere in terms of Fourier series [90,160].

Chapter 4

Kernel-independent Fast Multipole Methods for highly oscillatory problems

Contents

In this chapter we are interested in the kernel-independent FMM formulations and in particular in the polynomial interpolation based methods. We describe such methods and the techniques used to accelerate their running times in practice. This chapter also presents the general approach to deal with oscillatory kernels in the high-frequency regime in a kernel-independent formalism, namely the direction-based approaches.

With the deployment of FMMs to a large diversity of scientific areas (see Sect. 2.2.4), many kernel-independent formulations [57,94,95,103,164,170,218,219,224] have emerged in the literature. One of the most popular of these approaches, namely *kifmm*, is described in App. B.1 and diagonal kernel-idependent approaches are presented in App. B.2. In this chapter, we are mostly interested in the interpolation-based FMMs for oscillatory kernels and especially on the polynomial interpolation techniques allowing to derive a fast summation scheme. In Sect. 4.1, we introduce basic notions on multivariate polynomial interpolation and present a FMM formulation deriving from it. In Sect. 4.2, we describe the *directional* approach for the treatment of the highly oscillatory kernels, extending the kernel-independent interpolation-based FMM formulations to such kernels.

4.1 Interpolation-based FMMs

Interpolation techniques are exploited in many ways in hierarchical methods [89, 122, 217]. Deriving from multivariate polynomial interpolation, efficient hierarchical summation schemes can be obtained [51, 103, 111, 126].

Function interpolation using polynomials is a large topic (see [108] for a pedestrian survey of the multivariate polynomial interpolation problem). In this section, we restrict ourselves to the multivariate Lagrange problem. We are especially interested in the FMM formulations that can be derived from Lagrange interpolation.

4.1.1 Polynomial interpolation

Let $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{C}$ be two point clouds with N points each. Let $\mathbf{x}_i \in \mathcal{X}$ be the i^{th} element of $\mathcal{X}$ and $\mathbf{y}_i$ be the i^{th} element of $\mathcal{Y}$. Let Π^d be the space of d-variate polynomials. The *Lagrange interpolation problem* can be stated as

$$\textit{Given } V \subset \Pi^d \textit{ find } p \in V \textit{ such that } \forall i \in [\![1, N]\!], \;\; p(\mathbf{x}_i) = \mathbf{y}_i. \tag{4.1}$$

The $\mathbf{x}_i$'s are referred to as the *interpolation nodes*, $\mathcal{X}$ as the *interpolation grid* and p as the *interpolation polynomial*.

4.1.1.1 Lagrange formula

For $d = 1$ and any such $\mathcal{X} = \{x_1, ..., x_N\}, \mathcal{Y} = \{y_1, ..., y_N\}$, provided that $\mathcal{X}$ is composed of distinct points, an explicit form of the interpolation polynomial is known, following the *Lagrange interpolation formula*. Indeed, we have

$$p = \sum_{i=1}^{N} y_i S_i \in V := \Pi^1_{N-1}$$

where Π^1_{N-1} refers to the set of polynomials of degree at most $N-1$ and

$$S_i(x) := \prod_{\substack{j=1 \\ j \neq i}}^{N} \frac{x - x_j}{x_i - x_j}.$$

The term S_i is referred to as the *Lagrange polynomial* associated to the interpolation node x_i. Since for such a fixed pair $(\mathcal{X}, \mathcal{Y})$ the interpolation polynomial is unique, we can talk about *the* interpolation polynomial p and of *the* Lagrange polynomials on the grid $\mathcal{X}$.

4.1.1.2 Function interpolation and Lebesgue constant

Let $f \in \mathcal{C}^0([-1,1])$ be a continuous function defined on $[-1,1]$ and $\mathcal{X} \subset [-1,1]$ be a finite point cloud (still composed of N points). The polynomial interpolation of f on $\mathcal{X}$, denoted by $I_{\mathcal{X}}[f] \in V = \Pi^1_{N-1}$, is the solution of the Lagrange interpolation problem with $\mathcal{X}$, $\mathcal{Y} := \{f(x_i) \,|\, i \in [\![1, N]\!]\}$. A direct application of the Lagrange formula gives

$$I_{\mathcal{X}}[f](x) = \sum_{i=1}^{N} f(x_i) S_i(x)$$

that solves the interpolation problem in Eq. 4.1. Using these notations, $I_{\mathcal{X}}$ denotes a mapping from $\mathcal{C}^0([-1,1])$ to $\Pi^1_{N-1}([-1,1])$ the set of polynomials of degree at most $N-1$ on $[-1,1]$ (actually $I_{\mathcal{X}}$ is linear and is a projection).

Definition 4.1.1. *Let $f \in \mathcal{C}^0([-1,1])$. The uniform norm of f, denoted by $||f||_{\infty,[-1,1]}$, is defined by the following formula:*

$$||f||_{\infty,[-1,1]} := \sup_{x \in [-1,1]} |f(x)| = \max_{x \in [-1,1]} |f(x)|.$$

Now let the uniform norm of $I_{\mathcal{X}}$ on $[-1,1]$ be defined by

$$\Lambda(\mathcal{X}) := ||I_{\mathcal{X}}||_{\infty,[-1,1]} = \max_{x \in [-1,1]} \sum_{i=1}^{N} |S_i(x)|.$$

A result on the accuracy of the interpolation process can be derived and is expressed in Prop. 4.1.1.

Proposition 4.1.1. *([141]) $||f - I_{\mathcal{X}}[f]||_{\infty,[-1,1]} \leq (\Lambda(\mathcal{X}) + 1)||f - p^*||_{\infty,[-1,1]}$, where $p^* \in \Pi^1_{N-1}$ is the best uniform polynomial approximation of f in Π^1_{N-1}.*

$\Lambda(\mathcal{X})$ is called the *Lebesgue constant* and its growth rate with regard to the number of interpolation nodes (that is with regard to the cardinal of $\mathcal{X}$) gives an idea on the interpolation process stability when using the grid $\mathcal{X}$.

A well known example of quickly diverging Lebesgue constant is given by uniform samplings (equispaced interpolation nodes on $[-1,1]$). The interpolation of functions on equispaced nodes may diverge with regard to the number of interpolation orders (i.e. the number of interpolation nodes). Regularization methods may thus be needed when trying to deal with equispaced nodes. However, one may keep in mind that, even if the interpolation may not converge uniformly on equispaced grids (with an error growing exponentially on the endpoints, corresponding to the *Runge phenomenon*), the Lebesgue constant only gives an *upper bound* on the interpolation error.

4.1.1.3 Chebyshev polynomials

The Chebyshev polynomials (of the first kind) are polynomials defined as in Def. 4.1.2.

Definition 4.1.2. *Let T_n be the trigonometric polynomial such that*

$$T_n(x) = cos(n\, acos(x)), \qquad \forall x,\ |x| \leq 1.$$

Such T_n is called a Chebyshev polynomial (of the first kind) of order n.

The set of roots of any given Chebyshev polynomial is known and its elements are referred to as *Chebyshev nodes*. The roots of T_n on $[-1,1]$ are given by

$$\left\{ cos\left(\frac{2k-1}{2n} \pi \right) \,|\, k \in [\![1, n]\!] \right\}.$$

The Lagrange polynomials on Chebyshev nodes are given by

$$S_l(x) := \frac{1}{n} + \frac{2}{n}\sum_{k=1}^{n-1} T_k(x)T_k(x_l)$$

where x_l is the l^{th} Chebyshev node (see for instance [41, 172]). Such interpolation polynomials are widely used in numerical methods and the reason is expressed in Thm. 4.1.1.

Theorem 4.1.1. *([148]) For every absolutely continuous function f on $[-1,1]$, the sequence of interpolating polynomials constructed on Chebyshev nodes converges uniformly to f.*

4.1.1.4 Mappings

To work with interpolation rules defined on $[-1,1]$ on arbitrary intervals $[a,b]$, $a,b \in \mathbb{R}$, $a \leq b$, the simplest solution consists in introducing an invertible mapping $\gamma_{a,b} : [-1,1] \to [a,b]$ defined by

$$\gamma_{a,b}(x) = \left(\frac{a+b}{2} - a\right)x + \frac{a+b}{2} \tag{4.2}$$

and

$$\gamma_{a,b}^{-1}(x) = \frac{2x-b-a}{b-a}. \tag{4.3}$$

Hence, if the interpolation nodes $\mathcal{X}$ and the interpolation polynomials S_l's are defined on $[-1,1]$ and if $f : [a,b] \to \mathbb{C}$, the interpolation of f on $[a,b]$ can be derived from the polynomials and nodes on $[-1,1]$ by means of the mappings in Eqs. 4.2 and 4.3, following

$$I_{\mathcal{X}}[f](x) = \sum_{i=1}^{N} f(\gamma_{a,b}(x_i))S_i(\gamma_{a,b}^{-1}(x)).$$

4.1.1.5 Multivariate polynomial interpolation with tensorized 1D interpolation rules

A naive but convenient approach for solving the interpolation problem in Eq. 4.1, provided that the interpolation domain is the tensorization of intervals (that is a cuboid), consists in using a tensorized grid formed by 1D interpolation grids along each interval (see [108]). Let $C \subset \mathbb{R}^d$ a cuboid such that

$$C = [a_1, b_1] \times \ldots \times [a_d, b_d]$$

with $a_i \leq b_i$, $\forall i \in [\![1,d]\!]$. Let $\mathcal{X}_i =: \{x_l^{(i)}\}_l$, $\#\mathcal{X}_i =: N_i$ be a set of interpolation nodes in $[a_i,b_i]$ and $\{S_l^{(i)}\}_l$ be the associated interpolation polynomials. The tensorized rule is composed of nodes $\mathbf{x_l} \in \Pi_{i=1}^d \mathcal{X}_i$ for any multi-index $\mathbf{l} = (l_1, ..., l_d) \in \prod_{i=1}^{d} [\![1, N_i]\!]$ defined by

$$\mathbf{x_l} := (x_{l_1}^{(1)}, \ldots, x_{l_d}^{(d)})$$

and associated to the polynomials $S_\mathbf{l}$ such that, $\forall \mathbf{x} = (x_1, ..., x_d) \in C$, $\mathbf{l} = (l_1, ..., l_d)$

$$S_\mathbf{l}(\mathbf{x}) := \prod_{i=1}^{d} S_{l_i}^{(i)}(x_i).$$

$S_\mathbf{l}$ is referred to as the Lagrange polynomial associated to the node $\mathbf{x_l}$. Since the cells of any 2^d-tree as defined in Sect. 2.2.1.2 are cuboids, these tensorized interpolations rules can be used in the cells of a 2^d-tree. Notice that error bounds for the tensor product interpolation using Chebyshev rules are known [109, 112].

4.1.2 Black-box-FMM

For the reasons discussed in Sects. 4.1.1.3 and 4.1.1.5, in [103] has been proposed a FMM based on multivariate tensorized Chebyshev polynomials, called **b**lack-**b**ox-**FMM** (*bbfmm*). In this section, we first show how a multilevel fast multipole algorithm is derived from the polynomial interpolation (Sect. 4.1.2.1). We then describe the operators of the corresponding FMM formulation (Sect. 4.1.2.2).

4.1.2.1 Multilevel polynomial interpolation-based algorithm

The general idea exploited in *bbfmm* is to substitute the kernel function G on pairs of cells $(t,s) \subset \mathbb{R}^d \times \mathbb{R}^d$ in 2^d-trees that are well-separated with regard to the strict MAC (see Sect. 2.2.4.1) by its interpolation on $t \times s$. When using tensorized interpolation grids on $t \times s$, one obtains $2d$-dimensional multivariate Lagrange polynomials that are products of one-dimensional Lagrange polynomials (see Sect. 4.1.1.5). Each term of this product can be associated to t or s only, depending on the axis in $\mathbb{R}^{2d}$ that is covered by the corresponding one-dimensional interpolation rule. This divides the whole set of interpolation nodes into two parts: the multivariate nodes that are in t and those in s. Both of them can be considered leading to different interpolation rules with grids $\mathcal{K} := \{\mathbf{x}_1, ..., \mathbf{x}_{L^d}\}$ and $\mathcal{L} := \{\mathbf{y}_1, ..., \mathbf{y}_{L^d}\}$ respectively, assuming that the one-dimensional interpolation orders are all equal to $L \in \mathbb{N}\backslash\{0,1\}$. Hence, there exists a set of multivariate (d-dimensional) Lagrange polynomials associated to $\mathcal{K} \subset t$ only (we denote by T_k the polynomial associated to the node $\mathbf{x}_k$) and associated to $\mathcal{L} \subset s$ only (we denote by S_l the polynomial associated to the node $\mathbf{y}_l$). This gives the following approximation of G:

$$G(\mathbf{x},\mathbf{y}) \approx \sum_{\mathbf{x}_k \in \mathcal{K}} T_k(\mathbf{x}) \sum_{\mathbf{y}_l \in \mathcal{L}} G(\mathbf{x}_k,\mathbf{y}_l) S_l(\mathbf{y}) \tag{4.4}$$

$\forall (\mathbf{x},\mathbf{y}) \in t \times s$. The expression in Eq. 4.4 actually corresponds to a 3-terms approximation represented in Eq. 4.5:

$$G(\mathbf{x},\mathbf{y}) \approx \begin{bmatrix} T_1(\mathbf{x}) & \dots & T_{L^d}(\mathbf{x}) \end{bmatrix} \begin{bmatrix} G(\mathbf{x}_1,\mathbf{y}_1) & \dots & G(\mathbf{x}_1,\mathbf{y}_{L^d}) \\ \vdots & \ddots & \vdots \\ G(\mathbf{x}_{L^d},\mathbf{y}_1) & \dots & G(\mathbf{x}_{L^d},\mathbf{y}_{L^d}) \end{bmatrix} \begin{bmatrix} S_1(\mathbf{y}) \\ \vdots \\ S_{L^d}(\mathbf{y}) \end{bmatrix}. \tag{4.5}$$

Denoting by $Y_{|s}$ the particles of the source point cloud Y that are included in the cell s (i.e. the set $Y \cap s$), the approximation in Eq. 4.4 directly leads to

$$\sum_{\mathbf{y} \in Y_{|s}} G(\mathbf{x},\mathbf{y}) q(\mathbf{y}) \approx \sum_{\mathbf{x}_k \in \mathcal{K}} T_k(\mathbf{x}) \sum_{\mathbf{y}_l \in \mathcal{L}} G(\mathbf{x}_k,\mathbf{y}_l) \underbrace{\sum_{\mathbf{y} \in Y_{|s}} S_l(\mathbf{y}) q(\mathbf{y})}_{=:\mathcal{M}[s](\mathbf{y}_l)} \tag{4.6}$$

for any $\mathbf{x} \in t$. $\mathcal{M}[s]$ can be interpreted as an application defined on $\mathcal{L}$ giving the terms of the multipole expansion in s. Actually Eq. 4.6 defines a single level FMM.

A multilevel algorithm can then be obtained by observing that $\mathcal{M}[s]$ can be also considered as a continuous function of s (as a weighted sum of polynomials). Its interpolation on $s' \in Sons(s)$ can be performed. The same also holds for $\sum_{\mathbf{y} \in Y_{|s}} G(\mathbf{x},\mathbf{y}) q(\mathbf{y})$ as a function of $\mathbf{x}$, that can be interpolated in $t' \in Sons(t)$. Applying recursive interpolations, we finally come up with a hierarchical representation of the approximation of G on $t \times s$ that depends on the leaves in $Desc(t)$ and $Desc(s)$ (the descendants of t and s) and on the links between the elements of these two sets independently. This allows to describe the FMM operators as done in Sect. 4.1.2.2

4.1.2.2 *bbfmm* operators

In this section we present the general form of the $bbfmm$ operators as we did in Sect. 2.2.4.3 for different kernel-explicit formulations, but G refers here to any kernel (in the low-frequency regime).

P2M/L2P operators. Let s be a leaf. Let $q : Y_{|s} \to \mathbb{C}$ be a vector of charges on the particles of s. The **P2M** operator on s can be written as

$$\left(\mathbf{P2M}[s] \cdot q\right)(\mathbf{y}_l) := \sum_{\mathbf{y} \in Y_{|s}} S_l(\mathbf{y}) q(\mathbf{y})$$

$\forall \mathbf{y}_l \in \mathcal{L}$, where S_l is the Lagrange polynomial associated to $\mathbf{y}_l$ and $\mathcal{L}$ the interpolation grid on s. The red color is used to separate the elements depending on the particles from the one depending on the interpolation grid. The **L2P** operator on a leaf cell t is actually the transpose of the **P2M** one: instead of summing over the particles, the summation is applied on the interpolation polynomials (or equivalently on the interpolation nodes they are associated to). This can be written, for any local expansion defined on the interpolation grid $\mathcal{K}$ on t, i.e. any application $p : \mathcal{K} \to \mathbb{C}$, as

$$\left(\mathbf{L2P}[t] \cdot p\right)(\mathbf{x}) := \sum_{\mathbf{x}_k \in \mathcal{K}} T_k(\mathbf{x}) p(\mathbf{x}_k)$$

where $\mathbf{x}_k$ is the interpolation node in the grid $\mathcal{K} \subset t$ associated to the Lagrange polynomial T_k and $\mathbf{x} \in t$ is a particle.

M2M/L2L operators. The **M2M** operator is defined similarly than the **P2M** one, replacing the particles by the interpolation nodes of the son's interpolation grid. Let q be a multipole expansion in the cell $s' \in Sons(s)$, i.e. an application from the interpolation grid $\mathcal{L}' := \{\mathbf{y}'_1, ..., \mathbf{y}'_{L^d}\} \subset s'$ to $\mathbb{C}$. The **M2M** operator from the cell s' to s can expressed as

$$\left(\mathbf{M2M}[s, s'] \cdot q\right)(\mathbf{y}_l) := \sum_{\mathbf{y}'_u \in \mathcal{L}} S_l(\mathbf{y}'_u) q(\mathbf{y}'_u)$$

$\forall \mathbf{y}_l \in \mathcal{L}$, the interpolation grid in s, where S_l refers to the Lagrange polynomial associated to $\mathbf{y}_l$. The **L2L** operator is also obtained by transposing the **M2M** one. Hence, denoting by $\mathcal{K}' := \{\mathbf{x}'_1, ..., \mathbf{x}'_{L^d}\}$ the interpolation grid on the cell $t' \in Sons(t)$, by $\mathcal{K} := \{\mathbf{x}_1, ..., \mathbf{x}_{L^d}\}$ the interpolation grids in t and by p a local expansion in t, i.e. an application from $\mathcal{K}$ to $\mathbb{C}$, the **L2L** operator between t and t' can be written as

$$\left(\mathbf{L2L}[t', t] \cdot p\right)(\mathbf{x}'_u) := \sum_{\mathbf{x}_k \in \mathcal{K}} T_k(\mathbf{x}'_u) p(\mathbf{x}_k)$$

where T_k is the Lagrange polynomial associated to $\mathbf{x}_k$ and $\mathbf{x}'_u \in \mathcal{K}'$.

M2L operator. Let t and s be two well-separated cells. Let q be a multipole expansion in s, i.e. an application from the interpolation grid $\mathcal{L}$ in s to $\mathbb{C}$. The **M2L** operator between t and s, with an interpolation grid $\mathcal{K}$ on t, can be written as

$$\left(\mathbf{M2L}[t, s] \cdot q\right)(\mathbf{x}) := \sum_{\mathbf{y} \in \mathcal{L}} G(\mathbf{x}, \mathbf{y}) q(\mathbf{y}) \tag{4.7}$$

$\forall\, \mathbf{x} \in \mathcal{K}$. Notice that this **M2L** operator definition strongly recalls the direct computation in Eq. 2.1. Indeed, the problem of evaluating a **M2L** operator actually is a small N-body problem, where the particles are the interpolation nodes of the source and target cell interpolation grids.

4.1.3 Compressions

The interpolation process can be seen as a first coarse low-rank approximation of the approximated kernel, the ranks being equal to the number of points in the interpolation grids. In practice, the matrix representations of the **M2L** operators are still too large for highly efficient computations with reasonable targeted accuracy. However, the low-rank property of the non-oscillatory asymptotically smooth kernels restricted to well-separated cells can be exploited to add another level of low-rank

approximation. In this section, we present techniques that are used for the compression of the **M2L** matrices. It is important to keep in mind that these compressions are costly, but they are performed only during a precomputation step and supposed to be known and stored when the FMM application step runs.

4.1.3.1 Singular Value Decomposition

As presented in Sect. 2.2.5.2, the SVD algorithm provides an explicit way for constructing low-rank approximations with controlled errors. This algorithm gives the optimal low-rank approximation when truncating accordingly to a given targeted error. Because the matrix to be compressed has to be entirely known to run the SVD factorization algorithm, if G can be evaluated on any pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{K} \times \mathcal{L}$ in $\mathcal{O}(1)$ operations for interpolation grids $\mathcal{K}$ and $\mathcal{L}$ on well-separated cells t and s respectively, the needed memory and operation count to generate the input of the SVD algorithm are both $\mathcal{O}(\#\mathcal{K}\#\mathcal{L})$. In addition, the SVD algorithm complexity is $\mathcal{O}(\#\mathcal{K}^3 + \#\mathcal{L}^2\#\mathcal{K} + \#\mathcal{L}\#\mathcal{K}^2)$. Thus, the SVD-based compressions are costly. A low-rank approximation approach based on SVDs has been considered in [103].

4.1.3.2 Adaptive Cross Approximation

A well known efficient alternative to SVDs in the context of matrices generated by kernel evaluations (such as the **M2L** matrices) is the Adaptive Cross Approximation (ACA) [34–36]. The purpose of ACA is to approximate the SVD of matrices obtained by discretizing integral operators on well-separated sets. An interesting variant of ACA, called *partially pivoted* ACA does not need to know the entire approximated matrix. One only needs to be able to generate the entries of the compressed matrices *on-the-fly*. This is easy to do in the context of *bbfmm* (the kernel G and the interpolation grids are known). The complexity of the partially pivoted ACA algorithm applied to a **M2L** matrix associated to interpolation grids $\mathcal{K}$ and $\mathcal{L}$ is $\mathcal{O}(r^2(\#\mathcal{K} + \#\mathcal{L}))$ (see [163]), where r is the rank of the computed low-rank approximation. The approximations obtained by (partially pivoted) ACA are not optimal: for a fixed targeted accuracy the SVD algorithm finds the minimal numerical rank, which is not the case of ACA. However, because of its complexity compared to the SVD algorithm, this is a classical choice in the implementations of fast hierarchical methods. In the context of *bbfmm*, ACA has been used for instance in [174].

4.1.3.3 Block-compression

A method compressing at once all the **M2L** matrices at a tree level is proposed in [103] using SVDs and adapted in [174] with ACA applications. These compressions allow to approximate each **M2L** matrix $K_{t,s}$ between well-separated cells t and s at a fixed level l of the 2^d-tree as a product of the form

$$K_{t,s} \approx J_l C_{t,s} F_l \tag{4.8}$$

where $\exists r \in \mathbb{N}^*$ such that $J_l \in \mathbb{C}^{L^d \times r}$, $C_{t,s} \in \mathbb{C}^{r \times r}$, $F_l \in \mathbb{C}^{r \times L^d}$ and J_l, F_l are the same for all such $K_{t,s}$, L being the interpolation order in a single axis and supposed to be equal in each direction. This compression is more costly than an individual compression of each possible **M2L** matrix of the interaction lists up to permutations (see [173]), but this reduces the size of multipole and local expansions for the **M2L** application step (from L^d to k) as well as the overall cost of the **M2L** evaluation step, which is known to be the most time consuming of the (low-frequency) FMM far field computation.

4.1.3.4 Recompression

The block-compression of Sect. 4.1.3.3 can be further optimized by adding another level of compression (see [173]). The idea is simple since this only consists in applying a SVD on each of the

$C_{t,s}$'s of Eq. 4.8. If these $C_{t,s}$'s have a low rank, this may result in an additional acceleration in the **M2L** evaluation step. Notice that SVDs are applied to relatively small matrices.

4.1.3.5 Proper Generalized Decomposition

Another kind of compression is proposed in [161], based on the Proper Generalized Decomposition (PGD), aiming at reducing the cost of the SVDs in the precomputation step. The kernel is approximated by a sum of Hadamard products. Authors show better performances using the PGD algorithm than SVDs.

4.1.4 Fast Fourier Transform techniques

In [41, 42, 188, 199], the idea of considering uniformly sampled interpolation nodes (grids referred to as *equispaced grids*) in cells of same radius in 2^d-trees is used to obtain a Toeplitz structure for each **M2L** matrix in *bbfmm*. The same kind of trick is used for instance in [219] applied to another kind of multipole methods. Indeed, if G verifies a translational invariance property

$$G(\mathbf{x},\mathbf{y}) = G(\mathbf{x}-\mathbf{y}) = G(x_1 - y_1, ..., x_d - y_d)$$

for any $\mathbf{x},\mathbf{y} \in \mathbb{R}^d$, then the **M2L** matrices of *bbfmm* are block-Toeplitz. This structure can be exploited to break down the complexity of the matrix-vector product (see [159]). We present in this section the reasons of this optimization. In the following set of definitions, rows and columns of considered matrices are indexed from 0 to $P-1$, $P \in \mathbb{N}$.

Definition 4.1.3. *A Toeplitz matrix is a matrix $T \in \mathbb{C}^{L\times L}$ such that*

$$\exists\, t \;:\; [\![-(L-1), L-1]\!] \to \mathbb{C}$$

with

$$T_{i,j} = t(j-i)$$

$\forall i,j \in [\![0, L-1]\!]$.

Definition 4.1.4. *A circulant matrix is a Toeplitz matrix $C \in \mathbb{C}^{P\times P}$ such that*

$$\exists \begin{cases} \tilde{c} \;:\; [\![0, P-1]\!] \to \mathbb{C} \\ c \;:\; [\![-(P-1), P-1]\!] \to \mathbb{C} \end{cases}$$

verifying $\forall k \in [\![-(P-1), -1]\!]$, $c(k) = \tilde{c}(k+P)$ and

$$C_{i,j} = c(j-i)$$

$\forall i,j \in [\![0, P-1]\!]$.

Definition 4.1.5. *A block-0-Toeplitz (resp. block-0-circulant) matrix is a Toeplitz (resp. circulant) matrix. For any $d \in \mathbb{N}^*$, a block-d-Toeplitz (resp. block-d-circulant) matrix is a block matrix, Toeplitz (resp. circulant) by block and whose blocks are $(d-1)$-block-Toeplitz (resp. $(d-1)$-block-circulant).*

Hence, any (block-)Toeplitz matrix $A \in \mathbb{C}^{L\times L}$ can be embedded in a (block-)circulant one, and by applying a zero-padding, one can express A as the restriction of a larger circulant matrix. In the simple block-0-Toeplitz case (which corresponds to the 1D case), this larger matrix has $P := 2L-1$ rows and columns and may read

$$C = \begin{bmatrix} A & \tilde{A} \\ \hat{A} & \bar{A} \end{bmatrix}$$

where the applications c and $\tilde{c}$ generating the entries of the circulant matrix C (as in Def. 4.1.4) are obtained by periodizing the application t generating the Toeplitz matrix A (as in Def. 4.1.3). This can be written

$$\tilde{c}(i) = \begin{cases} t(i) & \text{if } i \in [\![0, L-1]\!] \\ t(i-2L+1) & \text{if } i \in [\![L, 2L-1]\!] \end{cases}.$$

One may notice that this circulant embedding is not unique. Nevertheless, assuming that we want to compute Av, $v \in \mathbb{C}^L$, and denoting by $\tilde{v} \in \mathbb{C}^P$ the zero-padding of v on the $P-L$ last coordinates, we obtain

$$C\tilde{v} = \begin{bmatrix} A & \tilde{A} \\ \hat{A} & \bar{A} \end{bmatrix} \begin{bmatrix} v \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} Av \\ \hat{A}v \end{bmatrix}$$

whose restriction to the first L coordinates gives the targeted result.

Any circulant matrix $C \in \mathbb{C}^{P\times P}$ can be explicitly diagonalized in a Fourier basis, that is

$$\exists!\ D \in \mathbb{C}^{P\times P},\ \ D_{i,j} = 0\ \ \forall i \neq j \quad | \quad C = \mathbb{F}^* D \mathbb{F}$$

where $\mathbb{F} \in \mathbb{C}^{P\times P}$ and

$$\mathbb{F}_{k,l} = e^{2i\pi \frac{kl}{P}}.$$

The crucial point is that the products by $\mathbb{F}$ and $\mathbb{F}^*$ can be applied with complexity $\mathcal{O}(P \log\ P)$ by means of Fast Fourier Transforms (FFT) and Inverse Fast Fourier Transforms respectively. However, the gain on the precomputation step is only obtained if D can be efficiently computed.

Theorem 4.1.2. *Let $C \in \mathbb{C}^{P\times P}$ be a circulant matrix and D be the diagonal matrix such that $C = \mathbb{F}^* D \mathbb{F}$. We have,*

$$D_{l,l} = C_{0,0} + \sum_{k=1}^{P-1} C_{0,P-k} e^{2i\pi \frac{lk}{P}} \tag{4.9}$$

$\forall l \in [\![0, P-1]\!]$.

The Eq. 4.9 can be interpreted as the Discrete Fourier Transform of the application c generating the entries of C (as in Def. 4.1.4). All the values of c are listed in both the first row and the first column of C. In other words, using matrix notations, we have

$$D = diag(\mathbb{F}C_0) \tag{4.10}$$

where C_0 refers to the first column of C. Because the product by $\mathbb{F}$ can be accelerated using FFT, the assembly of D from C_0 costs $\mathcal{O}(P\ log\ P)$ operations. In practice, because the matrix A we are dealing with corresponds to an array of nodal evaluations of the Green kernel G on (differences between) interpolation nodes, we have to add the cost of P evaluations of G to this cost. Fortunately, since G can be evaluated in $\mathcal{O}(1)$ flops in practice, this cost can be neglected.

This circulant embedding process applied to the multivariate tensorized interpolation on equispaced grids generates block-$(d-1)$-Toeplitz matrices for a problem in $\mathbb{R}^d$ such that, denoting by $\subset$ the embedding into a block-$(d-1)$-circulant matrix:

$$A \subset C = \mathbb{F}_d^* D \mathbb{F}_d$$

where $\mathbb{F}_d := \mathbb{F}^{\otimes d}$, i.e. the tensorization of $\mathbb{F}$ d times (see App. A). An example of a block-1-Toeplitz matrix (i.e. for $d = 2$) and its circulant embedding is provided in Eq. 4.11, where

$a, b, c, d, e, f, g, h, i \in \mathbb{C}$.

$$\begin{bmatrix} \begin{bmatrix} a & b \\ c & a \\ g & h \\ i & g \end{bmatrix} & \begin{bmatrix} d & e \\ f & d \\ a & b \\ c & a \end{bmatrix} \end{bmatrix} \subset \begin{bmatrix} \begin{bmatrix} a & b & c \\ c & a & b \\ b & c & a \end{bmatrix} & \begin{bmatrix} d & e & f \\ f & d & e \\ e & f & d \end{bmatrix} & \begin{bmatrix} g & h & i \\ i & g & h \\ h & i & g \end{bmatrix} \\ \begin{bmatrix} g & h & i \\ i & g & h \\ h & i & g \end{bmatrix} & \begin{bmatrix} a & b & c \\ c & a & b \\ b & c & a \end{bmatrix} & \begin{bmatrix} d & e & f \\ f & d & e \\ e & f & d \end{bmatrix} \\ \begin{bmatrix} d & e & f \\ f & d & e \\ e & f & d \end{bmatrix} & \begin{bmatrix} g & h & i \\ i & g & h \\ h & i & g \end{bmatrix} & \begin{bmatrix} a & b & c \\ c & a & b \\ b & c & a \end{bmatrix} \end{bmatrix} \tag{4.11}$$

Remark 4.1.1. *The entry* $(\mathbf{k}, \mathbf{l})$, $\mathbf{k}, \mathbf{l} \in [\![0, P-1]\!]^d$ *of the multivariate Fourier matrix seen as a block-matrix is given by* $e^{2i\pi \frac{\langle \mathbf{k}, \mathbf{l} \rangle}{P}}$, *using multi-index notations.*

There are many advantages of using these FFT techniques instead of the low-rank approximations:

- The **M2L** matrices are evaluated *exactly*;
- Only L^d kernel evaluations are needed per **M2L** matrix, where L is equal to the one-dimensional interpolation order, compared to $(L^d)^2$ using SVD techniques;
- Due to the cost of FFT, each diagonalization costs $\mathcal{O}(dP^d log(P))$;
- Highly efficient FFT applications can be performed using *state-of-the-art* implementations [107];
- The matrices $\mathbb{F}$ and $\mathbb{F}^*$ are commonly applied to each multipole and local expansion respectively. This means that their application cost can be counted only *once* per expansion, and they can be applied before and after the **M2L** evaluation step, resulting in diagonal **M2L** matrix applications with complexity $\mathcal{O}(P^d) = \mathcal{O}(2^d L^d)$ which is linear with respect to the number of interpolation nodes at a fixed dimension.

The main drawback of the circulant embedding techniques in the *bbfmm* context comes from the induced memory footprint. Due to the zero-padding, the multipole and local expansions are larger than in *bbfmm* (that uses the low-rank methods of Sects. 4.1.3.4 and 4.1.3.3). In [41, 210], a comparison between low-rank approximations and FFT techniques is provided, exhibiting better FFT performances when the interpolation order is greater than 4 in each dimension for the implementation used in [210].

4.2 Directional approaches

The traditional methods for kernel-independent FMMs struggle to fastly approximate highly oscillatory kernels in the high frequency regime. Fortunately, there exists an approach that allows to extend a kernel-indepedent FMM formulation for non-oscillatory kernels to oscillatory ones. We describe in this section on which mathematical bases this extension relies (Sect. 4.2.1) and how a FMM can be derived from these, providing explicit formula in the *bbfmm* context (Sect. 4.2.2).

4.2.1 Directional low-rank property

In [94], the ideas presented in [47] (and detailed in App. C) are adapted to derive a *directional low-rank property* (called directional parabolic separation condition). See also [95, 96]. The general

approach consists in splitting the kernel G into a product of a *smooth* (far from its singularity) kernel K and an oscillatory function:

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

where K is defined by $K(\mathbf{x},\mathbf{y}) := G(\mathbf{x},\mathbf{y})e^{-i\kappa|\mathbf{x}-\mathbf{y}|}$. Let u be a point on the unit sphere $\mathbb{S}^{d-1}$ (which is referred to as a *direction*). We have

$$G(\mathbf{x},\mathbf{y}) = e^{i\kappa\langle\mathbf{x},u\rangle}\underbrace{\left(K(\mathbf{x},\mathbf{y})e^{i\kappa\langle\mathbf{x}-\mathbf{y},\frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|}-u\rangle}\right)}_{=:K_u(\mathbf{x},\mathbf{y})}e^{-i\kappa\langle\mathbf{y},u\rangle} \tag{4.12}$$

where the middle term does not oscillate, provided that $|\frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|}-u| \leq \frac{1}{w\kappa}$, where $|\mathbf{x}-\mathbf{y}|w \leq 2\pi$. One approximation of the kernel K_u is performed for each needed u. In practice, the existence of such a single u can be ensured (see [94]) for two cells s and t with the same radius provided that their distance $dist(s,t)$ is such that

$$dist(s,t) \geq w^2\kappa.$$

A simple computation in [50] illustrates these directional constraints. Indeed, the red part of Eq. 4.12 gives

$$\begin{aligned}\kappa\langle\mathbf{x}-\mathbf{y},\frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|}-u\rangle &= \kappa|\mathbf{x}-\mathbf{y}| - \kappa\langle\mathbf{x}-\mathbf{y},u\rangle \\ &= \kappa|\mathbf{x}-\mathbf{y}|(1-cos\angle(\mathbf{x}-\mathbf{y},u)) \\ &\approx \frac{\kappa}{2}|\mathbf{x}-\mathbf{y}|sin^2\angle(\mathbf{x}-\mathbf{y},u)\end{aligned}$$

where $\angle(\mathbf{a},\mathbf{b})$ denotes the angle between $\mathbf{a},\mathbf{b} \in \mathbb{R}^d$. Thus, if $\angle(\mathbf{x}-\mathbf{y},u)$ tends to 0, $\kappa\langle\mathbf{x}-\mathbf{y},\frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|}-u\rangle$ also does. In this case, the argument of the middle exponential in Eq. 4.12 tends to 0 and the oscillations stop at a certain point. This leads to Thm. 4.2.1, that expresses a *directional low-rank property*.

Theorem 4.2.1. *([94]) Let u be a direction. Let $Y_r := B(0,r)$ be a ball centered at 0 and let X_r be such that*

$$X_r := \{\mathbf{x}\,|\,\angle(\mathbf{x},u) \leq \frac{1}{r\kappa},\ |\mathbf{x}| \geq \kappa r^2\}. \tag{4.13}$$

For any $\epsilon > 0$, there exists a semi-separable approximation of $e^{i\kappa|\mathbf{x}-\mathbf{y}|}/|\mathbf{x}-\mathbf{y}|$ with $T(\epsilon)$ terms, were $T(\epsilon)$ is independent of r, such that

$$\left|e^{i\kappa|\mathbf{x}-\mathbf{y}|}/|\mathbf{x}-\mathbf{y}| - \sum_{n=1}^{T(\epsilon)}\alpha_n(\mathbf{x})\beta_n(\mathbf{y})\right| \leq \epsilon$$

$\forall\mathbf{x} \in X_r$, $\mathbf{y} \in Y_r$, where $\alpha_n : X_r \to \mathbb{C}$, $\beta_n : Y_r \to \mathbb{C}$.

Remark 4.2.1. *A set X_r as in Thm. 4.2.1 is actually a wedge directed by u with angle $\frac{2}{r\kappa}$ and apex located at the origin but deprived of its intersection with the interior of the ball of radius κr^2. This correspdonds to the set in red on Fig. 4.1. On this figure, the blue set corresponds to Y_r.*

A method built on the property expressed in Thm. 4.2.1 is said to be *directional*. As opposed to the Brandt's method (see App. C), for a given pair of well-separated cells, the directions are not computed using interpolation in the directional approach of [38,50,52,94–96,174,204], but they are extracted from a finite direction set different for each cell size.

The general method somehow consists, for a fixed direction, in finding an approximation of the kernel function in a wedge whose angle depends on the wavenumber and on the diameter of the source cell. The number of such approximations grows with the size of the cells (in a similar way than in the Brandt's method, see App. C): doubling the radius of a cell multiplies by a factor 2^{d-1} the number of directions associated to this cell in order to cover the sphere with wedges such as in Thm. 4.2.1.

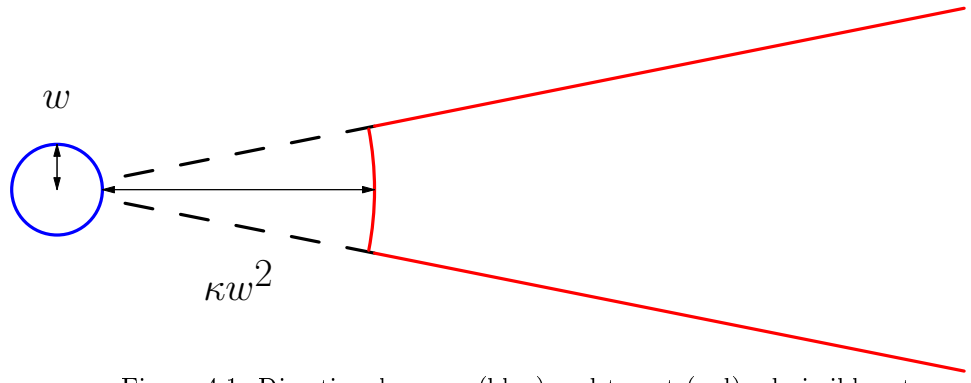


Figure 4.1: Directional source (blue) and target (red) admissible sets.

4.2.2 Directional FMM

The directional low-rank property of Thm. 4.2.1 can be used to derive FMM-like algorithms, as described in [94] in a $kifmm$ framework (see App. B.1 for a high level description of *kifmm*) and extended in [174] to the *bbfmm* case (see Sect. 4.1.2). When combined with a FMM formulation, this directional property allows the definition of wideband methods that become *directional* in the high frequency regime. We define the high frequency regime in a 2^d-tree as the levels in which the radii of the cells in terms of wavelength are greater or equal than 1. Since the kernel-independent methods converge in the low-frequency regime, they can be applied in the deepest levels of the 2^d-trees. Once the high-frequency levels are reached during the upward pass, the multipole expansions become directional, in the sense that they depend on a direction and are valid for approximations on a wedge related to this direction only. The transformation of a directional multipole expansion into a local one (actually with the same direction) keeps this dependence.

4.2.2.1 Nested directions

The directional FMM relies in practice on links between directional approximations. To be more precise, considering 2^d-trees, when the (directional) multipole expansions of a cell are assembled from its son's ones, we want a given multipole expansion corresponding to a given direction to only depend on the multipole expansions of its sons corresponding to a single given (and easy to determine) direction. Of course, the same discrete set of directions is chosen for all the cells sharing the same radius. Let us denote by $\mathcal{D}_l$ this set of directions at level l of a 2^d-tree (which is unique because all the cells of a 2^d-tree at a fixed level have the same radius denoted by w_l, following the description in Sect. 2.2.1.2). To provide these links, $\mathcal{D}_l$ has to verify two properties:

$$\begin{cases} \forall t, s \text{ cells of radius } w_l, \mathit{Level}(t) = \mathit{Level}(s), \exists u \in \mathcal{D}_l \text{ such that } |\frac{ctr(t)-ctr(s)}{|ctr(t)-ctr(s)|} - u| \leq \frac{1}{\kappa w_l}, \\ \forall u \in \mathcal{D}_l, \exists u' \in \mathcal{D}_{l+1} \text{ such that } |u - u'| \leq \frac{1}{\kappa w_l}. \end{cases} \tag{4.14}$$

The first condition ensures that a sufficient amount of directions is considered to cover the sphere using wedges with angles dictated by the associated tree level. The second one links the direction sets of the different levels in a hierarchical way. Indeed, denoting $pred(u)$ a fixed choice of $u' \in \mathcal{D}_{l+1}$ verfying the second point of Eq. 4.14 for any $u \in \mathcal{D}_l$, we can introduce a *direction tree* $\mathbb{D}$ whose nodes are directions and respecting two assumptions on *pred* and $\mathbb{D}$. Let $u \in \mathbb{D}$ be a node (a

direction) in $\mathbb{D}$. We have

$$\begin{cases} Level(u) = l \Rightarrow u \in \mathcal{D}_l \\ |pred(u) - u| \leq \frac{1}{\kappa w_l} \end{cases}.$$

A similar construction is presented for instance in [50].

Remark 4.2.2. *The direction tree $\mathbb{D}$ can be extended to a structure that also takes into account the low-frequency regime, using "negative levels" corresponding to the low-frequency regime, where only a single node exists with value 0 (which is technically not a direction). Because $e^{i\kappa\langle \mathbf{x},0\rangle} = 1$, the modifications involving the associated planewaves (complex exponentials with a dot product argument scaled by the wavenumber and directed by vector in $\mathbb{R}^d$) do not apply.*

4.2.2.2 Complexity and directional MAC

Two new elements have to be considered in the complexity analysis of the directional FMM. First, the MAC that can be derived from Thm. 4.2.1 is much more restrictive in the high frequency regime than the low-frequency MACs such as the strict MAC of Sect. 2.2.4.1 or the adaptive one of Sect. 2.2.4.2, due to the square factor in Eq. 4.13 and to the direction (i.e. the wedge) dependency. In addition, the **M2M**/**L2L** operator application needs to take into account the relations between the directional expansions combined with the links between cells in the 2^d-trees. Because of the increasing number of expansions associated to a cell when the tree level decreases (i.e. from the leaves to the root), these **M2M**/**L2L** operator evaluations are much more time consuming in the high-frequency regime than in the low-frequency one. Hence, $\mathcal{O}(N\,logN)$ complexities were exhibited in [94, 174]. An important point is that this complexity does not depend on the particle distribution [174].

In [50], an adapative formulation of the directional MAC is provided, giving

$$\begin{cases} \kappa \left| \frac{ctr(t)-ctr(s)}{|ctr(t)-ctr(s)|} - u \right| \leq \frac{\eta_1}{2\,max\{rad(t),rad(s)\}} \\ 4\,\kappa\,max\{rad(t)^2, rad(s)^2\} \leq \eta_2\,dist(t,s) \\ 2\,max\{rad(t), rad(s)\} \leq \eta_2\,dist(t,s) \end{cases} \tag{4.15}$$

where $\eta_1, \eta_2 > 0$ and t, s are two cells. The increasing number of directions from the leaves to the root of the 2^d-trees is clear in the first point of Eq. 4.15: the right-hand side of the inequality tends to 0 as the radii of t and s increase, meaning that the minimal number of directions u to verify this first point for any t, s increases from the leaves to the root of the 2^d-trees.

4.2.2.3 Generating directions

The problem of generating the directions can be solved by considering a polyhedron with vertices lying on the unit sphere and by recursively splitting the faces of this polyhedron. The projections of the barycenter of each cell of the induced mesh (or the projection of the vertices of this mesh as in Fig. 4.2) on the unit sphere can then be used to generate a set of directions. This idea is used in [50, 94, 174] with the unit cube on which each face is subdivided into four squares recursively in 3D. The generated set of directions is proven to be compatible with a directional approach [94].

4.2.2.4 Interpolation-based FMM with directional approach

In [174], the directional approach of [94] is formulated in the *bbfmm* case, resulting in a reformulation of operators presented in Sect. 4.1.2.2, including the directional dependency. The stability of the interpolation process between directional expansions in this context is studied in [52]. In this section, we summarize these directional operators. The main difference between directional and non-directional methods is that multiple expansions, each depending on a direction, are associated

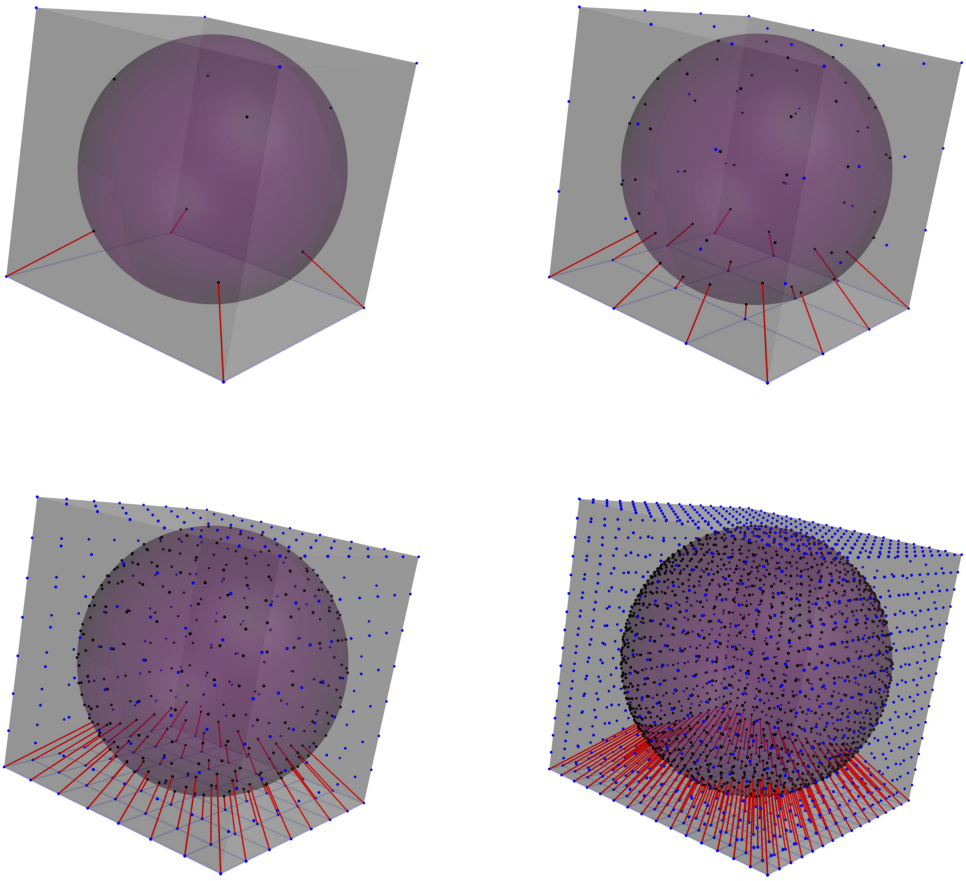


Figure 4.2: Recursive decomposition of the faces of the cube (grid on the lower face) and projection (black) of vertices (blue) of the induced mesh (for the lower grid, the distance between vertices on the cube and the sphere is represented in red).

to each cell. In addition, the *cell-cell* interactions can only be performed between expansions associated to directions that verify the directional MAC in Eq. 4.15 when the high frequency regime is reached. We exhibit in red the elements added in each operator by the directional approach to the *bbfmm* formalism presented in Sect. 4.1.2.2.

Directional P2M/L2P operators. Let s be a leaf. Let $q : Y_{|s} \to \mathbb{C}$ be a vector of charges on the particles of s. The directional **P2M** operator on s with direction $u \in \mathcal{D}_{Level(s)}$ can be written as

$$\left(\mathbf{P2M}_u[s] \cdot q\right)(\mathbf{y}_l) := e^{i\kappa\langle \mathbf{y}_l, u\rangle} \sum_{\mathbf{y} \in Y_{|s}} S_l(\mathbf{y}) e^{-i\kappa\langle \mathbf{y}, u\rangle} q(\mathbf{y})$$

$\forall \mathbf{y}_l \in \mathcal{L}$, where S_l is the Lagrange polynomial associated to $\mathbf{y}_l$ and $\mathcal{L}$ the interpolation grid on s. The definition of the directional **L2P** operator directly follows: for any local expansion defined on the interpolation grid $\mathcal{K}$ on t, i.e. any application $p : \mathcal{K} \to \mathbb{C}$

$$\left(\mathbf{L2P}_u[t] \cdot p\right)(\mathbf{x}) := e^{i\kappa\langle \mathbf{x}, u\rangle} \sum_{\mathbf{x}_k \in \mathcal{K}} T_k(\mathbf{x}) e^{-i\kappa\langle \mathbf{x}_k, u\rangle} p(\mathbf{x}_k)$$

where $\mathbf{x}_k$ is the interpolation node in the grid $\mathcal{K} \subset t$ associated to the Lagrange polynomial T_k, $\mathbf{x} \in t$ is a particle and $u \in \mathcal{D}_{Level(t)}$.

Directional M2M/L2L operators. Let q be a multipole expansion in the cell $s' \in Sons(s)$, i.e. an application from the interpolation grid $\mathcal{L}' := \{\mathbf{y}'_1, ..., \mathbf{y}'_{L^d}\} \subset s'$ to $\mathbb{C}$. The directional **M2M** operator with direction u from the cell s' to s can expressed as

$$\left(\mathbf{M2M}_u[s, s'] \cdot q\right)(\mathbf{y}_l) := e^{i\kappa\langle \mathbf{y}_l, u\rangle} \sum_{\mathbf{y}'_r \in \mathcal{L}} S_l(\mathbf{y}'_r) e^{-i\kappa\langle \mathbf{y}'_r, u\rangle} q(\mathbf{y}'_r)$$

$\forall \mathbf{y}_l \in \mathcal{L}$, the interpolation grid in s, where S_l refers to the Lagrange polynomial associated to $\mathbf{y}_l$ and $\forall\, u \in \mathcal{D}_{Level(s)}$. In practice, such an operator is applied on a directional multipole expansion associated to $pred(u)$ in the direction tree $\mathbb{D}$. Once again, the directional **L2L** operator definition derives from the **M2M** one. Hence, denoting by $\mathcal{K}' := \{\mathbf{x}'_1, ..., \mathbf{x}'_{L^d}\}$ the interpolation grid on the cell $t' \in Sons(t)$, by $\mathcal{K} := \{\mathbf{x}_1, ..., \mathbf{x}_{L^d}\}$ the interpolation grids in t and by p a (directional) local expansion in t (with direction $u' \in Sons(u)$ in $\mathbb{D}$), i.e. an application from $\mathcal{K}$ to $\mathbb{C}$, the directional **L2L** operator between t and t' can be written as

$$\left(\mathbf{L2L}_u[t', t] \cdot p\right)(\mathbf{x}'_r) := e^{i\kappa\langle \mathbf{x}_r, u\rangle} \sum_{\mathbf{x}_k \in \mathcal{K}} T_k(\mathbf{x}'_r) e^{-i\kappa\langle \mathbf{x}'_k, u\rangle} p(\mathbf{x}_k)$$

where T_k is the Lagrange polynomial associated to $\mathbf{x}_k$ and $\mathbf{x}'_u \in \mathcal{K}'$, for any $u \in \mathcal{D}_{Level(t')}$.

Directional M2L operator. The way we wrote the **M2L** in Sect. 4.1.2.2 can be used directly in the directional framework. The handling of the directional aspect in the **M2L** operator evaluation is of algorithmic concerns: the result of a (directional) **M2L** operator application to a directional multipole expansion with direction u has to be added to a directional local expansion with direction u. However, this does not affect the definition of this operator.

Part II

Symmetries in Fast Multipole Method

Chapter 5

High level approach of hierarchical methods

Contents

The FMM formulations are usually described as algorithms instead of matrix factorizations. In the same time, the different FMM formulations benefit from optimizations (e.g. symmetry based optimizations [166, 173]) that can be generalized to other formulations. However, the lack of a rigorous framework allowing to describe the matrix structure makes these generalizations sometimes non trivial.

Early work on the matrix representation of the FMM has been proposed in [196] based on explicit expansions (i.e. on a particular FMM formulation). The theoretical framework we describe in Sect. 5.1 uses a quite different approach and applies to various hierarchical methods.

We want to handle all the common aspects of the FMMs using a general theoretical framework describing the algorithm, independently of the explicit realization of a FMM (i.e. independently of the formulation, kernel or expansion types), which is the purpose of Sect. 5.2. We also present in Sect. 5.3 a set of general results on the structure of the FMM according to the space representations of the formulation.

5.1 Matrix form

In this section, we describe the mathematical tools we use to derive the matrix form of the hierarchical methods, i.e. the factorization of the matrices obtained using a method such as a FMM, a hierarchical matrix or a treecode. In Sect. 5.1.1 we recall the general aspects of the N-body problems from a linear algebra viewpoint. In Sect. 5.1.2 we reformulate the underlying idea of the well-separateness and in Sect. 5.1.3 we present a general definition of the tree structures used in the hierarchical methods. In Sect. 5.1.4, we introduce the key tools in our approach, that are the concepts of *freely generated vector spaces* and *rank distributions*. Finally, in Sect. 5.1.5 we reformulate the naive N-body problem using the objects presented in Sect. 5.1.4.

5.1.1 Spaces

We start with the matrix-vector product expression of a general N-body problem (see Sect. 2.2.5.1), recalling first this expression using a slightly more abstract framework than in Sect. 2.2.5.1.

Let $\mathcal{X}$ be any finite set. We denote by $\mathbb{C}[\mathcal{X}]$ the set of applications from $\mathcal{X}$ to $\mathbb{C}$. Any element of $\mathbb{C}[\mathcal{X}]$ can be represented by an element of $\mathbb{C}^{\#\mathcal{X}}$ (actually $\mathbb{C}[\mathcal{X}] \equiv \mathbb{C}^{\#\mathcal{X}}$ where $\equiv$ denotes that there exists an isomorphism between the right term and the left one), that is it can be represented by a vector. We can express the N-body problem of Sect. 2.1.1 (i.e. Eq. 2.1) using this notation. X and Y still refer to the target and source point clouds respectively. Then, let $p \in \mathbb{C}[X]$ and $q \in \mathbb{C}[Y]$. We obtain that Eq. 2.1 is a linear mapping from $\mathbb{C}[Y]$ to $\mathbb{C}[X]$, so this mapping can be represented by a matrix $A \in \mathbb{C}^{\#X \times \#Y}$. We have:

$$p = Aq. \tag{5.1}$$

This actually corresponds to Eq. 2.17 in Sect. 2.2.5.1. Each entry of A is here an evaluation of G on a point of X and one of Y. The difference between the two expressions lies in the spaces. Indeed, vector and matrices were considered in Eq. 2.17 but we look at linear mapping between applications in Eq. 5.1.

Denoting $q_{\mathbf{y}} := q(\mathbf{y})$ and $p_{\mathbf{x}} := p(\mathbf{x})$, the source or target particles can be somehow used as index sets for the entries of the vector representations of p and q. This allows to bypass the ordering of the particles. Since A is linear, it can be represented in $\mathbb{C}^{\#X \times \#Y}$ with the matrix of Eq. 2.17 and we naturally extend this index notation to A by

$$A_{\mathbf{x},\mathbf{y}} := G(\mathbf{x}, \mathbf{y})$$

for any $\mathbf{x} \in X$ and $\mathbf{y} \in Y$, G being the kernel of the N-body problem.

We shall discuss in the rest of this chapter how these notations can be handled using a well suited mathematical framework and how to extend them to deal with restrictions of this matrix A according to particular space decompositions.

5.1.2 Approximability

At the most abstract level, hierarchical methods are based on the observation that, if a group of target particles and a group of source particles are *well-separated* (or *admissible*), then the corresponding part of Eq. 2.1 (or equivalently Eq. 5.1) can be efficiently approximated (see Sect. 2.2.2).

Instead of tackling this problem by directly considering the particles, the idea is to consider subsets of $\mathbb{R}^d$ in which some of these particles lie and to compare them (see Sect. 2.2.3.1 and 2.2.4.1). Because we restrict ourselves to subsets of $\mathbb{R}^d$ in which the particles can *possibly* lie, we can consider the smallest balls containing the target and source point clouds. The application that decides if two subsets are admissible is called here an **approximability condition** (which has quite the same role as the MACs of Sect. 2.2.4.1 and 2.2.4.2). The purpose of the Def. 5.1.1 (see below) is to formalize what exactly such an approximability condition will represent, even if we can understand its role without introducing a detailed formalism.

Indeed, when two subsets are admissible, the kernel G is supposed to be *semi-separable* (with an error ϵ) on the restrictions of the target and source point clouds according to these admissible subsets. This semi-separability means that, on admissible subsets $t, s \subset \mathbb{R}^d$, $G : t \times s \to \mathbb{C}$ can be decomposed (up to a certain error) as a sum of weighted products of applications of $t \to \mathbb{C}$ and $s \to \mathbb{C}$ independently. These functions, with no additional assumption, depend on both t and s and their number depends on the semi-separability of G on (t, s). The approximability condition is chosen in practice so that only a small number of functions has to be chosen in order to retrieve a "good" approximation of the kernel on (t, s). This is the reason why we consider that these numbers are arguments of the approximability condition, so as the tolerance $\epsilon > 0$. Formally, this reads:

Definition 5.1.1. *Let $\bar{X}, \bar{Y} \in \mathbb{R}^d$, $R_X, R_Y \in \mathbb{R}^{*+}$ such that $X \subset B(\bar{X}, R_X)$ and $Y \subset B(\bar{Y}, R_Y)$. For a fixed $\epsilon > 0$, $r_0, r_1 \in \mathbb{N}^*$, an **approximability condition** associated to the kernel G is an application $\mathcal{A} : B(\bar{X}, R_X) \times B(\bar{Y}, R_Y) \to \{0, 1\}$ such that $\forall b = (t, s) \subseteq B(\bar{X}, R_X) \times B(\bar{Y}, R_Y)$, $\mathcal{A}(t, s) = 1$ if and only if*

$$\exists \begin{cases} \Psi_b & : t \to \mathbb{C}^{r_0}, \\ \Phi_b & : s \to \mathbb{C}^{r_1}, \\ \Delta_b^{(t,s)} & \in \mathbb{C}^{r_0 \times r_1} \end{cases}$$

with

$$\forall \mathbf{x} \in t,\ \forall \mathbf{y} \in s,\ |G(\mathbf{x}, \mathbf{y}) - \Psi_b^*(\mathbf{x}) \Delta_b \Phi_b(\mathbf{y})| \leq \epsilon. \tag{5.2}$$

*If t and s are such that $\mathcal{A}(t, s) = 1$, they are said to be **well-separated** or **admissible**.*

Remark 5.1.1. *The term Δ_b can actually be removed from Def. 5.1.1 and r_0 can be chosen to be equal to r_1. However, this general approximation form with a three terms factorization will be used to express in the same way various methods in the following of this chapter. For the sake of clarity, we keep this general form all along this chapter.*

All the elements of this decomposition a priori depend on t and s, justifying to tag each of them using these subsets.

The restriction of the N-body problem to $b = (t, s)$ such that $\mathcal{A}(t, s) = 1$, i.e. the restriction of Eq. 2.1 to (t, s) or equivalently the restriction of Eq. 5.1 to the block[1] $(t \cap X) \times (s \cap Y)$, allows the approximate factorization of Eq. 5.2 to be exploited in order to reduce the application cost of the considered subproblem. Indeed, we have

$$\begin{aligned} \sum_{\mathbf{y} \in s \cap Y} G(\mathbf{x}, \mathbf{y}) q(\mathbf{y}) &\approx \sum_{\mathbf{y} \in s \cap Y} \Psi_b^*(\mathbf{x}) \Delta_b \Phi_b(\mathbf{y}) q(\mathbf{y}) \\ &= \Psi_b^*(\mathbf{x}) \Delta_b \sum_{\mathbf{y} \in s \cap Y} \Phi_b(\mathbf{y}) q(\mathbf{y}) \end{aligned} \tag{5.3}$$

for any $\mathbf{x} \in t$ (and especially for $\mathbf{x} \in t \cap X$) and where the notation $\approx$ is used to emphasize that the $\mathcal{O}(\epsilon)$ error in Eq. 5.2 is neglected in Eq. 5.3.

The term $\sum_{\mathbf{y} \in s \cap Y} \Phi_b(\mathbf{y}) q(\mathbf{y})$ in the last line of Eq. 5.3 is a matrix-vector product and costs $\mathcal{O}(\#(s \cap Y) r_1)$ flops to be evaluated. The product of the result with Δ_b costs $\mathcal{O}(r_0 r_1)$ flops to be evaluated and knowing the result, the remaining product by $\Psi_b(\mathbf{x})$ costs $\mathcal{O}(r_0 \#(t \cap X))$ to be performed for any $\mathbf{x} \in t \cap X$. This three-steps scheme for this part of the whole computation has a total complexity of $\mathcal{O}(\#(s \cap Y) r_1 + r_0 \#(t \cap X) + r_0 r_1) = \mathcal{O}(2KM + K^2)$ if we assume that $r_0 = r_1 = K$ and $\#(s \cap Y) = \#(t \cap X) = M$ as opposed to the $\mathcal{O}(M^2)$ for a naive evaluation. If K is "small enough", this results in a lower operation count.

Thinking in terms of the matrix form in Eq. 5.1, this means that admissible blocks of A corresponding to admissible restrictions of the target and source point clouds can be approximated by factorizations. If $K << M$ (see Rem. 5.1.2 for the precise condition), these factorizations are low-rank factorizations (a schematic representation of this situation is depicted in Fig. 5.1).

[1] Since t, s may not be finite sets in Def. 5.1.1, the intersection with the point clouds is needed to evaluate only the target and source particles.

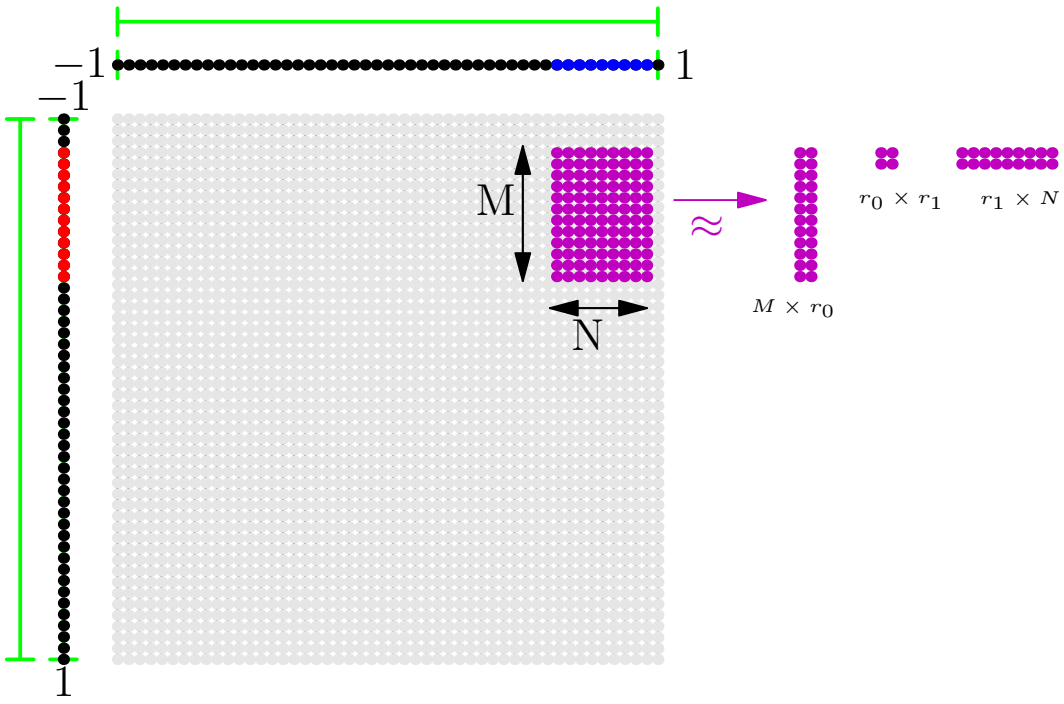


Figure 5.1: Representation of the low rank approximation (purple, right of arrow) of a block (purple, left of arrow) of an admissible couple of target set of particles (red) and source set of particles (blue) for an uniform distribution of particles along $[-1, 1]$. Convex hull of these point clouds are printed in green. The corresponding approximability condition is defined by $\mathcal{A}(t, s) = 1 \Leftrightarrow rad(t) + rad(s) \leq |ctr(t) - ctr(s)|$ (i.e. the adaptive MAC of Sect. 2.10).

Remark 5.1.2. *To obtain a lower operation count than the direct problem (without approximation), we need to have at least*

$$\frac{2KM + K^2}{M^2} < 1 \Leftrightarrow 2K + \frac{K^2}{M} < M$$
$$\Leftrightarrow 2K(1 + \frac{K}{2M}) < M.$$

Since $K \leq M$ (the rank of a square matrix cannot exceed its number of rows/columns), a sufficient criterion is given by

$$K < \frac{M}{3}.$$

5.1.3 Hierarchical decomposition

From an abstract viewpoint, the idea of hierarchical methods is to find (and possibly maximize the number or sizes of) patterns $(t \cap X, s \cap Y) \in B(\bar{X}, R_X) \times B(\bar{Y}, R_Y)$ (i.e. sub-blocks of A) such that $\mathcal{A}(t, s) = 1$, provided that r_0 and r_1 are sufficiently small to allow a cost reduction. To do so, structured decompositions of $B(\bar{X}, R_X)$ and $B(\bar{Y}, R_Y)$ separately are introduced as **hierarchical decompositions**.

Definition 5.1.2. *A **hierarchical decomposition** of a convex subset $E \subset \mathbb{R}^d$ refers to a finite sequence $(\mathcal{U}_k)_k$ of partitions of E where $\mathcal{U}_k$ is a collection of subsets of E, i.e. $\underset{u \in \mathcal{U}_k}{\cup} u = E$ for any k, verifying:*

$$\begin{cases} \mathcal{U}_0 = E, \\ u \in \mathcal{U}_{k+1} \Rightarrow \exists!\ v \in \mathcal{U}_k \mid u \subseteq v. \end{cases}$$

To provide an intuitive idea on how this hierarchical decomposition $(\mathcal{U}_k)_k$ of a set $E \subset \mathbb{R}^d$ is made, one can point out that if $u \in \mathcal{U}_k$, then its own hierarchical decomposition is contained in $\mathcal{U}_p$, $p > k$. So a partition of u is contained in $\mathcal{U}_p$. In addition, because $\mathcal{U}_p$ is a partition of E, this partition of u is unique for a fixed p and stems from the partition of u in $\mathcal{U}_{p-1}$ (using the second point of Def. 5.1.2). Examples of hierarchical decompositions are provided in Exp. 5.1.1. These examples will be reused all along this chapter.

Example 5.1.1. *Let us consider a non-uniform hierarchical decomposition $(\mathcal{U}_k)_k$, $k = 0, ..., 3$ of the segment $[-1, 1]$:*

$$\begin{aligned}
&k = 0 \qquad \mathcal{U}_0 = \{[-1, 1]\}\\
&k = 1 \qquad \mathcal{U}_1 = \{[-1, 0), [0, 1]\}\\
&k = 2 \qquad \mathcal{U}_2 = \{[-1, 0), [0, 0.1), [0.1, 1]\}\\
&k = 3 \qquad \mathcal{U}_3 = \{[-1, 0), [0, 0.01), [0.01, 0.1), [0.1, 0.2), [0.2, 0.3), [0.3, 1]\}.
\end{aligned}$$

Another hierarchical decomposition $(\mathcal{V}_k)_k$, $k = 0, ..., 3$ of the same segment is given by the following sequence:

$$\begin{aligned}
&k = 0 \qquad \mathcal{V}_0 = \{[-1, 1]\}\\
&k = 1 \qquad \mathcal{V}_1 = \{[-1, 0), [0, 1]\}\\
&k = 2 \qquad \mathcal{V}_2 = \{[-1, -0.5), [-0.5, 0), [0, 0.5), [0.5, 1]\}\\
&k = 3 \qquad \mathcal{V}_3 = \{[-1, -0.75), [-0.75, -0.5), [-0.5, -0.25), [-0.25, 0), [0, 0.25), [0.25, 0.5), [0.5, 0.75), [0.75, 1]\}.
\end{aligned}$$

A crucial point here is that any hierarchical decomposition $(\mathcal{U}_k)_k$ can be represented as a tree whose nodes are elements of a hierarchical decomposition (i.e. subsets of $\mathbb{R}^d$).

Definition 5.1.3. *The **tree representation** T of a hierarchical decomposition $(\mathcal{U}_k)_k$ of a set E is the tree such that any node $n \in T$ verifies:*

$$\begin{cases}
n \in \mathcal{U}_{Level(n)},\\
n \subseteq Father(n),\\
n = \underset{n' \in Sons(n)}{\cup} n',\\
[Level(n) = Level(n') \text{ and } n \cap n' \neq \emptyset] \Rightarrow n = n'
\end{cases}$$

where Father associates to a node of a tree the node of its parent in this tree, Sons associates to a node the set of its sons in the tree and Level associates to a node its level in the tree.

Remark 5.1.3. *This definition implies that $Root(T) \in \mathcal{U}_0 = \{E\}$, so $Root(T) = E$, where Root associates to a tree its root node. In addition, T has a number of level equal to the number of terms in the sequence $(\mathcal{U}_k)_k$.*

Remark 5.1.4. *There exists a bijection between the nodes at a fixed level k of the tree representation of a hierarchical decomposition and the elements of the associated partition of E, i.e. with $\mathcal{U}_k$.*

Example 5.1.2. *We present in Fig. 5.2 the tree representations of the hierarchical decompositions of $[-1, 1]$ in the example 5.1.1. These two decompositions are composed of 4 partitions of $[-1, 1]$, so the associated trees have 4 levels.*

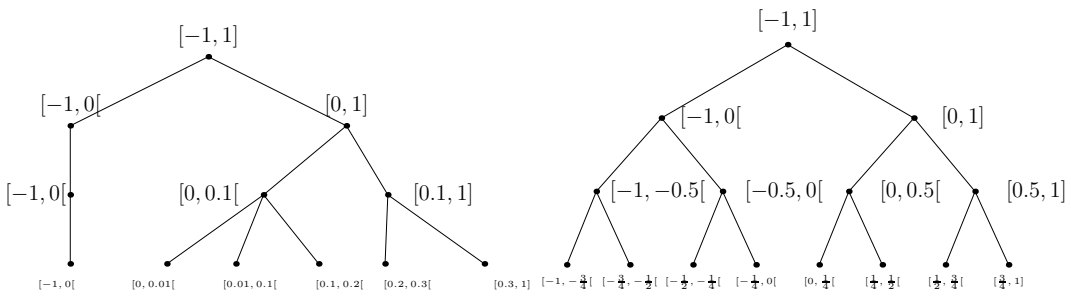


Figure 5.2: Tree representation of $(\mathcal{U}_k)_k$ (left) and $(\mathcal{V}_k)_k$ (right)

The tree representation of $(\mathcal{V}_k)_k$ coincides with a 2^1-tree (binary tree). Compared to Sect. 2.2.1.2, the problem of locating the particles lying in the boundaries of cells (for instance a particle located at 0) is addressed thanks to the fourth condition in Def. 5.1.3.

Relying on such simple structured multilevel representations of subsets of $\mathbb{R}^d$, we can hierarchically compare corresponding nodes according to an approximability condition. The next definition links our point clouds with these tree representations.

Definition 5.1.4. *A **tree representation** of a point cloud $\mathcal{X}$ refers to a tree representation $\mathscr{C}$ of a hierarchical decomposition of the smallest ball encompassing $\mathcal{X}$.*

*A node of $\mathscr{C}$ is also called a **cell**. We use the notation $c \in \mathscr{C}$ if c is a node (a cell) of $\mathscr{C}$.*

These node are different from the particles lying in them. When considering a node, the information about the exact localization of the particles is (a priori) lost. The restriction of a point cloud $\mathcal{X}$ with a tree representation $\mathscr{C}$ to the particles lying in c, $c \in \mathscr{C}$, is thus given by $c \cap \mathcal{X}$. Because we imposed the hierarchical decompositions to be formed by partitions, a single particle lies in exactly one leaf of the tree representation (and in only one cell at each level of this tree representation).

Remark 5.1.5. *Using different norms, the smallest ball encapsulating a point cloud may have different forms. In the case of 2^d-trees, the L^∞ norm is chosen.*

We consider a tree representation of X (our target point cloud, see Sect. 5.1.1), that will be denoted by $\mathscr{T}$ and one of Y (the source point cloud) which will be denoted by $\mathscr{S}$. Let $s \in \mathscr{S}$ and $t \in \mathscr{T}$. Due to Def. 5.1.3 $\forall c, \tilde{c} \in \mathscr{X}$, $\mathscr{X} = \mathscr{T}, \mathscr{S}$, $Level(c) < Level(\tilde{c})$:

$$c \cap \tilde{c} \neq \emptyset \Rightarrow \exists K \in \mathbb{N}^*, c'_0, ..., c'_K \in \mathscr{X} \mid \begin{cases} c'_{k+1} \in Sons(c'_k), & \forall k = 1, ..., K-1 \\ c'_0 = c \\ \tilde{c} \in Sons(c'_K) \end{cases}.$$

Such a $\tilde{c}$ is a descendant of c in $\mathscr{X}$ (see Sect. 2.2.1). Denoting by $Desc(c)$ the set of descendants of a cell $c \in \mathscr{X}$, we obtain that for any approximability condition $\mathcal{A}$ and any $t \in \mathscr{T}$, $s \in \mathscr{S}$,

$$\mathcal{A}(t, s) = 1 \Rightarrow \mathcal{A}(t', s') = 1, \quad \forall t' \in Desc(t), \ s' \in Desc(s)$$

because an approximation as in Def. 5.2 and applicable in $t \times s$ (so for any $\mathbf{x} \in t \cap X$ and any $\mathbf{y} \in s \cap Y$) can be applied in any restriction of this set.

5.1.4 Freely generated vector spaces

Our goal is to present a matrix framework that allows calculus in a context where tree representations $\mathscr{T}$ and $\mathscr{S}$ come into play as index sets, so as the particles in each cell of those trees. Such index sets are actually unordered, which makes the presentation slightly more abstract. This discussion is based on the concepts of **freely generated vector spaces** and **rank distribution**.

Definition 5.1.5. *Let $\mathcal{X}$ be a finite set. A **rank distribution** r over $\mathcal{X}$ is a map $r : \mathcal{X} \to \mathbb{N}^*$. If $\forall x \in \mathcal{X}$, $r(x) = 1$, then r is said to be trivial and we denote $\mathbf{1}_{\mathcal{X}} := r$ the trivial rank distribution on $\mathcal{X}$.*

Definition 5.1.6. *Let $\mathcal{X}$ be a finite set and r be a rank distribution over $\mathcal{X}$. The **freely generated vector space** $\mathbb{C}[r]$ is the vector space*

$$\mathbb{C}[r] := \prod_{x \in \mathcal{X}} \mathbb{C}^{r(x)}$$

where $\prod$ denotes the cartesian product.

Remark 5.1.6. *For the trivial rank distribution on a finite set $\mathcal{X}$, we have that $\mathbb{C}[\mathbf{1}_{\mathcal{X}}]$ coincides with $\mathbb{C}[\mathcal{X}] \equiv \mathbb{C}^{\#\mathcal{X}}$.*

In our context, $\mathcal{X}$ will either refer to a point cloud or the nodes of a tree. Let r be a rank distribution over $\mathcal{X}$. Any factor $\mathbb{C}^{r(x)}$ can be regarded as a subspace of $\mathbb{C}[r]$, so that we can write:

$$\mathbb{C}[r] \equiv \bigoplus_{x \in \mathcal{X}} \mathbb{C}^{r(x)}.$$

Definition 5.1.7. *Let r be a rank distribution over $\mathcal{X}$ and $u \in \mathbb{C}[r]$. The restriction of u to $x \in \mathcal{X}$ is denoted by $u|_x \in \mathbb{C}^{r(x)}$.*

Next, let $\mathcal{X}$ and $\mathcal{Y}$ be two finite sets, for any pair of rank distributions $r_{\mathcal{X}} : \mathcal{X} \to \mathbb{N}^*$ and $r_{\mathcal{Y}} : \mathcal{Y} \to \mathbb{N}^*$, a linear map $\eta : \mathbb{C}[r_{\mathcal{Y}}] \to \mathbb{C}[r_{\mathcal{X}}]$ can be defined by prescribing a family of linear maps $\eta_{x,y} : \mathbb{C}^{r_{\mathcal{Y}}(y)} \to \mathbb{C}^{r_{\mathcal{X}}(x)}$ for $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and we have

$$\eta(u)|_x = \sum_{y \in \mathcal{Y}} \eta_{x,y}(u|_y). \tag{5.4}$$

This expression is an abstraction of a matrix defined blockwise. We may now easily define the restriction of A (see Eq. 5.1) to a pair of cells. We drop the "|" notation since this does not induce any ambiguity.

As presented in Sect. 5.1.1, A can be interpreted as a linear mapping from $\mathbb{C}[X]$ to $\mathbb{C}[Y]$, that is a linear mapping between $\mathbb{C}[\mathbf{1}_X]$ to $\mathbb{C}[\mathbf{1}_Y]$ thanks to Rem. 5.1.6. Hence, the mappings $A_{\mathbf{x},\mathbf{y}}$, $Aut(\mathbb{C}^1) \equiv \mathbb{C}$ (with Aut denoting the set of automorphisms) as defined in Eq. 5.4 are scalars, for any $\mathbf{x} \in X$ and $\mathbf{y} \in Y$. They can be easily deduced from the N-body problem formulation in Eq. 2.1: the element $A_{\mathbf{x},\mathbf{y}}$ refers to the kernel evaluation $G(\mathbf{x}, \mathbf{y})$. In addition, we have the isomorphisms

$$\begin{cases} \mathbb{C}[\mathbf{1}_{t \cap X}] \equiv \mathbb{C}[t \cap X] \equiv \mathbb{C}[r_X(t)] \\ \mathbb{C}[\mathbf{1}_{s \cap Y}] \equiv \mathbb{C}[s \cap Y] \equiv \mathbb{C}[r_Y(s)] \end{cases} \tag{5.5}$$

where $t \in \mathscr{T}$ and a source cell $s \in \mathscr{S}$ and r_X, r_Y are defined such that

$$r_X(t) := \#(t \cap X), \quad r_Y(s) := \#(s \cap Y). \tag{5.6}$$

As a consequence of Eq. 5.5 we can consider the block of A corresponding to the N-body problem restriction to the particles in t and s by means of the rank distributions in Eq. 5.6 (isomorphism between the second and third terms in the two lines of Eq. 5.5). In particular, this implies that $\mathscr{T}$ and $\mathscr{S}$ can be used as index sets to refer to the corresponding blocks $A_{t,s}$ of A. In addition, the sets $t \cap X$ and $s \cap Y$ can be used as index sets to refer to the elements of such a block $A_{t,s}$ (isomorphism between the first and second terms in the two lines of Eq. 5.5). Explicit formulas are provided in Def. 5.1.8.

Definition 5.1.8. *The **restriction** of A to a target particle $\mathbf{x} \in X$ and a source particle $\mathbf{y} \in Y$ is the element $A_{\mathbf{x},\mathbf{y}}$.*

*The **restriction** of A to a target cell $t \in \mathscr{T}$ and a source cell $s \in \mathscr{S}$ is the linear map $A_{t,s} : \mathbb{C}[s \cap Y] \to \mathbb{C}[t \cap X]$ such that $\forall u \in \mathbb{C}[s \cap Y] \equiv \mathbb{C}^{\#(s \cap Y)}$*

$$(A_{t,s}u)_{\mathbf{x}} = \sum_{\mathbf{y} \in s \cap Y} A_{\mathbf{x},\mathbf{y}} u_{\mathbf{y}}$$

$\forall \mathbf{x} \in t \cap X$.

Remark 5.1.7. *For this definition, we indeed used the trivial rank distributions on $t \cap X$ and $s \cap Y$. So $A_{t,s}$ can be represented as a matrix in $\mathbb{C}^{\#(t \cap X) \times \#(s \cap Y)}$ due to Rem. 5.1.6.*

Thanks to Def. 5.2, any block $A_{t,s}$ such that $t \in \mathscr{T}$, $s \in \mathscr{S}$ are admissible with regard to a given approximability condition can be factorized, up to a given error. In addition, due to the hierarchical space decomposition structure, the blocks induced by sons of both t and s can be regarded as restrictions of left and right factors of this factorization.

We can illustrate the hierarchical structure of the tree representations on such restrictions through the relation:

$$(A_{t,s}u)_{|_{t'}} = \sum_{s' \in Sons(s)} A_{t',s'} u_{s'}$$

for any $u \in \mathbb{C}[s \cap Y]$, $t' \in Sons(t)$.

This formula is the bridge between restrictions of A and the hierarchical decomposition behind the tree structures we introduced in Sect. 2.2.1.2. It is now possible to manipulate A from the viewpoint of space decompositions. In the following, we will navigate in A not only from the viewpoint of the hierarchical decompositions, but also according to a given approximability condition, which will widely use this notion of freely generated vector spaces and non trivial rank distributions.

To be more precise, given two rank distributions $r_{\mathscr{T}}$ and $r_{\mathscr{S}}$ on $\mathscr{T}$ and $\mathscr{S}$ respectively, we will consider operators S from $\mathbb{C}[r_{\mathscr{S}}] \to \mathbb{C}[r_{\mathscr{T}}]$ and their **adjoints** S^* with regard to the scalar product:

$$\langle\langle a, b \rangle\rangle_{r_{\mathscr{H}}} := \sum_{c \in \mathscr{H}} \langle a_c, b_c \rangle$$

for any $a, b \in \mathbb{C}[r_{\mathscr{H}}]$ and using $\langle \cdot \rangle$ to denote the standard scalar product in the subspace $\mathbb{C}^{r_{\mathscr{H}}(c)}$ of $\mathbb{C}[r_{\mathscr{H}}]$ for any fixed $c \in \mathscr{H}$, $\mathscr{H} = \mathscr{T}, \mathscr{S}$. To emphasize how this notion is close to the standard matrix adjoint notion, one may think in terms of matrix in all subspaces of $\mathbb{C}[r_{\mathscr{H}}]$, $\mathscr{H} = \mathscr{T}, \mathscr{S}$ using the definition of this scalar product. S being a mapping between $\mathbb{C}[r_{\mathscr{S}}]$ to $\mathbb{C}[r_{\mathscr{T}}]$, we have:

$$\begin{aligned} \langle\langle a, Sb \rangle\rangle_{r_{\mathscr{T}}} &= \sum_{c \in \mathscr{T}} \langle a_c, (Sb)_c \rangle \\ &= \sum_{c \in \mathscr{S}} \langle (S^*a)_c, b_c \rangle \\ &=: \langle\langle S^*a, b \rangle\rangle_{r_{\mathscr{S}}}. \end{aligned}$$

This defines S^* as the adjoint of S.

5.1.5 Near and far fields

Since *a priori* not the entire A will be efficiently approximated by local factorizations, we want to separate A into two parts: the first one will be composed of untouched blocks (i.e. non-admissible blocks which will not be approximated) and approximated ones (i.e. a well chosen set of admissible blocks). This untouched part is called the **near field** of A and can be characterized using an approximability condition.

Definition 5.1.9. *Let $\mathcal{A}$ be an approximability condition. Let t be a leaf cell of $\mathscr{T}$. The near field $\mathcal{N}(t)$ of t is the set $\{s \in \mathscr{S} \mid \mathcal{A}(t,s) = 0$ **and** s is a leaf in $\mathscr{S}\}$.*

Using the previously introduced notions, we can define the near field part of A in Eq. 5.1 as the application $A_{\mathcal{N}}$, where:

$$(A_{\mathcal{N}})_{\mathbf{x},\mathbf{y}} := \begin{cases} A_{\mathbf{x},\mathbf{y}} \text{ if } \exists t \in \mathscr{T}, s \in \mathscr{S}, \mathbf{x} \in t, \mathbf{y} \in s, \mathcal{A}(t,s) = 0 \text{ and } t,s \text{ are both leaves,} \\ 0 \text{ otherwise.} \end{cases}$$

This can be expressed more compactly, introducing the relevant rank distribution.

Definition 5.1.10. *The **natural rank distribution** of a tree representation $\mathscr{H}$ of a point cloud $\mathcal{X}$ is the rank distribution $N_{\mathscr{H}}$ such that*

$$N_{\mathscr{H}}(c) := \#(c \cap \mathcal{X})$$

for any $c \in \mathscr{H}$.

In other words, $N_{\mathscr{H}}$ counts for any cell $c \in \mathscr{H}$ the number of particles of $\mathcal{X}$ lying in c. We then obtain:

$$A_{\mathcal{N}} : \mathbb{C}[N_{\mathscr{S}}] \to \mathbb{C}[N_{\mathscr{T}}]$$

defined such that

$$(A_{\mathcal{N}})_{t,s} = A_{t,s}\delta_{leaf}(t)\delta_{leaf}(s)(1 - \mathcal{A}(t,s)) \tag{5.7}$$

where δ_{leaf} is a boolean application with tree node arguments, which is equal to 1 if and only if its argument is a leaf. Indeed, the right term in Eq. 5.7 vanishes on each admissible pair (t,s) with regard to $\mathcal{A}$ and is equal to the restriction of A to t and s if (t,s) is a non-admissible pair of leaf cells.

$A_{\mathcal{N}}$ is composed of direct interactions between particles. Such interactions are depicted in Fig. 5.3 where we considered two equal target and source point clouds, generated by a uniform distribution over $[-1,1]$ (restricted in the source particles to the 4 first of them). This interaction graph has a dashed line between a target and a source particle if and only if this interaction is computed. This is a dense graph.

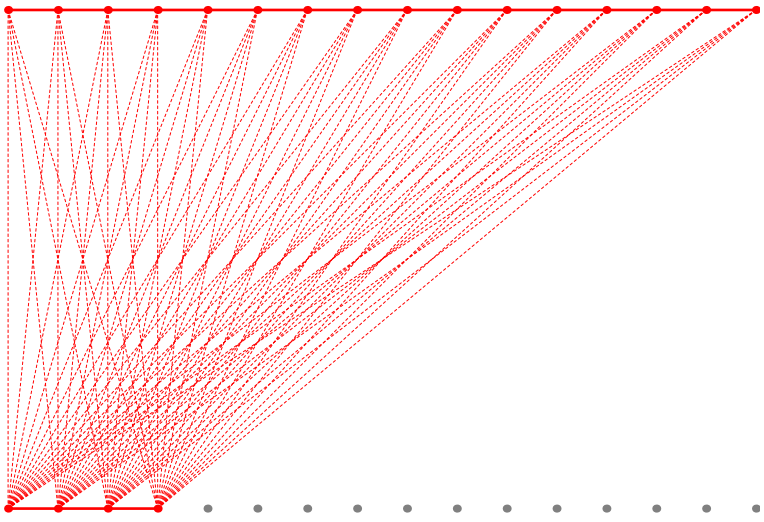


Figure 5.3: Interaction graph for direct computation between a target point cloud (top points) and a source one (bottom points). Only the interactions involving the first four particles of the source point cloud are depicted.

Considering the tree representation viewpoint, each non-zero block of $A_{\mathcal{N}}$ corresponds to a source cell s and a target cell t such that such a dense interaction graph is drawn by the involved computations (locally). Then, denoting the far field part of A by $A_{\mathcal{F}} := A - A_{\mathcal{N}}$, hierarchical methods are based on the following assumptions:

- $A_{\mathcal{N}}$ is a block sparse operator which can be applied in linear time;
- Blocks of $A_{\mathcal{F}}$ can be efficiently approximated by low-rank expressions.

To be more precise, the interaction graph between admissible cells will be modified by the low rank structure of admissible blocks and their shape will depend on the method. A series of examples is given in Sect. 5.2, clarifying in different manners the effective structure of $A_{\mathcal{F}}$ for the hierarchical methods we described in Chap. 2 (i.e. the treecodes, the FMMs and the hierarchical matrices).

An interesting result can already be provided, linking our definition of $A_{\mathcal{F}}$ to a well known sparse format.

Definition 5.1.11. *Let $\mathcal{A}$ be an approxiability condition. Let t be a cell in $\mathscr{T}$. The* ***interaction list*** *$\Lambda_{\mathcal{A}}(t)$ of t is the set $\{s \in \mathscr{S} \mid \mathcal{A}(t,s) = 1$ **and** $\mathcal{A}(Father(t), Father(s)) = 0\}$.*

As a block operator $A_{\mathcal{F}} : \mathbb{C}[N_{\mathscr{S}}] \to \mathbb{C}[N_{\mathscr{T}}]$, $A_{\mathcal{F}}$ can be seen as a Compressed Sparse Row (see [186] Sect. 3.4) operator providing $\Lambda_{\mathcal{A}}(t)$ for any $t \in \mathscr{T}$.

Remark 5.1.8. *The set of admissible pairs $(t,s) \in \mathscr{T} \times \mathscr{S}$ overlaps in $\mathbb{R}^d \times \mathbb{R}^d$. Without the condition on the non-admissibility of the fathers in Def. 5.1.11, some interactions would be evaluated more than one time (up to the number of levels in the tree representations).*

To conclude this paragraph, we reformulate the approximated factorization in Eq. 5.2 using all these new notations.

Lemma 5.1.1. *Let $\mathcal{A}$ be an approximability condition. The following holds:*

$$s \in \Lambda_{\mathcal{A}}(t) \;\Rightarrow\; \exists \begin{cases} \Phi_b : \mathbb{C}[Y \cap s] \to \mathbb{C}^{r_1} \\ \Psi_b : \mathbb{C}[X \cap t] \to \mathbb{C}^{r_0} \\ \Delta_b \in \mathbb{C}^{r_0 \times r_1} \end{cases} \quad \text{such that } A_{t,s} = \Psi_b^* \Delta_b \Phi_b + \mathcal{O}(\epsilon) \tag{5.8}$$

for any $t \in \mathscr{T}$ and using $b := (t,s)$.

5.2 Hierarchical formats

In this section, we prescribe the rank distributions to use in order to recover the hierarchical methods described in Chap. 2. We will illustrate all the considered method through the scope of the same toy example, that is the tree representation of the hierarchical decomposition $(\mathcal{V}_k)_k$ introduced in Exp. 5.1.1. This tree representation can be depicted as in Fig. 5.4. We then suppose that each cell contains exactly 2 particles, generating a point cloud $\mathcal{P}$. We want to solve the N-body problem between $\mathcal{P}$ and itself with a given kernel G. Taking the strict MAC (see Sect. 2.2.4.1) as approximability condition, two cells at the same tree level are admissible if and only if their ancestors are not and they are separated by at least a cell. In Fig. 5.5, we depicted the interaction lists of the cells encompassing the four first particles and the ones of their ancestors. When the N-body problem is solved exactly, the computations needed to obtain the solution for these four particles is the dense graph of Fig. 5.3. We will see how the hierarchical methods affect this graph.

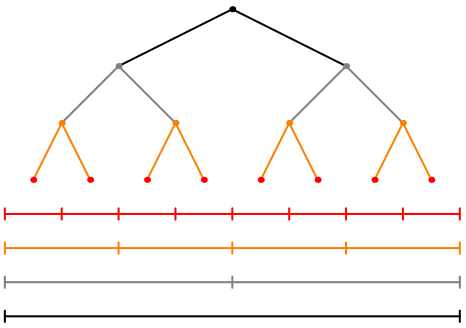


Figure 5.4: Tree representation (binary tree) of the hierarchical decomposition of $[-1,1]$ given by $(\mathcal{V}_k)_k$ of Exp. 5.1.1. The colors are used to illustrate the decomposition at each level (i.e. one color per level).

5.2.1 $\mathcal{H}$-matrix format

The simplest and more general hierarchical compression format for the considered matrices A is the $\mathcal{H}$-matrix format. It can be handled by the framework we presented, using appropriate rank distributions. Let $N_{\mathscr{T}\times\mathscr{S},Y} : \mathscr{T}\times\mathscr{S} \to \mathbb{N}^*$ and $N_{\mathscr{T}\times\mathscr{S},X} : \mathscr{T}\times\mathscr{S} \to \mathbb{N}^*$ be two rank distributions defined as

$$N_{\mathscr{T}\times\mathscr{S},Y}((t,s)) = \begin{cases} r_1 & \text{if } s \in \Lambda_{\mathcal{A}}(t) \\ 0 & \text{otherwise} \end{cases}$$

$$N_{\mathscr{T}\times\mathscr{S},X}((t,s)) = \begin{cases} r_0 & \text{if } s \in \Lambda_{\mathcal{A}}(t) \\ 0 & \text{otherwise} \end{cases}$$

where r_0 and r_1 are the ranks in Eq. 5.8.

Remark 5.2.1. *Since r_0, r_1 are defined for a particular admissible block $(t,s) \in \mathscr{T}\times\mathscr{S}$, they do not have to be the same for all pairs of target/source cells.*

The notations the factorization in Eq. 5.8 are also used in the following formulas. Let V, W be two mappings such that

$$\begin{cases} V : \mathbb{C}[X] \to \mathbb{C}[N_{\mathscr{T}\times\mathscr{S},X}] \\ W : \mathbb{C}[Y] \to \mathbb{C}[N_{\mathscr{T}\times\mathscr{S},Y}] \end{cases}$$

defined by

$$(Wq)_b := \begin{cases} \sum_{\mathbf{y}\in s\cap Y} \Phi_b(\mathbf{y}) q(\mathbf{y}) & \text{if } s \in \Lambda_{\mathcal{A}}(t) \\ 0 & \text{otherwise} \end{cases}$$

$$(V\tilde{q})_b := \begin{cases} \sum_{\mathbf{x}\in t\cap X} \Psi_b(\mathbf{x}) \tilde{q}(\mathbf{x}) & \text{if } s \in \Lambda_{\mathcal{A}}(t) \\ 0 & \text{otherwise} \end{cases}.$$

or equivalently by

$$W_{b,\mathbf{y}} := \begin{cases} \Phi_b(\mathbf{y}) & \text{if } s \in \Lambda_{\mathcal{A}}(t) \\ 0 & \text{otherwise} \end{cases}$$

$$V_{b,\mathbf{x}} := \begin{cases} \Psi_b(\mathbf{x}) & \text{if } s \in \Lambda_{\mathcal{A}}(t) \\ 0 & \text{otherwise} \end{cases}.$$

for any $b = (t,s) \in \mathscr{T}\times\mathscr{S}$, $q \in \mathbb{C}[Y]$, $\tilde{q} \in \mathbb{C}[X]$. Then, we define the mapping S from $\mathbb{C}[N_{\mathscr{T}\times\mathscr{S},Y}]$ to $\mathbb{C}[N_{\mathscr{T}\times\mathscr{S},X}]$ by

$$S_{b_0,b_1} := \begin{cases} \Delta_b & \text{if } b := (t,s) = b_0 = b_1 \text{ and } s \in \Lambda_{\mathcal{A}}(t) \\ 0 & \text{otherwise} \end{cases}$$

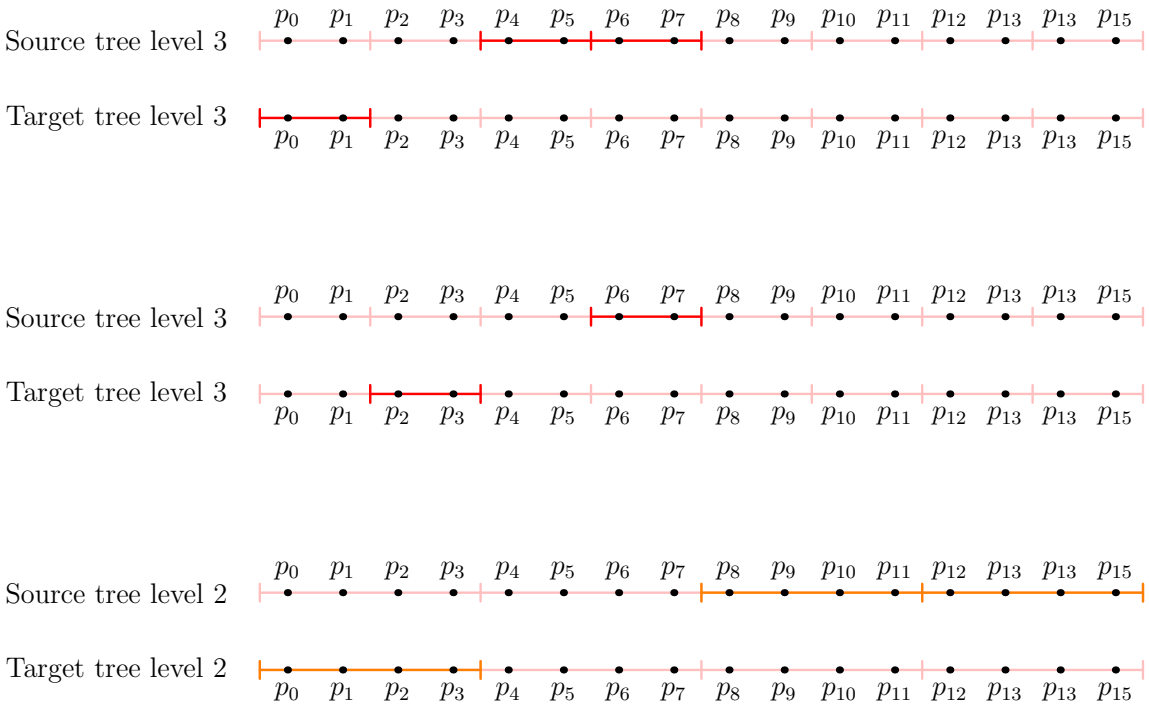


Figure 5.5: Interaction lists for the cells containing the first four target particles of the representation in Fig. 5.4 with two particles per leaf cell using the strict MAC of Sect. 2.2.4.1. There is no well-separated pairs of cells at the top two levels (i.e. levels 0 and 1) so only the two other levels are depicted. The non pale cells of the source level corresponds to the cells in the interaction list of the non pale one in the corresponding target level for each of the three illustrated cells.

for any $b_0, b_1 \in \mathscr{T} \times \mathscr{S}$. As a consequence of Lem. 5.1.1, we obtain

$$\begin{aligned} A &= A_{\mathcal{N}} + A_{\mathcal{F}} \\ &\approx \boxed{A_{\mathcal{N}} + V^* \cdot S \cdot W.} \end{aligned} \tag{5.9}$$

We used the notation $\approx$ to handle the terms $\mathcal{O}(\epsilon)$ appearing in Prop. 5.1.1. This notation will be reused in this chapter. The factorization of the far field of A in Eq. 5.9 leads to a sparse linear operator (i.e. S) but lying in a different space than A. Indeed, the index sets of S are the product of nodes of $\mathscr{T}$ and $\mathscr{S}$, not the particles. The cardinal of $\mathscr{T} \times \mathscr{S}$ can be larger than the dimensions of A (seen as a matrix) but the sparsity of S strongly compensates this difference for common choices of approximability conditions (see the different MACs in Chap. 2). The space mappings in the factorization of $A_{\mathcal{F}}$ in Eq. 5.9 can be listed as follows

$$\mathbb{C}[Y] \xrightarrow{W} \mathbb{C}[N_{\mathscr{T}\times\mathscr{S},Y}] \xrightarrow{S} \mathbb{C}[N_{\mathscr{T}\times\mathscr{S},X}] \xrightarrow{V^*} \mathbb{C}[X].$$

Remark 5.2.2. *The number of non-zero blocks in S is exactly equal to the number of cell-cell interactions performed by the multiplication by the representation of A in the $\mathcal{H}$-matrix format.*

The index sets used for the spaces $\mathbb{C}[N_{\mathscr{T}\times\mathscr{S},Y}], \mathbb{C}[N_{\mathscr{T}\times\mathscr{S},X}]$ being the product $\mathscr{T} \times \mathscr{S}$, the particles do not directly appear in the term S of Eq. 5.9. Since the low rank approximations used in the hierarchical matrices are usually obtained through SVD (see Sect. 2.2.5.2) or ACA (see Sect. 4.1.3.2) applications, for a given pair of admissible cell (t, s) the ranks r_0 and r_1 are often equal in practice. In this case, because the ACA factorization results in two terms and the SVD one also does by multiplying one of the left or right matrices by the singular values, the blocks Δ_b of Eq.

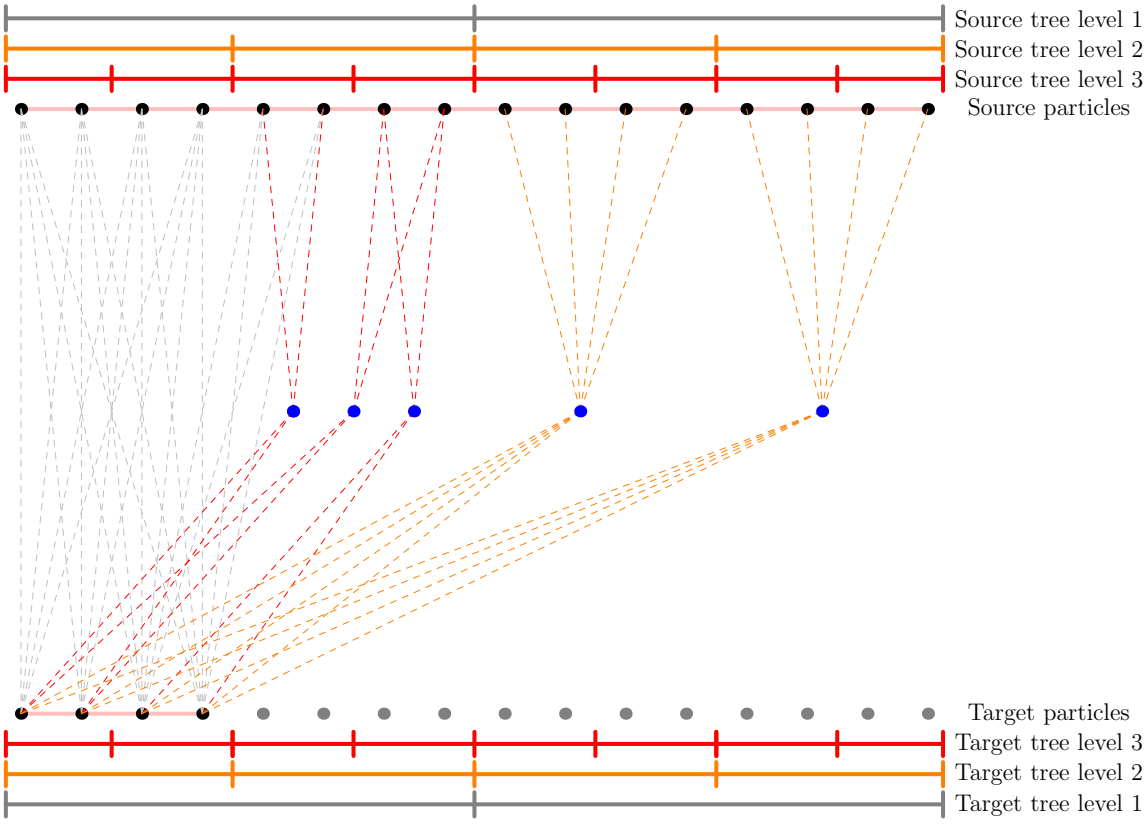


Figure 5.6: Interaction graph for $\mathcal{H}$-matrix with admissibility condition of Fig. 5.1 and the binary tree representation of $(\mathcal{V})_k$ in Fig. 5.2 and illustrated in Fig. 5.4. The same colors than in Fig. 5.4 are used for each level of the tree representation. Near field interactions (gray dashed), interactions at level 3 (red dashed), interactions at level 2 (orange dashed). The far field *cell-cell* interactions are represented using blue dots.

5.8 actually are represented by identity matrices. Each of these identity matrices corresponds to a *cell-cell* interaction in the far field. Hence, the term S can be avoided in Eq. 5.9, resulting in the shorter factorization:

$$\boxed{A \approx A_{\mathcal{N}} + V^* \cdot W.} \tag{5.10}$$

This reduction can be understood differently by simply modifying the expression of W into $\tilde{W}$ defined by

$$\left(\tilde{W}q\right)_b := \begin{cases} \Delta_b \displaystyle\sum_{\mathbf{y}\in s\cap Y} \Phi_b(\mathbf{y})q(\mathbf{y}) & \textit{if } s \in \Lambda_{\mathcal{A}}(t) \\ 0 & \textit{otherwise} \end{cases}.$$

Regarding the interaction graph and symbolizing by nodes the non-zero elements of S, the new shape can be drawn as in Fig. 5.6. These new nodes of course represent an amount of operation depending on the numerical rank of the involved blocks of A, but supposed to be $\mathcal{O}(1)$.

5.2.2 Treecodes

The general $\mathcal{H}$-matrix factorization of Sect. 5.2.1 is based on low rank approximations on each admissible pairs $(t, s) \in \mathscr{T} \times \mathscr{S}$ in which all the terms depend both on the target cell t and the source one s. This generated mappings in large spaces (i.e. $\mathbb{C}[N_{\mathscr{T}\times\mathscr{S},Y}]$ and $\mathbb{C}[N_{\mathscr{T}\times\mathscr{S},X}]$). To the

contrary, the treecode method is built on the assumption that only the left and middle factors in the factorization in Eq. 5.8 depends on both t and s but the right-hand one (i.e. $\Phi_b = \Phi_s$) only depends on s (see Sect. 2.2.3).

As opposed to the hierarchical matrix format, treecodes perform *particle-cell* interactions instead of *cell-cell* interactions. All rank distributions thus have to be adapted in order to represent treecodes in terms of matrix factorization using our framework. To keep our definition of the approximability condition, we consider that a particle is a cell centered in its position with a radius equal to zero. Because of the *particle-cell* interactions, we only have a tree representation $\mathscr{S}$ of the hierarchical decomposition of the source space.

Let $R_{\mathscr{S}}$ be a rank distribution over $\mathscr{S}$ such that for a well chosen approximability condition (such as the MAC of Eq. 2.2.3.1), we have, for a target particle $\mathbf{x} \in X$

$$s \in \Lambda_{\mathcal{A}}(\mathbf{x}) \;\Rightarrow\; \exists \begin{cases} \Phi_s : \mathbb{C}[Y \cap s] \to \mathbb{C}^{R_{\mathscr{S}}(s)} \\ \Psi_s(\mathbf{x}) \in \mathbb{C}^{R_{\mathscr{S}}(s)} \end{cases} \quad \textit{such that } A_{\mathbf{x},s} = \Psi_s(\mathbf{x})^* \Phi_s + \mathcal{O}(\epsilon). \tag{5.11}$$

We thus want to define linear mappings V, W such that the far field $A_{\mathcal{F}}$ can be approximated by the factorization $A_{\mathcal{F}} \approx V^* \cdot W$. In terms of vector spaces, this may be written

$$\mathbb{C}[Y] \xrightarrow{W} \mathbb{C}[R_{\mathscr{S}}] \xrightarrow{V^*} \mathbb{C}[X]. \tag{5.12}$$

The operators V and W can be defined by

$$W_{s,\mathbf{y}} := \begin{cases} \Phi_s(\mathbf{y}) & \textit{if } \mathbf{y} \in s \\ 0 & \textit{otherwise} \end{cases} \tag{5.13}$$

$$V_{s,\mathbf{x}} := \begin{cases} \Psi_s(\mathbf{x}) & \textit{if } \mathbf{x} \in t \\ 0 & \textit{otherwise} \end{cases}$$

using the notations of Eq. 5.11. We obtain the same form as in Eq. 5.10 but with different definitions of V and W based on different rank distributions according to the specific approximation used for treecodes. However, this factorization is still not fully satisfactory since it hides the relations between tree nodes in the upward pass (see Sect. 2.2.3.2). Indeed, we would like to factorize W in such a way that this factorization handles the hierarchical structure inherited from the tree representation of the underlying hierarchical space decomposition. This can be done by introducing *tree restrictions*.

Definition 5.2.1. *Let $\mathscr{H}$ be the tree representation of a hierarchical decomposition. The tree restriction $\mathscr{H}_l$ refers to the set $\mathscr{H}_l := \{h \in \mathscr{H} \mid Level(h) \geq l\}$ for any $l \in [\![0, Depth(\mathscr{H})]\!]$.*

The tree restriction $\mathscr{H}_l$ is composed of all cells of $\mathscr{H}$ that have a level greater or equal to l.

Remark 5.2.3. *There exists a naive inclusion chain on tree restrictions:*

$$\mathscr{H}_{Depth(\mathscr{H})} \subset \mathscr{H}_{Depth(\mathscr{H})-1} \subset \ldots \subset \mathscr{H}_0 = \mathscr{H}.$$

We are going to exploit the *nestedness property* of W which is properly defined in Def. 5.2.2. This consists in a general abstract way of defining the relations between matrices associated to cells and the blocks of W according to the links of the tree representation.

Definition 5.2.2. *Let $\epsilon > 0$ be as in Eq. 5.11. Let $\mathcal{X}$ be a point cloud, $\mathscr{H}$ be the tree representation of a hierarchical decomposition of the smallest ball encompassing $\mathcal{X}$ and $r_{\mathscr{H}}$ be a rank distribution over $\mathscr{H}$. A mapping $B : \mathbb{C}[\mathcal{X}] \to \mathbb{C}[r_{\mathscr{H}}]$ is said to verify the nestedness property if and only if there exist rank distributions $r_{\mathscr{H}_l}$, mappings $B_l : \mathbb{C}[r_{\mathscr{H}_l}] \to \mathbb{C}[r_{\mathscr{H}_{l-1}}]$ for any $l \in [\![1, Depth(\mathscr{H})-1]\!]$, $B_\dagger : \mathbb{C}[\mathcal{X}] \to \mathbb{C}[r_{\mathscr{H}_{Depth(\mathscr{H})}}]$ such that*

$$\exists \begin{cases} E_h \in \mathbb{C}^{r_{\mathscr{H}_{l-1}}(Father(h)) \times r_{\mathscr{H}_l}(h)} & \textit{for any } h \in \mathscr{H} \textit{ such that } Level(h) \neq Depth(\mathscr{H}) \\ \tilde{E}_h \in \mathbb{C}^{r_{\mathscr{H}_{Depth(\mathscr{H})-1}}(Father(h)) \times \#(\mathcal{X} \cap h)} & \textit{for any } h \in \mathscr{H} \textit{ such that } Level(h) = Depth(\mathscr{H}) \end{cases}$$

verifying

$$B = B_1 \cdot \ldots \cdot B_{Depth(\mathscr{H})-1} \cdot B_\dagger + \mathcal{O}(\epsilon) \tag{5.14}$$

with $r_{\mathscr{H}_0} = r_{\mathscr{H}}$,

$$(B_l)_{h_0,h_1} = \begin{cases} Id & \text{if } h_0 = h_1 \\ E_{h_1} & \text{if } h_1 \in Sons(h_0) \\ 0 & \text{otherwise} \end{cases}$$

and $\forall q \in \mathbb{C}[\mathcal{X}]$, $\forall h$ *with* $Level(h) = Depth(\mathscr{H})$,

$$(B_\dagger \cdot q)_h = \tilde{E}_h \cdot q_{|\mathcal{X} \cap h}.$$

The notation $B_{l,\dagger} := B_l \cdot \ldots \cdot B_{Depth(\mathscr{H})-1} \cdot B_\dagger$ *is used to refer to the last* $Depth(\mathscr{H}) - l + 1$ *terms of Eq. 5.14.*

In this section, the matrices $\tilde{E}_h$ are only defined for leaf cells (i.e. cells h such that $Depth(\mathscr{H}) = Level(h)$ using our definition of a tree representation provided in Def. 5.1.3). Actually, Def. 5.2.2 is quite easy to understand. The nestedness property of W guarantees the existence of linear mappings E_s, for any $s \in \mathscr{S}$, such that

$$\begin{aligned} (W \cdot q)_s &\approx \sum_{s' \in Sons(s)} E_{s'} \cdot \left(W_{Level(s)+1,\dagger} \cdot q\right)_{s'} \\ &\approx \sum_{s' \in Sons(s)} E_{s'} \cdot \sum_{s'' \in Sons(s')} E_{s''} \cdot \left(W_{Level(s')+1,\dagger} \cdot q\right)_{s''} \\ &\approx \ldots \end{aligned}$$

for any $q \in \mathbb{C}[Y]$. This may be written in a more compact form, reodering the terms of the decomposition

$$\begin{aligned} W_{h,s} &\approx \delta_{h=s} \sum_{s' \in Sons(s)} E_{s'} \cdot \left(W_{Level(s)+1,\dagger}\right)_{h,s'} \\ &\approx \delta_{h=s} \sum_{\substack{\tilde{s} \in Desc(s) \\ Level(\tilde{s}) = Depth(\mathscr{S})}} \underbrace{\left(\prod_{u \in Ancestors(\tilde{s})} E_u \right)_{s,\tilde{s}}}_{\substack{\text{Uniquely defined with a fixed} \\ \text{element ordering imposed by} \\ \text{the chain in Rem. 5.2.3}}} \tilde{E}_{\tilde{s}} \end{aligned} \tag{5.15}$$

where $\delta_{h=s}$ is equal to 1 if $h = s$ and to 0 otherwise. Notice that the first line of Eq. 5.15 preserves a factorization that disappears in the second line since the sum is placed out of the product.

Hence, any non-zero block of $W_{l,\dagger}$, indexed with cells of $\mathscr{S}_l$, is accessed using a cell of $\mathscr{S}$ and can be obtained using the sons of this cell. This can be repeated recursively. We easily see that the term B_l in Eq. 5.14 involves larger input and output vector spaces than B_{l+1} since the sequence $(\mathscr{S}_l)_l$ verifies $\mathscr{S}_{l+1} \subset \mathscr{S}_l$. In other terms, the size of B_l increases as l decreases. Fortunately, Def. 5.2.2 implies that these operators are sparse assuming that each cell has $\mathcal{O}(1)$ sons.

The upward pass (see Sect. 2.2.3.2) of the treecode can be represented by assuming that the term W in Eq. 5.13 verifies the nestedness property. Hence, there exist mappings W_l, $W_\dagger$ and ranks distributions $R_{\mathscr{S}_l}$ such that the chain 5.12 becomes

$$\mathbb{C}[Y] \xrightarrow{W_\dagger} \mathbb{C}[R_{\mathscr{S}_{Depth(\mathscr{S})}}] \xrightarrow{W_{Depth(\mathscr{S})-1}} \ldots \xrightarrow{W_2} \mathbb{C}[R_{\mathscr{S}_2}] \xrightarrow{W_1} \mathbb{C}[R_{\mathscr{S}}] \xrightarrow{V^*} \mathbb{C}[X]$$

and the factorization in Eq. 5.10 gives

$$\begin{aligned} A &\approx A_{\mathcal{N}} + V^* \cdot W \\ &= \boxed{A_{\mathcal{N}} + V^* \cdot W_1 \cdot \ldots W_\dagger.} \end{aligned}$$

In Fig. 5.7 is represented the interaction graph corresponding to the treecode approach.

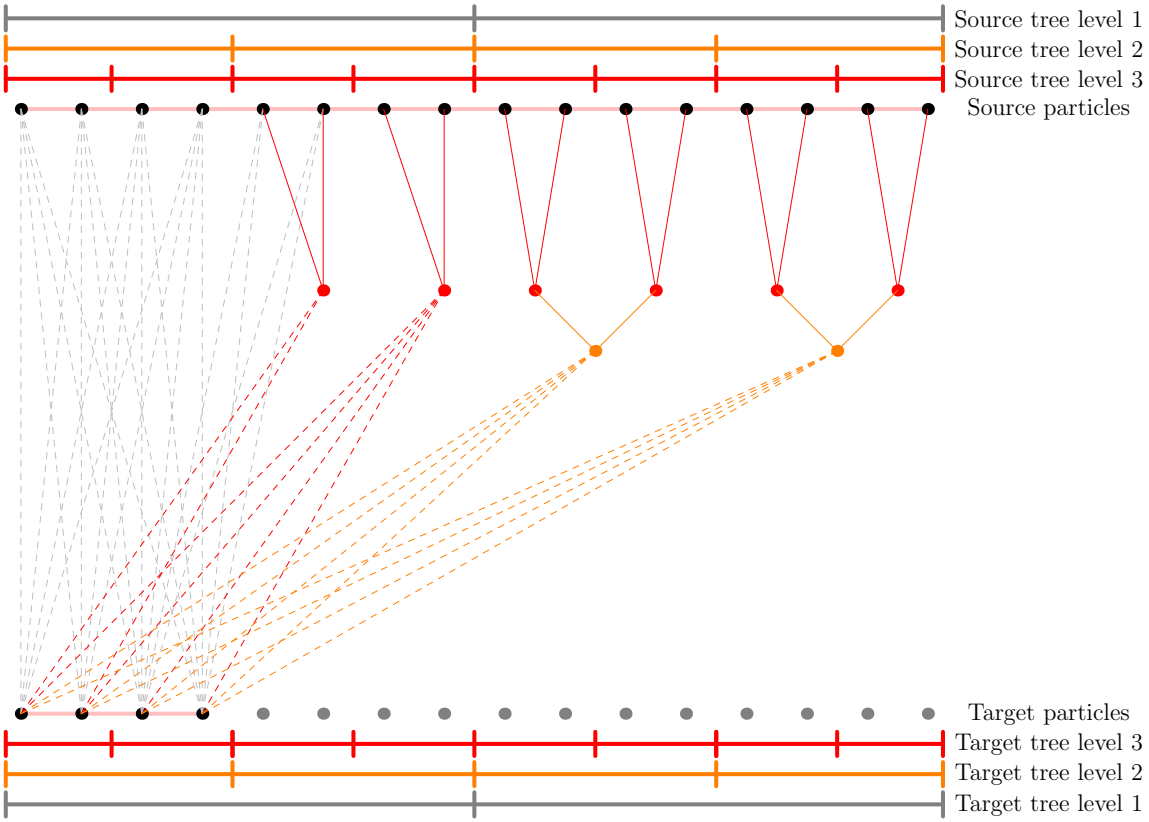


Figure 5.7: Interaction graph for the treecode. The near field part is represented in dashed gray. The hierarchical assembly of far field approximations in the source cells are depicted with continuous lines using the level colors of Fig. 5.4. The *particle-cell* interactions are represented with dashed lines using the level colors of the involved cells. Concerning the MAC, we suppose that the particles of a target cell are well-separated from a source cell if the two cells are well-separated with respect to the strict MAC of Sect. 2.2.4.1 and as illustrated in Fig. 5.5.

5.2.3 Fast Multipole Method

The last method we want to present using our framework is the Fast Multipole Method, based on an additional assumption on the structure of the approximation of the far field compared to the treecode.

5.2.3.1 Matrix formulation

Actually, considering the theoretical framework we develop in this chapter, the FMM can be handled using a combination of the hierarchical matrix and treecode formulations. Indeed, there are both a target and a source tree and we want to represent *cell-cell* interactions in the factorization, which is also the case in the hierarchical matrix format. We thus have a representation as in Eq. 5.9 using rank distributions over the product tree $\mathscr{T} \times \mathscr{S}$

$$A \approx A_{\mathcal{N}} + V^* \cdot S \cdot W. \tag{5.16}$$

However, the kernel approximation is finner in the FMM case. As for the treecode, the source approximations (the $\Phi_b = \Phi_s$'s in Eq. 5.8) only depend on the source cell. The same is also assumed for the target approximation, that only depends on the corresponding target cell. Hence,

we first reformulate the kernel approximation exploited in the FMM. This time, we consider rank distributions $R_{\mathscr{T}}$ on $\mathscr{T}$ and $R_{\mathscr{S}}$ on $\mathscr{S}$. We have

$$s \in \Lambda_{\mathcal{A}}(t) \;\Rightarrow\; \exists \begin{cases} \Phi_s : \mathbb{C}[Y \cap s] \to \mathbb{C}^{R_{\mathscr{S}}(s)} \\ \Psi_t : \mathbb{C}[X \cap t] \to \mathbb{C}^{R_{\mathscr{T}}(t)} \\ \Delta_{t,s} \in \mathbb{C}^{R_{\mathscr{T}}(t) \times R_{\mathscr{S}}(s)} \end{cases} \quad \textit{such that } A_{t,s} = \Psi_t^* \Delta_{t,s} \Phi_s + \mathcal{O}(\epsilon) \tag{5.17}$$

with

$$(A_{t,s})_{\mathbf{x},\mathbf{y}} = \Psi_t^*(\mathbf{x}) \Delta_{t,s} \Phi_s(\mathbf{y}) + \mathcal{O}(\epsilon)$$

for any $\mathbf{x} \in t$, $\mathbf{y} \in s$. Hence, defining

$$V : \mathbb{C}[X] \to \mathbb{C}[R_{\mathscr{T}}], \qquad W : \mathbb{C}[Y] \to \mathbb{C}[R_{\mathscr{S}}], \qquad S : \mathbb{C}[R_{\mathscr{S}}] \to \mathbb{C}[R_{\mathscr{T}}]$$

such that

$$V_{\mathbf{x},t} := \begin{cases} \Psi_t(\mathbf{x}) & \textit{if } \mathbf{x} \in t \\ 0 & \textit{otherwise} \end{cases}, \qquad W_{\mathbf{y},s} := \begin{cases} \Phi_s(\mathbf{y}) & \textit{if } \mathbf{y} \in s \\ 0 & \textit{otherwise} \end{cases}, \qquad S_{t,s} := \begin{cases} \Delta_{t,s}(\mathbf{x}) & \textit{if } s \in \Lambda_{\mathcal{A}}(t) \\ 0 & \textit{otherwise} \end{cases}$$

we recover the approximation in Eq. 5.16 but with different (smaller) spaces than with the hierarchical matrix format. Indeed, instead of the product set $\mathscr{T} \times \mathscr{S}$, the two sets $\mathscr{T}$ and $\mathscr{S}$ are considered separately, giving

$$\mathbb{C}[Y] \xrightarrow{W} \mathbb{C}[R_{\mathscr{S}}] \xrightarrow{S} \mathbb{C}[R_{\mathscr{T}}] \xrightarrow{V^*} \mathbb{C}[X].$$

This intermediate factorization corresponds to a single level FMM as sometimes referred to in the literature (see for instance [70, 74, 174]). This is not sufficient to handle the full structure of the (multilevel) FMM factorization. Indeed, by assuming that V and W both verify the nestedness property according to $\mathscr{T}/R_{\mathscr{T}}$ and $\mathscr{S}/R_{\mathscr{S}}$ respectively, we end up with

$$\begin{aligned} A &\approx A_{\mathcal{N}} + \quad V^* \quad \cdot S \cdot \quad W \\ &\approx \boxed{A_{\mathcal{N}} + V_{\dagger}^* \cdot \ldots \cdot V_1^* \cdot S \cdot W_1 \cdot \ldots W_{\dagger}.} \end{aligned} \tag{5.18}$$

The linear maps involved in Eq. 5.18 have names in the FMM literature [131, 218] that we already listed in Sect. 2.2.4.1. We summarize them in the following list.

1. $(W_{\dagger})_{s,\mathbf{y}}$ refers to the **Particle to Multipole (P2M)** operator on the cell s restricted to $\mathbf{y} \in s \cap Y$,
2. $(W_k)_{s,s'}$ denotes the **Multipole to Multipole (M2M)** operator from $s' \in Sons(s)$ to s (at level k),
3. $S_{t,s}$ is the **Multipole to Local (M2L)** operator between target cell t and source cell s,
4. $(V_k^*)_{t',t}$ denotes the **Local to Local (L2L)** operator from t to $t' \in Sons(t)$ at level k,
5. $\left(V_{\dagger}^*\right)_{\mathbf{x},t}$ refers to the **Local to Particle (L2P)** operator on the cell t restricted to $\mathbf{x} \in t \cap X$,
6. $(A_{\mathcal{N}})_{t,s}$ is the **Particle to Particle (P2P)** operator between cells t and s.

As restrictions of linear maps in $\mathbb{C}$-vector spaces, these operators can all be represented as matrices, and we will use the same terminology for the matrices associated to these operators.

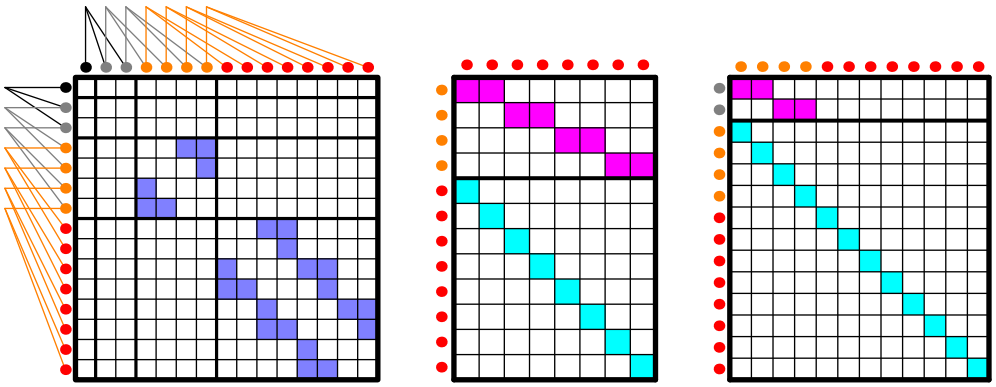


Figure 5.9: Shape of S (left), W_3 (center) and W_2 (right) for equal binary trees using a standard admissibility condition (only neighbors at each level are not admissible). Blue blocks correspond to admissible blocks, magenta blocks are M2M matrices and cyan blocks represent identity matrices. The input and output index sets of W_3 are $\mathscr{S}_3$ and $\mathscr{S}_2$ respectively. Those of W_2 are $\mathscr{S}_2$ and $\mathscr{S}_1$. The trees (or tree restrictions) are somehow "flattened" to obtain index sets, as represented on the left and on the top of the operator S.

Remark 5.2.4. *These definitions of $W_\dagger$ and $V_\dagger$ are consistent with the two stopping criteria MaxDepth and Ncrit (see Sect. 2.2.1.2) for the 2^d-tree construction. This is straightforward for MaxDepth and to extend it to Ncrit, one just has to notice that a hierarchical decomposition does not have to be composed of same size elements at each level and that we allowed elements of a level of a partition to be equal to another at the next level. This means that, under this framework, all mappings between particles and multipole/local expansions are operated on the leaves of the tree representation of the hierarchical decomposition, but a certain amount of tree nodes on entire tree branches may be equal.*

To be more precise, we compare on Fig. 5.8 the effective needed tree structure to store $(\mathcal{U}_k)_k$ (right) to its tree representations in Fig. 5.2 (left), used for the formulas we provided.

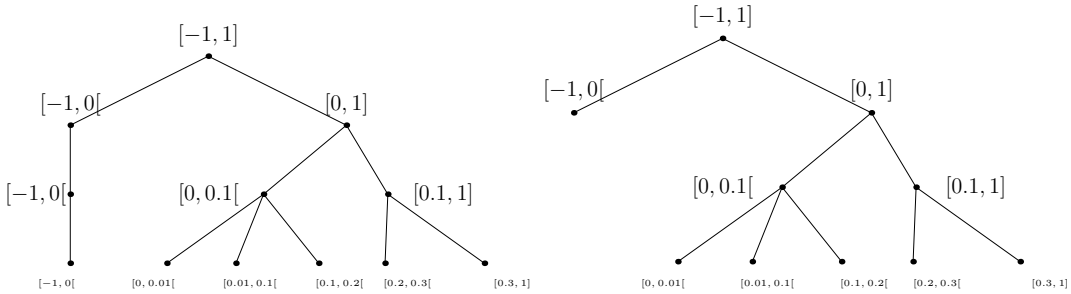


Figure 5.8: Tree representations of $(\mathcal{U}_k)_k$ with (left) and without (right) storage of the redundant elements.

Hence, $W^\dagger$ is theoretically applied to the leaf $[-1,0[$ of the tree representation $\mathscr{H}$ of $(\mathcal{U}_k)_k$ corresponding (i.e. at level $Depth(\mathscr{H})$) but this application can be practically done at the cell corresponding to $[-1,0[$ at the first tree level. This shortcut is taken on the effective tree structure.

Recalling in Fig. 5.4 the binary tree representation of $(\mathcal{V}_k)_k$ of Exp. 5.1.1 and still using the strict MAC, we can represent as in Fig. 5.9 the matrix form of these operators by flattening this tree to index their input and output spaces.

In terms of interaction graph, we obtain something similar to the treecode graph, except that the reuse of blocks both for interactions and assembly of other blocks adds another hierarchical aspect.

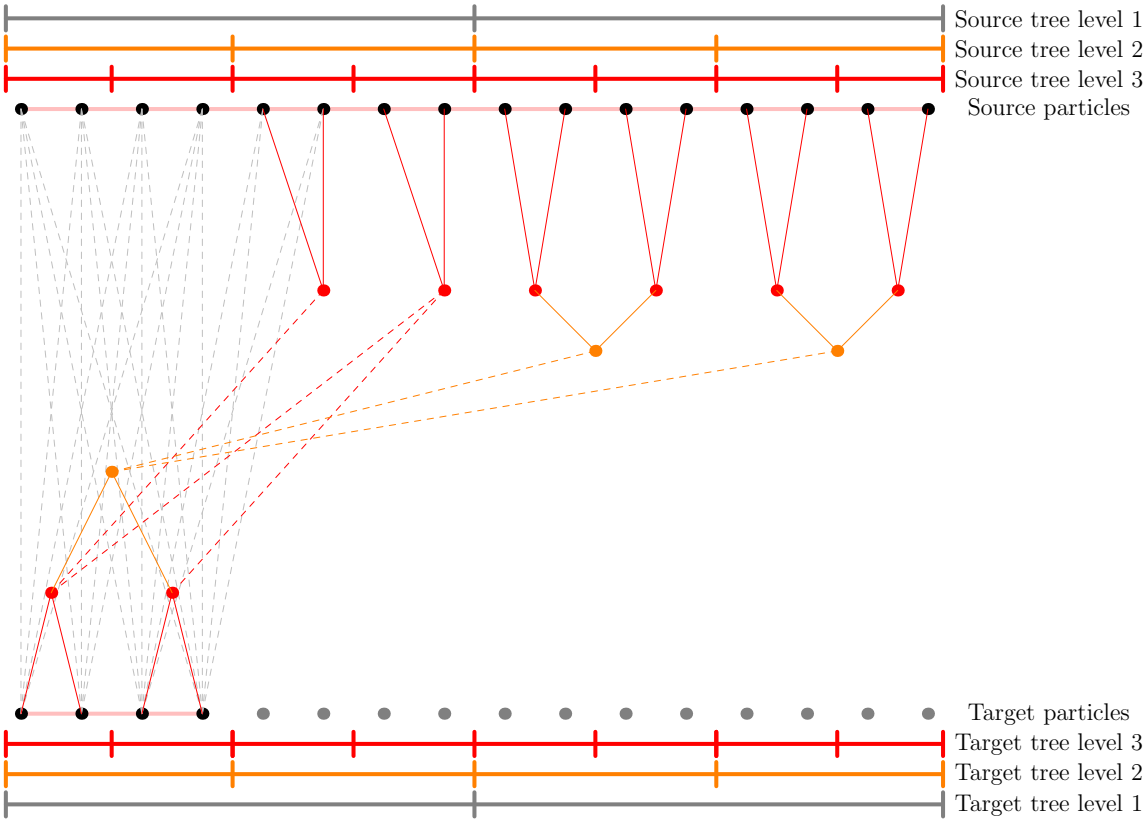


Figure 5.10: Interaction graph for the FMM. The near field part is represented in dashed gray. The hierarchical assembly of far field approximations in the source cells (i.e. application of W) are depicted with continuous lines using the level colors of Fig. 5.4. Using the same conventions, the application of W^* is also depicted on the target using continuous lines. The *cell-cell* interactions are represented with dashed lines using the level colors of the involved cells. Concerning the MAC, we suppose that the particles of a target cell are well-separated from a source cell if the two cells are well-separated according to the strict MAC of Sect. 2.2.4.1 and as illustrated in Fig. 5.5.

Such a graph is depicted on Fig. 5.10. One may notice that this interaction graph is sparser than the treecode one (see Fig. 5.7), which is itself sparser than the hierarchical matrix one (see Fig. 5.6), which is obviously itself sparser than the dense matrix one (see Fig. 5.3). This illustrates how the FMM approach can perform compared to the two other methods when considering in terms of tree relations.

The multipole and local expansions, as well as the rest of the FMM terminology, can also be defined using the framework we presented in this chapter (see Def. 5.2.3).

Definition 5.2.3. *Let s (resp. t) be a source (resp. target) cell. Then $\mathbb{C}[s \cap Y]$ (resp. $\mathbb{C}[t \cap X]$) is said to be the **charges** (resp. **potentials**) associated to $\mathbb{C}[s \cap Y]$ (resp. $\mathbb{C}[t \cap X]$).*
*Let $r_{\mathscr{S}}$ (resp. $r_{\mathscr{T}}$) be a rank distribution over $\mathscr{S}$ (resp. $\mathscr{T}$). For any $s \in \mathscr{S}$ (resp. $t \in \mathscr{T}$), an element of $\mathbb{C}^{r_{\mathscr{S}}(s)}$ (resp. $\mathbb{C}^{r_{\mathscr{T}}(t)}$) is called a **multipole** (resp. **local**) **expansion** in the cell s (resp. t).*

In practice, starting from Eq. 5.18, the vector obtained by

$$(W_{Level(s)+1,\dagger} \cdot q)_{|_s}$$

is named **the** multipole expansion associated to a source cell $s \in \mathscr{S}$, for q a vector of charges. In

the same manner, **the** local expansion associated to $t \in \mathscr{T}$ is the vector given by

$$(S \cdot W \cdot q)_{|_t}.$$

5.2.3.2 Grids

As charges or potentials are associated to particles, in practice the coefficients of multipole/local expansions correspond to elements of a given set associated to the involved cell, but not directly linked with its geometry. To be more precise, depending on the FMM formulation, there often exists a set $\mathbb{H}$ such that for any $s \in \mathscr{S}$, $\exists \mathbb{G}_s \subset \mathbb{H}$ (resp. $t \in \mathscr{T}$, $\exists \mathbb{G}_t \subset \mathbb{H}$) with

$$\begin{cases} \mathbb{C}[r_{\mathscr{S}}] \equiv \bigoplus\limits_{s \in \mathscr{S}} \mathbb{C}[\mathbb{G}_s] \\ \mathbb{C}[r_{\mathscr{T}}] \equiv \bigoplus\limits_{t \in \mathscr{T}} \mathbb{C}[\mathbb{G}_t] \end{cases}.$$

We emphasize that $\mathbb{C}[\mathbb{G}_s] \equiv \mathbb{C}^{r_{\mathscr{S}}(s)}$ but the first term is more precise than the second one: it gives an explicit index set to work with. Due to the definition of the rank distribution, $\#\mathbb{G}_s < \infty$ (resp. $\#\mathbb{G}_t < \infty$).

For any cell c, such a $\mathbb{G}_c$ is called the **grid** associated to c. If c is a source cell, then $\mathbb{G}_c$ is a grid associated to a multipole expansion and if c is a target cell, $\mathbb{G}_c$ is a grid associated to a local expansion. To be even more precise, the rank distributions give the cardinal of these grids.

These grids may have (depending on $\mathbb{H}$) useful interpretations (possibly geometric) which lead to optimizations in effective FMM implementations. We present what these grids correspond to for various FMM formulations.

hf-fmm presented in Chap. 3 is based on an integral expansion of the Helmholtz kernel on the unit sphere. Hence, cubatures on the sphere with integration orders depending on the tree levels are needed to derive practical algorithms. The terms of the multipole/local expansions are associated to cubature nodes. Hence, the grids $\mathbb{G}_c$ in *hf-fmm* correspond to these cubature grids and $\mathbb{H}$ denotes the unit sphere.

kifmm presented in Sect. B.1.2 is also based on the discretization of surface integrals (that do not have to be spheres) on surfaces containing each tree cells. The amount of nodes in each cubature does not necessarily have to vary. The grid of a cell corresponds to the cubature nodes on the associated surface. $\mathbb{H}$ thus corresponds to the reference surface (the pattern used for each cell relatively to its center)

Cauchy FMM presented in Sect. B.2.2 is also built through (contour) integral expressions. Hence, the grids are, once again, formed by cubature nodes. Once again, $\mathbb{H}$ corresponds to the contour in the integral.

Interpolation-based FMM In the case of interpolation-based FMM (see Sect. 4.1.2), the grid of a cell is an interpolation grid with nodes located in this cell. FFT-based techniques exploit tensorized equispaced grids (see Sect. 4.1.4) whereas the low-rank approximation approaches usually use tensorized Chebyshev interpolation grids. $\mathbb{H}$ corresponds here to the cell itself.

Fourier-based FMM The FMM entirely expressed in the Fourier domain described in Sect. B.2.1 has grids of the form $\{\mathbf{j} \in \mathbb{Z}^d \,|\, |\mathbf{j}| \leq p\}$ for fixed $p < \infty$ and d being the dimension. It is straightforward to deduce that $\mathbb{H}$ is nothing more than $\mathbb{Z}^d$.

3D Laplace FMM The expansion elements of the 3D Laplace FMM presented in Sect. 2.2.4.3 correspond to pairs of indices associated to a spherical harmonic with order and degree less than a fixed integer (assuming that the approximation order does not vary from one cell to the next). Hence the grids are the truncated $\mathbb{N}_Z$ introduced in Sect. 2.12 (see Rem. 2.2.6). Here, $\mathbb{H}$ corresponds to $\mathbb{N}_Z$.

Remark 5.2.5. *The main important point to understand is that the grids are not necessarily linked with a cell, only the reverse is true. Indeed, all the cells at a same tree level in hf-fmm have the same grid (i.e. the cubature grid at this tree level, see Sect. 3.1.3). The case of the Fourier-based FMM also simply illustrates this idea: the Fourier modes do not depend on a particular cell, but all the multipole expansions in this method are expressed on given Fourier modes.*

The interpolation-based FMM is a more difficult case: the interpolation grids are unique in each cell. However, since on 2^d-trees, the interpolation grids are actually the same up to a translation at a fixed tree level, one may consider that they are equal by expressing the interpolation nodes relatively to the cell center. In this case, the translation is related to the cell but not to the grid. This is one of the starting point of the considerations discussed in Sect. 5.3.

There may exist symmetries in the grids that can be exploited to obtain fast evaluations of the **M2L** operators (or even of the other FMM operators): for instance, the equispaced grids used in the FFT-based techniques in the interpolation-based FMM in order to perform FFTs and Hadamard products instead of low-rank approximations and products (see Sect. 4.1.4). In practice, each FMM formulation is optimized through its specific grids. Nevertheless, we wonder to what extent a general approach for the treatment of the FMM operators through the grid spectrum and according to the tree structure can be achieved. Derivation of such an approach is the purpose of Sect. 5.3.

5.3 Structure of the FMM

We discuss in this section how to take into account the symmetries inherited from well suited choices of hierarchical decompositions and the corresponding tree representations (namely 2^d-trees). Working outside the scope of a specific FMM formulation allows us to specify the necessary conditions for a FMM formulation to exploit these symmetries in order to (possibly) reduce the computational cost. To be more precise, we will work only on *hypotheses* on the entries of the matrices involved in the FMM (**P2M**/**M2M**/**M2L**/**L2L**/**L2P** matrices) without specification on how the coefficients are actually computed. This makes the discussion independent on the FMM formulation.

5.3.1 Symmetries

In this section, we consider two types of invariance on the **M2L** matrices that can be used in practice in the FMM context. The first of them is a translational invariance and is described in Sect. 5.3.1.1 and the second one is a more general group invariance and is described in Sect. 5.3.1.2.

5.3.1.1 Translational invariance

Let us suppose that the entries of $S_{t,s}$ defined in Eq. 5.18, for any $s \in \mathscr{S}$, $t \in \mathscr{T}$, only depend on an operator defined on the finite grids $\mathbb{G}_s \subset \mathbb{H}$ and $\mathbb{G}_t \subset \mathbb{H}$ (as introduced in Sect. 5.2.3.2), where the set $\mathbb{H}$ has to be defined according to the expansion type of the considered FMM formulation. Let $\mathbf{u} := ctr(t) - ctr(s)$, with $ctr(c)$ the center of the smallest ball containing the cell c, and let us also assume that there exists an operator

$$\mathfrak{B}_{\mathbf{u}} : \mathbb{H}^2 \to \mathbb{C}$$

such that for any $\hat{\mathbf{x}} \in \mathbb{G}_t$, $\hat{\mathbf{y}} \in \mathbb{G}_s$,

$$(S_{t,s})_{\hat{\mathbf{x}},\hat{\mathbf{y}}} = \mathfrak{B}_{\mathbf{u}}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \tag{5.19}$$

where $S_{t,s}$ is regarded as an operator from $\mathbb{C}[\mathbb{G}_s]$ to $\mathbb{C}[\mathbb{G}_t]$ and such that $\mathfrak{B}$ verifies a set of invariance properties to be precised.

Remark 5.3.1. *The notation $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ is used to emphasize that these points are not particles but elements in the grids associated to the cells and given relatively to these cells. According to Rem. 5.2.5, in the case of the interpolation-based FMM, they correspond to interpolation nodes whose positions are given relatively to the corresponding cell centers. In this case, the grids are the same for all cells at a same 2^d-tree level (see Rem. 5.2.5).*

$\mathfrak{B}$ is the operator that generates the entries of the **M2L** matrices according to the grids $\mathbb{G}_t$ and $\mathbb{G}_s$ associated to each cell and the vector obtained by the difference $\mathbf{u}$ of the centers of the two interacting cells. Actually, $\mathfrak{B}$ can be seen as a complex valued function involving three variables. The reason why we separate the arguments is because $\mathbf{u}$, as the only term depending both on the source and target cells, "specifies" $\mathfrak{B}$ for a pair of cells.

The main point is that the existence of such a same $\mathfrak{B}$ for all pairs of cells imposes an invariance property, which is described in Prop. 5.3.1 when the grids are equal (see Rem. 5.2.5).

Proposition 5.3.1. *Suppose that there exists $\mathfrak{B}$ as defined in Eq. 5.19 (the same for all pairs of well-separated cells). Let $\tilde{s} = s + \mathbf{z}$ be a cell, $\mathbf{z} \in \mathbb{R}^d$, and $\tilde{t} = t + \mathbf{z}$, with $\mathbb{G}_t = \mathbb{G}_{\tilde{t}}$ and $\mathbb{G}_s = \mathbb{G}_{\tilde{s}}$. Then $S_{t,s} = S_{\tilde{t},\tilde{s}}$.*

Proof.

$$\begin{aligned} ctr(\tilde{t}) - crt(\tilde{s}) &= ctr(t) + \mathbf{z} - (ctr(s) + \mathbf{z}) \\ &= ctr(t) - crt(s) \end{aligned}$$

which implies that for any $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \mathbb{G}_t \times \mathbb{G}_s$,

$$\begin{aligned} (S_{\tilde{t},\tilde{s}})_{\hat{\mathbf{x}},\hat{\mathbf{y}}} &= \mathfrak{B}_{ctr(\tilde{t})-crt(\tilde{s})}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \\ &= \mathfrak{B}_{ctr(t)-crt(s)}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \\ &= (S_{t,s})_{\hat{\mathbf{x}},\hat{\mathbf{y}}}. \end{aligned}$$

□

Remark 5.3.2. *An operator $\mathfrak{B}$ as in Eq. 5.19 exists in most application cases. For instance, in the case of the interpolation-based FMM, one only has to assume that the approximated kernel is translationally invariant to ensure that such $\mathfrak{B}$ exists. If the kernel is not translationally invariant, the **M2L** matrix between t and s and the **M2L** matrix between $t + \mathbf{z}$ and $s + \mathbf{z}$ are not equal in the general case, even if the interpolation grids are the same.*

As a corollary to Prop. 5.3.1, a first result on the structure of S linking a property on the trees $\mathscr{T}$ and $\mathscr{S}$ with the restrictions of S to some particular blocks can be obtained.

Definition 5.3.1. *A Toeplitz-block matrix is a block matrix which is blockwise Toeplitz.*

Remark 5.3.3. *As opposed to the block-Toeplitz matrices of Def. 4.1.5, the blocks of a Toeplitz-block matrix do not have to be Toeplitz themselves.*

Proposition 5.3.2. *Suppose that there exists $\mathfrak{B}$ as defined in Eq. 5.19 (the same for all pairs of well-separated cells). Let $\mathcal{U} := \{t_0, ..., t_n\} \subset \mathscr{T}$ be a finite set of cells such that $\mathbb{G}_{t_i} = \mathbb{G}_{t_j}$ for any $i, j \in [\![0, n]\!]$, $\mathcal{V} := \{s_0, ..., s_m\} \subset \mathscr{S}$ be a finite set of cells such that $\mathbb{G}_{s_i} = \mathbb{G}_{s_j}$ for any $i, j \in [\![0, m]\!]$. If there exists $\gamma \in \mathbb{R}^d$ such that $t_k = t_0 + k\gamma$ and $s_k = s_0 + k\gamma$, then*

$$S_{\mathcal{U},\mathcal{V}} := \begin{bmatrix} S_{t_0,s_0} & \cdots & S_{t_0,s_m} \\ \vdots & \ddots & \vdots \\ S_{t_n,s_0} & \cdots & S_{t_n,s_m} \end{bmatrix}$$

has a Toeplitz-block structure.

Proof. We only need to verify that for any indices $(i,j) \in \{0,...,n-1\} \times \{0,...,m-1\}$,

$$S_{t_{i+1},s_{j+1}} = S_{t_i,s_j}.$$

To do so, one only has to consider any pair $(\hat{\mathbf{x}},\hat{\mathbf{y}}) \in \mathbb{G}_t \times \mathbb{G}_s$ and to observe that

$$\begin{aligned}(S_{t_{i+1},s_{j+1}})_{\hat{\mathbf{x}},\hat{\mathbf{y}}} &= (S_{t_0+(i+1)\gamma,s_0+(j+1)\gamma})_{\hat{\mathbf{x}},\hat{\mathbf{y}}} \\ &= \mathfrak{B}_{t_0+(i+1)\gamma-(s_0+(j+1)\gamma)}(\hat{\mathbf{x}},\hat{\mathbf{y}}) \\ &= \mathfrak{B}_{t_0+i\gamma-(s_0+j\gamma)}(\hat{\mathbf{x}},\hat{\mathbf{y}}) \\ &= (S_{t_i,s_j})_{\hat{\mathbf{x}},\hat{\mathbf{y}}}.\end{aligned}$$

□

To be more precise, if a set of cells in $\mathscr{S}$ and a set of cells in $\mathscr{T}$ can be obtained by the same translation applied to its first element and to the result iteratively, then the restriction of S to these cells is Toeplitz-block. Perfect binary trees restricted to a given level respect this property.

This proposition can be extended to a linear combination of vectors, as provided in Prop. 5.3.3. Application examples of this result are detailed in Exp. 5.3.2.

Definition 5.3.2. *A Toeplitz-0-block matrix is a Toeplitz-block matrix. A block matrix A is Toeplitz-P-block if and only if A is Toeplitz-block with blocks themselves Toeplitz-$(P-1)$-block, $P \in \mathbb{N}^*$.*

Remark 5.3.4. *Let $n_0,...,n_P \in \mathbb{N}^*$ such that $\#\left(\prod_{b=0}^{P}[\![0,n_b]\!]\right) = \#\left([\![0,N]\!]\right) = N+1$. To extend the result of Prop. 5.3.2 to linear combinations of $P+1$ vectors γ_b's, the condition $t_k = t_0 + k\gamma$ of Prop. 5.3.2 becomes*

$$t_k = t_0 + \sum_{b=0}^{P} I_b(k)\gamma_b$$

where $I_b(k)$ denotes the b^{th} component of $(I_0(k),...,I_P(k)) \in \prod_{b=0}^{P}[\![0,n_b]\!]$ which is the k^{th} element in $\prod_{b=0}^{P}[\![0,n_b]\!]$ according to the lexicographical order, and where $I : [\![0,N]\!] \to \prod_{b=0}^{P}[\![0,n_b]\!]$ is the bijection associating to elements of $\prod_{b=0}^{P}[\![0,n_b]\!]$ their lexicographical index in $[\![0,N]\!]$. A similar reformulation has to be done for s_l. Hence, the mappings I and J in Prop. 5.3.3 only aim at switching between the linear combinations of vectors and the indexations of the sets $\mathcal{U}$ and $\mathcal{V}$ according to the the lexicographical order.

Proposition 5.3.3. *Suppose that there exists $\mathfrak{B}$ as defined in Eq. 5.19. Let $\mathcal{U} := \{t_0,...,t_N\} \subset \mathscr{T}$ be a finite set of cells such that $\mathbb{G}_{t_i} = \mathbb{G}_{t_j}$ for any $i,j \in [\![0,N]\!]$, $\mathcal{V} := \{s_0,...,s_M\} \subset \mathscr{S}$ be a finite set of cells such that $\mathbb{G}_{s_i} = \mathbb{G}_{s_j}$ for any $i,j \in [\![0,M]\!]$. We consider the two mappings*

$$I : [\![0,N]\!] \to \prod_{b=0}^{P}[\![0,n_b]\!] \qquad \text{and} \qquad J : [\![0,M]\!] \to \prod_{b=0}^{P}[\![0,m_b]\!]$$

such that $N = \prod_{b=0}^{P} n_b$, $M = \prod_{b=0}^{P} m_b$, $\forall k \in [\![0,N]\!]$, $I(k) = (I_0(k),...,I_P(k))$, $\forall l \in [\![0,M]\!]$, $J(l) = (J_0(l),...,J_P(l))$ and

$$k = \sum_{b=0}^{P} I_b(k) \prod_{q=0}^{b} n_q \text{ and } l = \sum_{b=0}^{P} J_b(l) \prod_{q=0}^{b} m_q.$$

If there exists $\gamma_0,...,\gamma_P \in \mathbb{R}^d$ such that $t_k = t_0 + \sum_{b=0}^{P} I_b(k)\gamma_b$ and $s_l = s_0 + \sum_{b=0}^{P} J_b(l)\gamma_b$ then $S_{\mathcal{U},\mathcal{V}}$ has a block-hierarchical-P-Toeplitz structure.

Proof. Let

$$\mathbb{T} : \prod_{b=0}^{P} ([\![0, n_b]\!] \times [\![0, m_b]\!]) \to \mathbb{C}$$

be defined as

$$\mathbb{T}(k_0, l_0; ...; k_P, l_P) := (\mathcal{S}_{\mathcal{U},\mathcal{V}})_{k,l}$$

with the notation $k_r := I_r(k)$ and $l_r := J_r(l)$, for any $r \in [\![0, P]\!]$. Using Prop. 5.3.2, for any $r \in [\![0, P]\!]$, if $k_i \neq n_r$ and $l_j \neq m_r$, $\forall i, j$, we have

$$\mathbb{T}(k_0, l_0; ...; k_r + 1, l_r + 1; ...; k_P, l_P) = \mathbb{T}(k_0, l_0; ...; k_r, l_r; ...; k_P, l_P).$$

Hence, for any $u = (u_0, ..., u_P) \in \{0, 1\}^{P+1}$ with $u_r = 0$ if $k_r = n_r$ or $l_r = m_r$, we have

$$\mathbb{T}(k_0 + u_0, l_0 + u_0; ...; k_P + u_P, l_P + u_P) = \mathbb{T}(k_0, l_0; ...; k_P, l_P).$$

This last expression corresponds to a characterization of a Toeplitz-P-block matrix. □

Remark 5.3.5. *The explicit definition of I and J are not needed if we allow the matrix $S_{\mathcal{U},\mathcal{V}}$ to be Toeplitz-P-block up to permutations.*

Remark 5.3.6. *Since any Toeplitz-block matrix can be transformed into a block matrix with Toeplitz blocks, Fourier techniques can be applied on blocks corresponding to sets $\mathcal{U}$ and $\mathcal{V}$ as in Prop. 5.3.3. We illustrate this kind of embedding in Exp. 5.3.1. Since the Fourier matrices can be evaluated with linearithmic complexity, the use of such embedding may result in performance gains in practical applications (see Sect. 4.1.4).*

Example 5.3.1. *The Toeplitz-block matrix in the left-hand side of Eq. 5.20 can be seen a matrix with blocks of size 2×2 that are all Toeplitz (see right-hand side of Eq. 5.20) by means of the permutations Q_0 and Q_1. All these blocks can be embedded into circulant matrices (see Sect. 4.1.4) and explicitely diagonalized in a Fourier basis.*

$$\underbrace{\begin{bmatrix} \begin{bmatrix} a & b \\ c & a \end{bmatrix} & \begin{bmatrix} d & e \\ f & d \end{bmatrix} \\ \begin{bmatrix} g & h \\ i & g \end{bmatrix} & \begin{bmatrix} a & b \\ c & a \end{bmatrix} \end{bmatrix}}_{\text{Toeplitz-block matrix}} = Q_0 \underbrace{\begin{bmatrix} \begin{bmatrix} a & d \\ g & a \end{bmatrix} & \begin{bmatrix} b & e \\ h & b \end{bmatrix} \\ \begin{bmatrix} c & f \\ i & c \end{bmatrix} & \begin{bmatrix} a & d \\ g & a \end{bmatrix} \end{bmatrix}}_{\text{Block matrix with Toeplitz blocks}} Q_1 \tag{5.20}$$

This result may seem quite abstract. We keep a general formulation of this proposition so that it can be exploited in very different situations. One may simplify the conclusion saying that, under the hypothesis of Prop. 5.3.3, $S_{\mathcal{U},\mathcal{V}}$ is Toeplitz-block and each of its blocks also is. This can be represented as a tree whose nodes refer to a block of $S_{\mathcal{U},\mathcal{V}}$ and the links between nodes express that a son is a block of the Toeplitz-block expression of its father. The number of levels of this tree is equal to the number of terms in the linear combinations expressing t_k and s_l from t_0 and s_0 respectively. Therefore, according to Rem. 5.3.6, Fourier techniques involving tensor products of Fourier matrices (whose number depends on the number of levels in the tree of the Toeplitz-block decomposition) can be exploited in practice to evaluate a part of the product by $S_{\mathcal{U},\mathcal{V}}$.

Example 5.3.2. *In Fig. 5.11 we present representations of sets $\mathcal{U}$ and $\mathcal{V}$ that can be used to obtain Toeplitz-block structures if the FMM formulation verifies the hypotheses of Prop. 5.3.3. According to the notations of the proposition, the first (left) representation in Fig. 5.11 leads to a Toeplitz-0-block structure since $[\![0, N]\!] = [\![0, M]\!] \equiv [\![0, 3]\!]$. The middle representation uses $[\![0, N]\!] \equiv [\![0, 3]\!] \times \{0\}$ and $[\![0, M]\!] \equiv [\![0, 3]\!] \times [\![0, 1]\!]$ so there is a Toeplitz-1-block structure.*

Finally, the representation on the right is a bit more complicated since it corresponds to $[\![0, N]\!] = [\![0, M]\!] \equiv [\![0, 1]\!]^2$ but the block of $S_{\mathcal{U},\mathcal{V}}$ corresponding to the interaction between the two hatched cells has to be set to zero (since these two cells are not admissible).

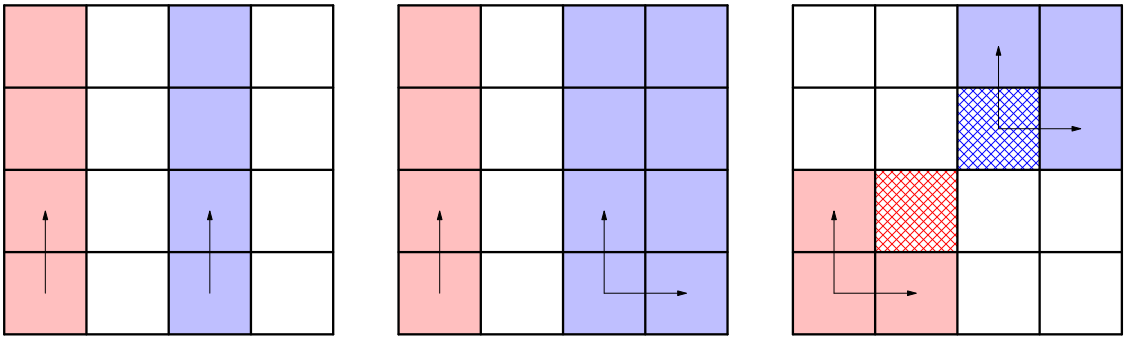


Figure 5.11: Three examples of patterns leading to Toeplitz-block structure on 2^2-trees using the strict MAC (see Sect. 2.2.4.1). Blue: source cells; Red: target cells. The vectors γ_p's of Prop. 5.3.3 are represented on each corresponding group of cells using arrows. There are no *cell-cell* interaction on the top two levels (because of the strict MAC). On the third one, all the red (target) cells of the left and middle picture interact with all the blue (source) ones. On the right picture, the same is true except for the pair of hatched cells (i.e. a single pair). If all the *cell-cell* interactions of one of these figures are handled together, Toeplitz-block structure appear thanks to Prop. 5.3.3.

A practical interesting case is the polynomial interpolation based FMM using equispaced interpolation grids (see Sect. 4.1.4) and 2^d-trees. Indeed, the interactions between groups of neighboring cell concatenations verify the hypothesis of Prop. 5.3.3 but this time, each block of $S_{\mathcal{U},\mathcal{V}}$ is itself (block-d)-Topelitz and a fully circulant embedding can be obtained.

5.3.1.2 Group invariance

Let $\mathcal{G}$ be a finite group acting on $\mathbb{R}^d$ as a group of rotations. We also suppose that $\mathbb{G}$ acts on $\mathbb{H}$. We denote by $g \cdot z$ the result of the action of $g \in \mathcal{G}$ on z (where $z \in \mathbb{R}^d$ or $z \in \mathbb{H}$). We also denote by $g \cdot Z$ the set of elements obtained by action of g on any point of a set Z (where $Z \subset \mathbb{R}^d$ or $Z \subset \mathbb{H}$).

Proposition 5.3.4. *If there exists a finite group $\mathcal{G}$ acting both on $\mathbb{G}_t$, $\mathbb{G}_s$ and $\mathbb{R}^d$ such that*

$$\forall g \in \mathcal{G}, \quad \mathfrak{B}_{g\cdot\mathbf{u}}(g \cdot \hat{\mathbf{x}}, g \cdot \hat{\mathbf{y}}) = \mathfrak{B}_{\mathbf{u}}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$$

with $g \cdot \mathbb{G}_s = \mathbb{G}_{g\cdot s}$ and $g \cdot \mathbb{G}_t = \mathbb{G}_{g\cdot t}$, then:

$$\forall g \in \mathcal{G}, \quad \exists P_g^{(Left)} \in \{0,1\}^{\#\mathbb{G}_t \times \#\mathbb{G}_t}, \; P_g^{(Right)} \in \{0,1\}^{\#\mathbb{G}_s \times \#\mathbb{G}_s}$$

such that

$$S_{g\cdot t, g\cdot s} = P_g^{(Left)} S_{t,s} P_g^{(Right)}$$

where $P_g^{(Left)}$, $P_g^{(Right)}$ are permutation matrices.

Proof. Since $g \cdot \mathbb{G}_r = \mathbb{G}_{g\cdot r}$, $r = t, s$, we have

$$\mathbb{G}_r = g^{-1} \cdot \mathbb{G}_{g\cdot r}. \tag{5.21}$$

We can thus construct a mapping $\mathbb{P}_g^{(r)}$ from $\mathbb{C}[\mathbb{G}_{g\cdot r}]$ to $\mathbb{C}[\mathbb{G}_r]$ such that

$$\left(\mathbb{P}_g^{(r)}\right)_{\mathbf{z}',\mathbf{z}} = \begin{cases} 1 & \text{if } \hat{\mathbf{z}}' = g^{-1} \cdot \hat{\mathbf{z}} \\ 0 & \text{otherwise} \end{cases}. \tag{5.22}$$

We introduce the operator

$$\begin{aligned} \mathbb{B}: \quad \mathbb{R}^d &\to \mathbb{C}^{\#\mathbb{G}_t \times \#\mathbb{G}_s} \\ \mathbf{u} &\mapsto \mathbb{B}(\mathbf{u}) \end{aligned}$$

with

$$\mathbb{B}(\mathbf{u})_{\hat{\mathbf{x}},\hat{\mathbf{y}}} := \mathfrak{B}_{\mathbf{u}}(\hat{\mathbf{x}},\hat{\mathbf{y}}).$$

Let $(\hat{\mathbf{x}},\hat{\mathbf{y}}) \in \mathbb{G}_{g\cdot t} \times \mathbb{G}_{g\cdot s}$. Let $(\hat{\mathbf{x}}',\hat{\mathbf{y}}') \in \mathbb{G}_t \times \mathbb{G}_s$ such that $(\hat{\mathbf{x}}',\hat{\mathbf{y}}') = (g^{-1}\cdot\hat{\mathbf{x}}, g^{-1}\cdot\hat{\mathbf{y}})$. We have:

$$\begin{aligned}
\mathbb{B}(g\cdot\mathbf{u})_{\hat{\mathbf{x}},\hat{\mathbf{y}}} &= \mathfrak{B}_{g\cdot\mathbf{u}}(\hat{\mathbf{x}},\hat{\mathbf{y}}) \\
(hypothesis) &= \mathfrak{B}_{\mathbf{u}}(g^{-1}\cdot\hat{\mathbf{x}}, g^{-1}\cdot\hat{\mathbf{y}}) \\
&= \mathbb{B}(\mathbf{u})_{g^{-1}\cdot\hat{\mathbf{x}},g^{-1}\cdot\hat{\mathbf{y}}} \\
&= \mathbb{B}(\mathbf{u})_{\hat{\mathbf{x}}',\hat{\mathbf{y}}'} \\
(Eq.\,5.22) &= \left(\mathbb{P}_g^{(t)}\mathbb{B}(\mathbf{u})\left(\mathbb{P}_g^{(s)}\right)^{-1}\right)_{\hat{\mathbf{x}},\hat{\mathbf{y}}}.
\end{aligned}$$

Since $\mathbb{B}(g\cdot\mathbf{u})$ is the matrix form of the operator $\mathcal{S}_{g\cdot t,g\cdot s}$ from $\mathbb{C}[\mathbb{G}_{g\cdot s}]$ to $\mathbb{C}[\mathbb{G}_{g\cdot t}]$ and $\mathbb{B}(\mathbf{u})$ is the matrix form of the operator $\mathcal{S}_{t,s}$ from $\mathbb{C}[\mathbb{G}_s]$ to $\mathbb{C}[\mathbb{G}_t]$, the result follows. □

Remark 5.3.7. *The mappings $\mathbb{B}(\mathbf{u})$ in the proof of Prop. 5.3.4 can be interpreted as **M2L** matrices. Each of them is associated to one single $\mathbf{u}$, i.e. to all the pairs of cells such that their center difference is equal to $\mathbf{u}$ (this will be detailed in Sect. 5.3.3). The main point is that for any $g \in \mathcal{G}$, such **M2L** matrix $\mathbb{B}(g\cdot\mathbf{u})$ can be deduced from $\mathbb{B}(\mathbf{u})$ by means of permutations. If these matrices are supposed to be precomputed in a FMM code, the result of Prop. 5.3.4 allows to reduce the cost of the these precomputations (see Sect. 5.3.3).*

The assumptions of Prop. 5.3.4 are called **rotational invariance of $\mathfrak{B}$**.

All the symmetry results are based on specific assumptions on the tree representations $\mathscr{T}$ and $\mathscr{S}$. To be illustrated, we need to specify such tree representations. This is the purpose of the next section.

5.3.2 Symmetries in 2^d-trees

The intensive use of symmetries to reduce the precomputation and application times is in practice linked with 2^d-tree representations of a (regular) hierarchical decomposition based on cubic boxes because of the simple symmetric structure of these trees. We provide in Def. 5.3.3 a definition of regular tree decompositions that handles more general trees than those presented in Chap. 2 (see Sect. 2.2.1.2) but one can think in terms of 2^d-trees as introduced in Sect. 2.2.1.2 in the remainder of this chapter since this is our main application context.

Definition 5.3.3. *A hierarchical decomposition $(\mathcal{U}_k)_k$ is **regular** if and only if $\exists\alpha \in \mathbb{R}^+$ with*

$$\begin{cases}
\forall u_k \in \mathcal{U}_k,\ u_k \text{ is convex,} \\
\forall u_k, v_k \in \mathcal{U}_k,\ |u_k| = |v_k|, \\
\forall u_{k+1} \in \mathcal{U}_{k+1}, \exists! u_k \in \mathcal{U}_k, R \in SO(d), T \in \mathbb{R}^d \text{ such that } u_{k+1} \subset u_k,\ \bar{u}_k = \alpha R\cdot\bar{u}_{k+1} + T
\end{cases}$$

where $|u|$ refers to the volume of $u \in \mathbb{R}^d$, $\bar{u}$ to the closure of u and $SO(d)$ to the set of isometries in $\mathbb{R}^{d\times d}$.

*We denote by 2^d-**tree** the tree representation of a regular hierarchical decomposition in the case where $\alpha = 2^d$.*

Because point clouds are easily encapsulated into d-cubes, the usual 2^d-tree choice corresponds to a regular hierarchical decomposition where all elements of all partitions are d-cubes (i.e. the trees of Sect. 2.2.1.2). We are going to present general results in this context, but the same kind of theory can be adapted to 2^d-trees with d-simplices instead of d-cubes, modifying the underlying group.

This abstract definition of 2^d-trees hides the very simple and intuitive properties we want.

Lemma 5.3.1. *A non leaf node of a 2^d-tree has 2^d sons.*

Proof. The third point of the definition gives

$$|u_k| = \alpha |u_{k+1}| = 2^d |u_{k+1}| \Rightarrow |u_{k+1}| = \frac{|u_k|}{2^d}$$

because R is an isometry and T is a translation, so their application on the cell do not affect its volume.

The volume $|u_k|$ is also equal to the sum of the volumes of its sons (due to the definition of a hierarchical decomposition). So if the node u_k has p sons, then

$$|u_k| = \sum_{q=0}^{p-1} \frac{|u_k|}{2^d} = p \frac{|u_k|}{2^d} \Rightarrow p = 2^d.$$

□

Remark 5.3.8. *In practice, only the cells containing particles may be kept (see Sect. 2.2.1.2). The reasoning of Rem. 5.2.4 still applies on the definition of 2^d-trees provided in Def. 5.3.3.*

Let $\mathfrak{D}_d$ be the group of isometries that keep the d-cube invariant, i.e. the set of linear applications that sends the d-cube into itself. This group is referred to as the hyperoctahedral group in the literature [110]. Let $\mathscr{T}$ be a 2^d-tree representation of X with cubic cells and $\mathscr{S}$ be a 2^d-tree representation of Y with cubic cells (that is the 2^d-trees presented in Chap. 2). Up to now and in the rest of this chapter, we use this choice of 2^d-trees. Let us also suppose that the roots of $\mathscr{T}$ and $\mathscr{S}$ are equal. We easily check that if $\mathscr{T}$ (resp. $\mathscr{S}$) is perfect and centered at 0, then it is invariant under the action of $\mathfrak{D}_d$.

Remark 5.3.9. *If $\mathscr{T}$ (resp. $\mathscr{S}$) is not centered at 0, then there exists a translation of it verifying the invariance property. Hence, any perfect 2^d-tree with cubic cells is invariant under the action of $\mathfrak{D}_d$.*

The important well known result is the following one [110].

Lemma 5.3.2. *$\mathfrak{D}_d \equiv \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d \times \mathfrak{S}_d$, where $\mathfrak{S}_d$ is the group of permutations of a finite set with d elements.*

There are two consequences to this lemma. First, there exists two groups of isometries of $\mathbb{R}^d$, $\mathfrak{D}_d^0$ and $\mathfrak{D}_d^1$ with

$$\begin{aligned} \mathfrak{D}_d > \mathfrak{D}_d^0 &\equiv \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d \\ \mathfrak{D}_d > \mathfrak{D}_d^1 &\equiv \mathfrak{S}_d \end{aligned}$$

where $>$ points out that the right-hand side is a subgroup of the left-hand side, and

$$\begin{aligned} \forall g \in \mathfrak{D}_d, \exists h_0 \in \mathfrak{D}_d^0, h_1 \in \mathfrak{D}_d^1 \text{ with } g = h_0 h_1, \\ \exists h_0 \in \mathfrak{D}_d^1, h_1 \in \mathfrak{D}_d^0 \text{ with } g = h_0 h_1 \end{aligned}$$

(the ordering of $\mathfrak{D}_d^0$ and $\mathfrak{D}_d^1$ are switched between the two lines), meaning that the linear representations of $\mathfrak{D}_d$ can be obtained by products of linear representations of $\mathfrak{D}_d^0$ and $\mathfrak{D}_d^1$ (including the corresponding isometries in $\mathbb{R}^d$). In practice, due to the fact that $\mathfrak{D}_d$ is not abelian for $d \geq 2$, one has to fix an application order on the isometries of $\mathfrak{D}_d^0$ and $\mathfrak{D}_d^1$. The second consequence is that the cardinal of $\mathfrak{D}_d$ can be easily computed

$$\begin{aligned} |\mathfrak{D}_d| &= |\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d| \times |\mathfrak{S}_d| \\ &= |\frac{\mathbb{Z}}{2\mathbb{Z}}|^d \times d! \\ &= 2^d d!. \end{aligned}$$

This provides an explicit estimation of the expected computational gain using these symmetries. It is important to visualize the action of isometries of both $\mathfrak{D}_d^0$ and $\mathfrak{D}_d^1$ on the unit hypercube in order to distinguish them from a geometrical viewpoint because they do not play the same role in the 2^d-trees. A simple way of doing so consists in looking at the representations of these groups as rotation matrices of $\mathbb{R}^{d\times d}$, i.e. as elements of $SO(d)$.

Elements of $\mathfrak{D}_d^0$ can be seen as compositions of at most d reflections with regard to hyperplanes $H_k^0 = 0$ in $\mathbb{R}^d$ where

$$\mathbf{z} = (z_0, ..., z_{d-1}) \in \mathbb{R}^d,\ H_k^0(\mathbf{z}) := z_k. \tag{5.23}$$

Corresponding matrices are filled by zeros, except on the diagonal, which is composed of -1 at entry k if the reflection with regard to the hyperplane H_k^0 is considered, and 1 otherwise. Because $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d$ is a tensorized finite abelian group, these matrices are commutative products of d diagonal matrices with ± 1 on the diagonal and at most one -1 on it, and each of these new matrices correspond to one subgroup $\frac{\mathbb{Z}}{2\mathbb{Z}} \leq \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d$ (that is the reflection with regard to one fixed hyperplane $H_k^0 = 0$).

Remark 5.3.10. *This choice of hyperplanes corresponds to a fixed realization of the action of $\mathfrak{D}_d^0$ on $\mathbb{R}^d$. This is the realization we are looking at in the following, because it fits with the symmetries in 2^d-trees, but one can keep in mind that rotations of these hyperplanes can be considered for other realizations of this group.*

Elements of $\mathfrak{S}_d$ can be considered as permutations of natural axes of $\mathbb{R}^d$. Corresponding matrices are simply permutation matrices. They may also be seen as reflections with regard to hyperplanes

$$\mathbf{z} = (z_0, ..., z_{d-1}) \in \mathbb{R}^d,\ H_{k,l}^1(\mathbf{z}) := z_k - z_l = 0,\ k \neq l.$$

Remark 5.3.11. *These hyperplanes are unique up to the previous realization choice for $\mathfrak{D}_d^0$.*

All these hyperplanes can be easily represented in one, two and three dimensions, as on Fig. 5.12.

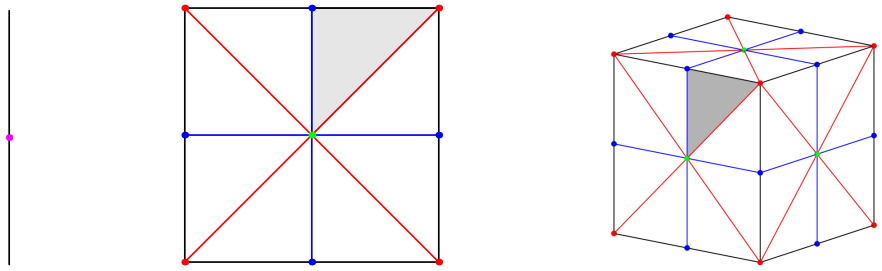


Figure 5.12: Reflection plans for $d = 1, 2, 3$ on the d-cube. Reflections of $\mathfrak{D}_d^0$ (resp. $\mathfrak{D}_d^1$) are represented in blue (resp. red).

The notion of orbit allows to understand how these hyperoctahedral symmetries can be used.

Definition 5.3.4. *Let $\mathcal{G}$ be a finite group acting on a space $\mathfrak{R}$. Let $\mathbf{x} \in \mathfrak{R}$. The set*

$$\langle \mathbf{x} \rangle_{\mathcal{G}} := \{g \cdot \mathbf{x} \,|\, g \in \mathcal{G}\}$$

denotes the orbit of $\mathbf{x}$ with regard to the action of $\mathcal{G}$ (sometimes called $\mathcal{G}$-orbit of $\mathbf{x}$).

The root of $\mathscr{T}$ and $\mathscr{S}$, as well as all their cells, are cubes, so each of them are invariant under the action of $\mathfrak{D}_d$. Each son c' of a non-leaf cell c can be identified from its father using the vector $ctr(c) - ctr(c')$. There are at most 2^d such different vectors per level of $\mathscr{T}$ and $\mathscr{S}$ because all radii of cells at a fixed level are equal. Giving a cell c, we have

$$\{ctr(c) - ctr(c') \mid c' \in Sons(c)\} = \langle ctr(c) - ctr(\bar{c}) \rangle_{\mathfrak{D}_d^0}$$

for any $\bar{c} \in Sons(c)$. An example of such a set is given in Fig. 5.13.

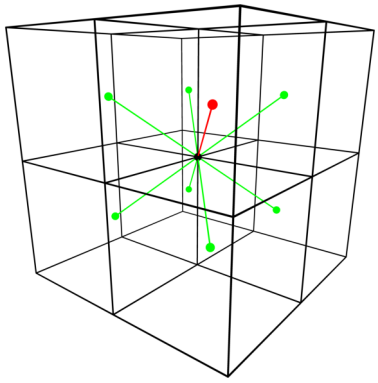


Figure 5.13: Octree (2^3-tree) case: the set of colored vectors is generated by the red one under the action of $\mathfrak{D}_3^0$.

The set of vector depicted on Fig. 5.13 is invariant under the action of $\mathfrak{D}_3^0$. Hence, the results of Prop. 5.3.4 can be adapted to the **M2M** / **L2L** matrices: one can deduce a certain amount of these matrices from others assuming that there exists an operator genereating the entries of these matrices that verifies an invariance property under the action of $\mathfrak{D}_3^0$.

However, we are mainly concerned by the **M2L** case. In Sect. 5.3.3, we present the main application of Props. 5.3.1 and 5.3.4 when dealing with 2^d-trees.

5.3.3 Set of M2L translations

In this section, we express an important result on the **M2L** matrices when dealing with 2^d-trees. We are interested in the minimal set of such different matrices to be computed at each tree level. To provide such result, we need to study the set of translation vectors (**M2L** translations) between source and target cells. For any cell c, we denote by $rad(c)$ the radius of the smallest ball encapsulating c and by $R(c)$ the side length of c.

Proposition 5.3.5. *Let $\mathscr{H} = \mathscr{T}, \mathscr{S}$. $\forall h \in \mathscr{H}$, $\mathbf{z} \in \mathbb{R}^d$ such that $\mathbf{z} = R(h)\sum_{k=0}^{d-1} i_k \mathbf{e}_k$, with $\mathbf{e}_k$ being the k^{th} natural basis vector of $\mathbb{R}^d$ and $i_k \in \mathbb{Z}$. The translation of h by $\mathbf{z}$ can be either a cell $\bar{h}$ of $\mathscr{H}$ or an empty set:*

$$(h + \mathbf{z}) \cap \mathscr{H} = \begin{cases} \bar{h} \in \mathscr{H},\ Level(\bar{h}) = Level(h) \\ \emptyset \end{cases} .$$

Also, if $\exists \mathbf{z}$ such that $h + \mathbf{z} \in \mathscr{H}$, then $\mathbf{z} = R(h)\sum_{k=0}^{d-1} i_k \mathbf{e}_k$.

The result of Prop. 5.3.5 is straightforward because any level of a perfect 2^d-tree with cubic cells forms a finite regular grid. However, this proposition characterizes the cells at a given tree level in terms of translations of a single cell.

Proposition 5.3.6. *Let $\mathcal{Z}_R := \{R\sum_{k=0}^{d-1} i_k \mathbf{e}_k \mid i_k \in \mathbb{Z}\ \forall k \in \{0, ..., d-1\}\} \subset \mathbb{R}^d$. $\forall R \in \mathbb{R}^+$, $\forall \mathbf{z} \in \mathcal{Z}_R$, $\langle \mathbf{z} \rangle_{\mathfrak{D}_d} \subset \mathcal{Z}_R$, where $\mathfrak{D}_d$ acts on $\mathbb{R}^d$ as a group of rotations.*

Proof. First, notice that $\mathcal{Z}_R = R\mathbb{Z}^d$. Since $-\mathbb{Z} = \mathbb{Z}$, $\mathcal{Z}_R$ is invariant under the action of $\mathfrak{D}_k^0$. The same is true for $\mathfrak{D}_k^1$ since $\mathcal{Z}_R = R\mathbb{Z}^d$ is the tensorization of the same space. Therefore, the orbit of any point of $\mathcal{Z}_R$ with regard to $\mathfrak{D}_d$ is included into $\mathcal{Z}_R$. □

This result is important because the difference of the centers of two cells at a same 2^d-tree level (such as a **M2L** translation) is included in $\mathcal{Z}_R$, R being the side length of the cells at this tree level (as a consequence of Prop. 5.3.5). To be more precise, such a difference is included in a particular ball centered at zero and intersected with $\mathcal{Z}_R$.

Corollary 5.3.1. *Let $B_\infty(0,K)$ be the ball defined by the $|\cdot|_\infty$ norm on $\mathbb{R}^d$:*

$$B_\infty(0,K) := \{\mathbf{z} = (z_1,...,z_d) \,|\, |\mathbf{z}|_\infty := \max_k |z_k| \leq K\}.$$

For any $K > 0$, $\mathbf{z} \in B_\infty(0,K) \cap \mathcal{Z}_R$, we have $\langle \mathbf{z} \rangle_{\mathfrak{D}_d} \subset B_\infty(0,K) \cap \mathcal{Z}_R$.

Proof. One only has to observe that the elements of $\mathfrak{D}_d$ acting on $\mathbb{R}^d$ are isometries, so they preserve the norm. The result thus follows Prop. 5.3.6. □

The consequence of Cor. 5.3.1 on the FMM on 2^d-tree with cubic cells is connected to the **M2L** *translations* defined in Def. 5.3.5.

Definition 5.3.5. *Let $\mathscr{T}$ and $\mathscr{S}$ be 2^d-trees with cubic cells and with equal roots. Let $\mathcal{A}$ be an approximability condition. The **set of M2L translations at a level** E of $\mathscr{T}$ and $\mathscr{S}$ is the set*

$$\mathcal{T}_{\mathcal{A},E}(\mathscr{T},\mathscr{S}) := \{ctr(t) - ctr(s) \mid Level(t) = Level(s) = E, t \in \mathscr{T}, s \in \mathscr{S} \text{ and } \mathcal{A}(t,s) = 1\}.$$

The crucial point is that if $Level(t) = Level(s)$ and $\mathcal{A}(t,s) = 1$, then $t = s + (ctr(t) - ctr(s))$. So there exists $\mathbf{z} \in \mathcal{T}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$ such that $t = s + \mathbf{z}$. This means that $\Lambda_{\mathcal{A}}(t)$ can be characterized in terms of translations of t by vectors of $\mathcal{T}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$. For a fixed target cell t, to each of the vectors in $\mathcal{T}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$ (i.e. to each **M2L** translation) is associated a unique **M2L** matrix. If the hypotheses of Prop. 5.3.1 are verified, i.e. if all **M2L** matrices corresponding to the same **M2L** translation are equal for any t, only a small amount of them have to be computed.

Definition 5.3.6. *An approximability condition $\mathcal{A}$ is said to be **normal** if and only if $\exists p_0, p_1$ two polynomials with non-negative coefficients such that:*

$$\mathcal{A}(t,s) = 1 \Leftrightarrow \begin{cases} p_0(rad(t)) + p_1(rad(s)) \leq dist(t,s) \\ p_0(rad(Father(t))) + p_1(rad(Father(s))) > dist(Father(t), Father(s)). \end{cases}$$

The motivation behind the introduction of normal approximability conditions is to determine the bounds (if they exist) of the associated interaction lists and their structure. The expression of such a normal approximability condition allows us to provide the computations for the MACs of Sect. 2.2.4.2

$$rad(s) + rad(t) \leq \eta\, dist(s,t) \quad and \quad Level(s) = Level(t)$$

with $\eta \in \mathbb{R}^+$, but also the condition appearing in the *directional* one of Sect. 4.15 of the form

$$rad(s)^2 \kappa \leq \eta\, dist(t,s) \quad and \quad Level(s) = Level(t)$$

where κ denotes the wavenumber of an oscillatory kernel, e.g. the Helmholtz kernel. These two approximability conditions will be used in the next chapters.

With a normal approximability condition $\mathcal{A}$ and because

$$dist(t,s) = max\{0, |ctr(t) - ctr(s)| - rad(t) - rad(s)\},$$

explicit bounds can be found for the set $\mathcal{T}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$ (see Def. 5.3.5). Indeed, $Level(t) = Level(s) \Rightarrow rad(t) = rad(s)$ and $rad(Father(t)) = 2rad(t)$. In addition, in 2^d-trees, if $rad(t) = rad(s)$,

$$dist(t,s) \leq dist(Father(t), Father(s)) + 2\sqrt{d}\, R(t)$$

thanks to Pythagora's theorem. So we obtain, for well-separated cells t and s:

$$\begin{aligned} & p_0(rad(t)) + p_1(rad(s)) \leq dist(t,s) \leq dist(Father(t), Father(s)) + 2\sqrt{d}\, R(t) \\ \Leftrightarrow\ & p_0(rad(t)) + p_1(rad(s)) \leq |ctr(t) - ctr(s)| - rad(t) - rad(s) \leq dist(Father(t), Father(s)) + 2\sqrt{d}\, R(t) \\ \Leftrightarrow\ & \underbrace{p_0(rad(t)) + p_1(rad(t)) + 2rad(t)}_{=:C^-_{\mathcal{A}}(E)} \leq |\underbrace{ctr(t) - ctr(s)}_{\in \mathcal{T}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})}| \leq \underbrace{dist(Father(t), Father(s)) + 2\sqrt{d}\, R(t) + 2rad(t)}_{=:C^+_{\mathcal{A}}(E)}. \end{aligned}$$

This gives a lower and an upper bound on the norm of the elements of $\mathcal{T}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$, depending only on the radii of cells of the level E. Indeed, we have

$$B_\infty(0, C^-_{\mathcal{A}}(E)) \subseteq \mathcal{T}_{\mathcal{A},E}(\mathscr{T},\mathscr{S}) \subseteq B_\infty(0, C^+_{\mathcal{A}}(E)).$$

Proposition 5.3.7. *Let $\mathcal{Z}_R$ be such as in Prop. 5.3.6 and $\mathcal{A}$ be a normal approximability condition. For any $h \in \mathscr{T} \cup \mathscr{S}$ such that $Level(h) = E$, the following holds:*

$$\mathcal{T}_{\mathcal{A},E}(\mathscr{T},\mathscr{S}) \subseteq \underbrace{\{\mathbf{z} \in \mathcal{Z}_{R(h)} \mid C^-_{\mathcal{A}}(E) \leq |\mathbf{z}| < C^+_{\mathcal{A}}(E)\}}_{=:\tilde{\mathcal{T}}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})}.$$

Such a $\tilde{\mathcal{T}}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$ is invariant under the action of $\mathfrak{D}_d$ thanks to Cor. 5.3.1 (applied on $B_\infty(0, C^-_{\mathcal{A}}(E))$ and $B_\infty(0, C^+_{\mathcal{A}}(E))$). This means that, if the hypotheses of Prop. 5.3.1 are verified, i.e. if all **M2L** matrices corresponding to the same **M2L** translation are equal, there are at most $\#\tilde{\mathcal{T}}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$ different **M2L** matrices at level E of the 2^d-trees.

In addition, if the **M2L** matrices verify the assumptions of Prop. 5.3.4 with $\mathcal{G} \equiv \mathfrak{D}_d$ (i.e. the grids are invariant under the action of $\mathfrak{D}_d$), it is clear that, by means of permutations, a certain amount of these matrices can be deduced from others. Hence, only a prescribed amount of them have to be computed to deduce all the matrices at a given level. This minimal set of matrices corresponds to the **M2L** matrices associated to the **M2L** translations in a *fundamental domain* of $\tilde{\mathcal{T}}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$ under the action of $\mathfrak{D}_d$, i.e. a minimal set such that the action of $\mathfrak{D}_d$ on this set allows to recover $\tilde{\mathcal{T}}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$.

Definition 5.3.7. *A **fundamental domain** of a topological space $\mathbb{X}$ under the action of a group $\mathcal{G}$ is a minimal $\mathcal{F}_{\mathcal{G}}(\mathbb{X}) \subseteq \mathbb{X}$ such that*

$$\mathbb{X} = \bigcup_{f \in \mathcal{F}_{\mathcal{G}}(\mathbb{X})} \langle f \rangle_{\mathcal{G}}.$$

A fundamental domain $\mathcal{F}_{\mathfrak{D}_d}(\tilde{\mathcal{T}}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})) := \mathcal{F}_{\mathfrak{D}_d}(\mathbb{R}^d) \cap \tilde{\mathcal{T}}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$ is not unique and its choice is arbitrary. Examples of fundamental domains for the hyperoctahedral group are depicted in gray on Fig. 5.12.

The conclusions of this section are depicted on Fig. 5.14 in a two-dimensional case. Using the strict MAC (see Sect. 2.2.4.1), there are four possible different shapes of interaction lists (Fig. 5.14 left). Since we are interested in the relative positions between the source and target cells, the union of the associated **M2L** translations corresponds to the entire set of **M2L** translations at this tree level (the arrows in Fig. 5.14 middle). This set is invariant under the action of $\mathfrak{D}_2$ (observing that it is composed of "two squares centered at zero"). Hence, by considering only a suitable choice of fundamental domain of $\mathfrak{D}_2$ and keeping only the **M2L** translations in this fundamental domain, one strongly reduces the size of the set of **M2L** translations (Fig. 5.14 right). If the FMM formulation verifies the hypotheses of Prop. 5.3.4, the matrices associated to other **M2L** translations than these last ones can be deduced from the matrices associated to the **M2L** translations in this fundamental domain by means of permutations.

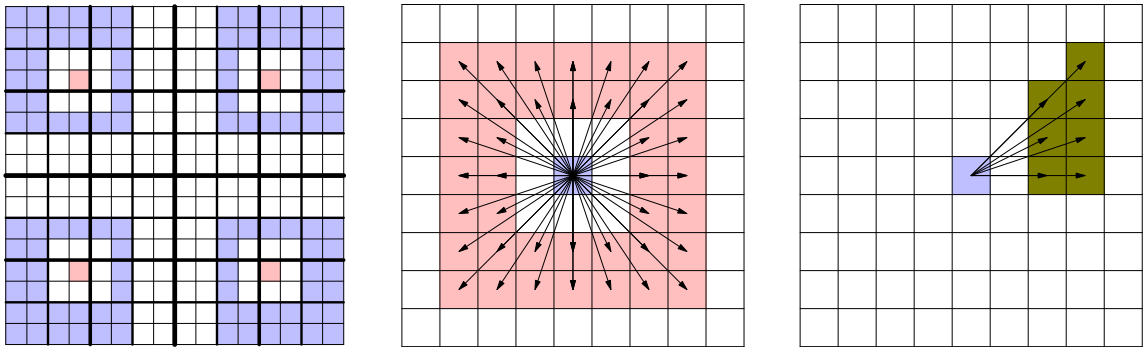


Figure 5.14: Right: interaction lists (blue) of four different target cells (red) using the strict MAC of Sect. 2.2.4.1; Middle: representation of $\tilde{\mathcal{T}}_{\mathcal{A},E}(\mathscr{T},\mathscr{S})$ for the strict MAC (source cell in blue and possible target cells in red); Right: representation of $\mathcal{F}_{\mathfrak{D}_d}(\tilde{\mathcal{T}}_{\mathcal{A},E}(\mathscr{T},\mathscr{S}))$ (green).

Chapter 6

Group-invariant cubature grids in the High-Frequency Fast Multipole Method

Contents

In this chapter, we present a method to explicitly block-diagonalize the elements of a matrix family verifying an invariance property with regard to the action of a given group. An important application of this method is proposed for the kernel-explicit FMM applied to the Helmholtz kernel in the high-frequency regime (i.e. *hf-fmm*). The presented block-diagonalization is used to derive fast FMM operator evaluation based on particular non-product cubature rules.

As mentioned in Sect. 3.2.1, a fast accelerated evaluation of the **M2M**/**L2L** operators is needed in *hf-fmm* to preserve an overall admissible cost. In practice, the most efficient non-destructive algorithms for these operators are based on FFT techniques allowed by the use of product cubature grids on the sphere with an uniform sampling in at least one of the two spherical angles (see Sect. 3.2). Since the integration order of the underlying rules is dictated both by the targeted overall accuracy of *hf-fmm* and by the size of the cells the rules are associated to (in terms of wavelength) as presented in Sect. 3.1.2, the number of nodes in these rules is fixed at each tree level. Thanks to the diagonal expression of the **M2L** matrices in *hf-fmm* (see Sect. 3.1.2), the **M2L** application cost is exactly equal to the number of nodes in the rule associated to the cells of the corresponding far field *cell-cell* interaction (which have to be at the same tree level). There are far more **M2L** applications than **M2M**/**L2L** ones (at least for mean particle distributions) in the global FMM algorithm. From a mathematical viewpoint, due to the diagonal form of the **M2L** operators, the only way of reducing the cost of the **M2L** application is to minimize the number of cubature nodes. The problem when doing so is that the best rules in terms of number of points prevent the use of these fast exact algorithms for the **M2M**/**L2L** operators. Since the **M2M**/**L2L** cost dominates on the upper levels of the 2^d-trees in *hf-fmm*, such a choice of rules should become inefficient in practice.

In the *hf-fmm* approach, because the size of the multipole/local expansions grows with the radius of the corresponding cells in the 2^d-trees, the storage of these expansions may also be a problem when dealing with large particle distributions in terms of wavelength. Hence, the multipole/local expansions having a number of terms equal to the number of cubature nodes in the rule of their tree level, minimizing the number of nodes in these rules also has an impact on the memory footprint of the overall method.

We thus wondered to what extent the use of *quasi-optimal* cubature grids in terms of number of nodes for a fixed integration order can be exploited in *hf-fmm* in order to reduce the **M2L** application cost and the memory footprint while keeping reasonable **M2M**/**L2L** application timings. To do so, we first consider the structure of these quasi-optimal cubature rules on the sphere as presented in Sect. 6.1. A first purely algebraic fast algorithm for product rules is then presented in Sect. 6.2 as an introduction to our approach. Then, we provide a purely algebraic approach to exactly block-diagonalize the matrices coming from the discretization of group-invariant operators evaluated on group-invariant sets in Sect. 6.3. In Sect. 6.4, we detail how the interpolation on the sphere problem using quasi-optimal rules can be written in the framework of our block-diagonalization method and we describe the effective implementation we propose to test our ideas, providing numerical results. This last point relies on a particular choice of quasi-optimal rules particularly interesting in the FMM context and benefiting from the results of Chap. 5.

6.1 Group-invariant cubature grids

In this section we present what are the best known cubature rules on the sphere, discussing first the meaning of the optimality of cubature rules in Sect. 6.1.1. The structure of such rules is linked to their construction. Therefore, we quickly recall the mathematical notions used to build these rules in Sect. 6.1.2 and we describe this structure.

6.1.1 Efficiency of a cubature rule

First of all, we need to compare the existing cubature rules over the sphere. This is done by introducing the efficiency notion.

6.1.1.1 McLaren's efficiency

To compare the efficiency of two cubature rules on the sphere in terms of number of points for a fixed integration order, an efficiency definition is introduced by McLaren in [171] and reads

$$E = \frac{(p+1)^2}{3N} \tag{6.1}$$

with p the order of the exactly integrated spherical harmonics and N the size (number of points) of the considered cubature rule. This measures the quantity we want: the higher the efficiency E, the less operations is performed per **M2L** application in *hf-fmm*. One may notice that the factor 3 is heuristically chosen but leads to Conj. 6.1.1.

Conjecture 6.1.1. *Asymptotically, to integrate exactly the spherical harmonics of order p, the minimal number of nodes is $N = \frac{(p+1)^2}{3}$.*

With regard to this definition and assuming the (asymptotic) validity of Conj. 6.1.1, the optimal cubature rules have an efficiency E close to 1.

6.1.1.2 Case of product rules

The product rules described in Sects. 3.1.1.2 and 3.1.1.3 and used in *hf-fmm* do not have an efficiency close to 1. Indeed, a quick computation reveals that the product uniform rule has an efficiency equal to $\frac{1}{3}$ and the Gauss-Legendre rule has an efficiency $\approx \frac{2}{3}$. Hence, these rules are far from the targeted quasi-optimality mentionned in Conj. 6.1.1.

6.1.1.3 Spherical t-design

Another kind of cubature grid is provided by *spherical t-designs*. A spherical t-design can be defined [132] as a cubature grid with equal weights at for all nodes. There exists an explicit lower bound on the number of nodes in such grids (see [132]). This means that the impact in *hf-fmm* of the use of such rules is bounded. To our knowledge, the spherical t-design on the sphere does not have an efficiency close to one [21, 37].

6.1.2 Invariant rules

An interesting point is that explicit expressions of *quasi-optimal* cubature rules (i.e. with efficiency approximately equal to 1 with regard to Conj. 6.1.1) exist. Starting from results given by Sobolev we are going to recall, highly efficient cubature rules can be designed [10, 153–158, 171]. All these cubature rules are invariant under the action of given finite groups of rotations acting on the sphere.

Definition 6.1.1. *(Invariant cubature rule [193]) Let $\mathcal{G}$ be a given finite group of rotations, a cubature rule invariant under the action of $\mathcal{G}$ is a cubature rule $\mathfrak{Q}$ such that for any $f : \mathbb{S}^2 \to \mathbb{C}$ and for any $g \in \mathcal{G}$*

$$\sum_{(\omega,\lambda)\in\mathfrak{Q}} \omega f(g\cdot\lambda) = \sum_{(\omega,\lambda)\in\mathfrak{Q}} \omega f(\lambda).$$

Such a cubature rule is said to be $\mathcal{G}$-invariant.

Hence, an invariant cubature rule under the action $\mathcal{G}$ refers to a cubature rule such that the set of cubature nodes is invariant under the rotations of $\mathcal{G}$.

Remark 6.1.1. *The number of nodes in an invariant cubature rule is not equal, in general, to a multiple of the group cardinality. Indeed, if an element of $\mathcal{G}$ does not modify the position of a node, the node is not duplicated. Hence, the set of elements of $\mathcal{G}$ that do not modify the position of a point does not lead to different cubature nodes.*

Group	Notation	Cardinal	Platonic solid	Abelian
Tetrahedral	$\mathbb{A}_4$	12	Tetrahedron	No
Octahedral	$\mathbb{S}_4$	24	Cube / Octahedron	No
Dodecahedral	$\mathbb{A}_5$	60	Dodecahedron / Icosahedron	No
Cyclic	$\mathbb{Z}/n\mathbb{Z}$	n	$\emptyset$	Yes
Dihedral	$\mathbb{D}_{2n}$	$2n$	$\emptyset$	No

Figure 6.1: Finite groups acting on the sphere. The three groups in blue correspond to invariance groups of Platonic solids.

Remark 6.1.2. *The definition 6.1.1 implies that for any* $(\omega, \lambda) \in \mathfrak{Q}$, $g \in \mathcal{G}$, *then* $(\omega, g \cdot \lambda) \in \mathfrak{Q}$ *by choosing* $f(\mu) = \begin{cases} 1 \text{ if } \mu = \lambda \\ 0 \text{ otherwise} \end{cases}$. *To be more precise, the cubature weights of nodes corresponding to different rotations of* $\mathcal{G}$ *of a same single node are equal.*

The following theorem can be found in [193] and [10].

Theorem 6.1.1. *Let* $\mathfrak{Q}$ *be a cubature rule invariant under* $\mathcal{G}$. *Then*

$$(\mathfrak{Q} \text{ integrates exactly all } f \in \mathbb{Y}_L) \Leftrightarrow (\mathfrak{Q} \text{ integrates exactly all } f \in \mathbb{Y}_{L,\mathcal{G}})$$

with

$$\begin{cases} \mathbb{Y}_L & := \{Y_l^m \mid l \in [\![0, L]\!], m \in [\![-l, l]\!]\} \\ \mathbb{Y}_{L,\mathcal{G}} & := \{Y_l^m \mid l \in [\![0, L]\!], m \in [\![-l, l]\!] \textbf{ and } Y_l^m(g \cdot \lambda) = Y_l^m(\lambda),\ \lambda \in \mathbb{S}^2, g \in \mathcal{G}\} \end{cases}.$$

This theorem allows to reduce the size of the systems involved in the computation of an invariant cubature rule and to obtain explicit expressions for large integration orders.

6.1.2.1 Finite invariance groups on the sphere

Because of Th. 6.1.1, to our knowledge, examples of cubatures on the sphere with efficiency close to (or even greater than) 1 are invariant under the action of a finite rotation group on the sphere.

There only exists a small number of finite rotation groups acting on the sphere. Following [226], these groups are listed in the Tab. 6.1. All these groups are actually subgroups of $SO(3)$ (see [226]), which corresponds to the set of rotations in $\mathbb{R}^3$. The three blue groups of this array correspond to the invariance groups of the five platonic solids. These solids are depicted on Fig. 6.2.

6.1.2.2 Usual grids

Cubature grids using the same rotational invariance as one of the Platonic solids (finite groups in blue in Tab. 6.1) have been studied in order to provide efficient cubatures. Notice that the cube and the octahedron, as the dodecahedron and the icosahedron, have the same invariance group. A remarkable example of cubature grid with an efficiency greater than one is given in [171], using the icosahedral symmetry group, exact up to the 14^{th} order. In [10] a way of constructing grids invariant under the action of the icosahedral group for higher orders is provided, resulting in efficiencies close to 1.

In [153–158], the cubatures developed by Lebedev for a large variety of orders between 5 and 131 have also an efficiency close to 1 and are invariant under the action of the octahedral group (i.e. $\mathfrak{D}_3$ using the convention of Sect. 5.3.2).

Because of their highly technical and time consuming computations, these rules have to be tabulated in order to be used in a FMM scope. In addition, these rules do not have a product expression. The only common structure of their grids comes from their rotational invariance under a finite group of rotations.

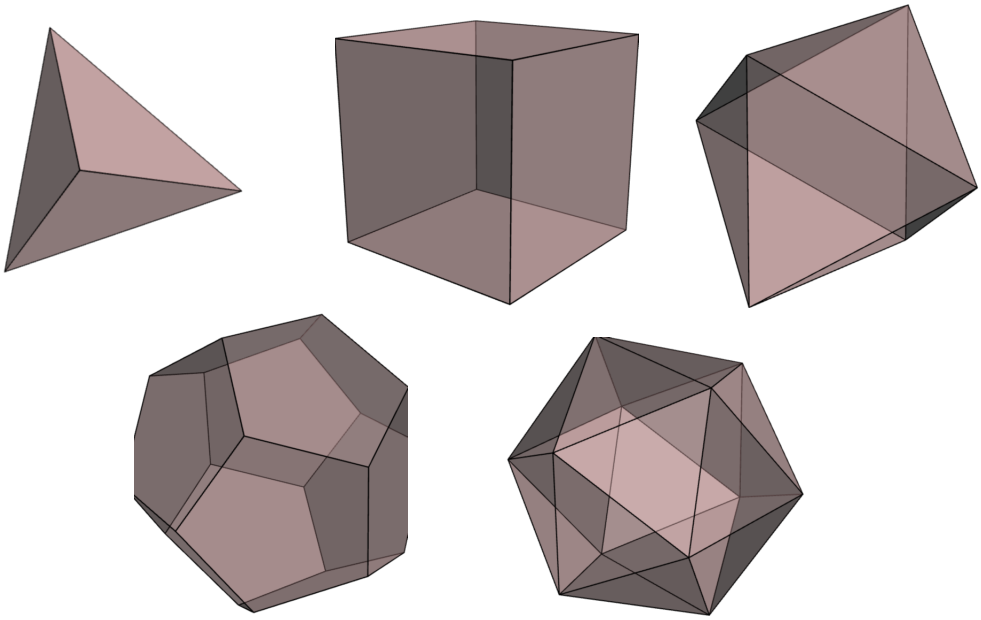


Figure 6.2: Representation of the five platonic solids. From top left to bottom right: tetrahedron, cube, octahedron, dodecahedron, icosahedron.

6.1.2.3 Structure and definitions

Thanks to Rem. 6.1.2, we know that many cubature nodes of these grids are associated to the same cubature weight. We present the general form of an invariant cubature rule in this paragraph, according to the underlying rotation group.

Let $\mathfrak{Q}$ be a cubature rule with cubature grid $\mathcal{N}$ and $F(\mathfrak{Q})$ be the intersection between a fundamental domain of $\mathcal{G}$ acting on the sphere (see Def. 5.3.7) with $\mathcal{N}$, i.e.

$$\begin{aligned} F(\mathfrak{Q}) &:= \mathcal{N} \cap \mathcal{F}_{\mathcal{G}}(\mathbb{S}^2) \\ &= \underset{\substack{\mathcal{N}_0 \subseteq \mathcal{N} \\ \langle \mathcal{N}_0 \rangle_{\mathcal{G}} = \mathcal{N}}}{argmin} \, \#\mathcal{N}_0 \end{aligned}$$

where $\langle \mathcal{N}_0 \rangle_{\mathcal{G}} := \underset{\lambda \in \mathcal{N}_0}{\cup} \langle \lambda \rangle_{\mathcal{G}}$ denotes the union of the $\mathcal{G}$-orbits of elements of $\mathcal{N}_0$ (the $\mathcal{G}$-orbit notion has been introduced in Def. 5.3.4). We will also need in the following another definition provided in Def. 6.1.2 and distinguishing two types of $\mathcal{G}$-orbits.

Definition 6.1.2. *The $\mathcal{G}$-orbit of λ is said to be regular if $\#\langle \lambda \rangle_{\mathcal{G}} = |\mathcal{G}|$, where $|\mathcal{G}|$ refers to the cardinal of $\mathcal{G}$. In the other case, its orbit is said to be singular.*

The set $F(\mathfrak{Q})$ is well defined using this second equality because Def. 6.1.1 implies that $g \cdot \mathcal{N} = \mathcal{N}$ for any $g \in \mathcal{G}$. $\mathcal{N}_0$ may not be unique and its choice is arbitrary. Let ω_λ be the cubature weight of $\lambda \in \mathcal{N}$. We may write the application of $\mathfrak{Q}$ to $f : \mathbb{S}^2 \to \mathbb{C}$, denoted by $\mathcal{Q}[f]$ as

$$\begin{aligned} \mathcal{Q}[f] &:= \sum_{\lambda \in F(\mathfrak{Q})} \omega_\lambda \sum_{g \in \mathcal{H}(\lambda)} f(g \cdot \lambda) \\ &= \sum_{\lambda \in F(\mathfrak{Q})} \omega_\lambda \sum_{g \in \mathcal{G}} f(g \cdot \lambda) \delta_{g \in \mathcal{H}(\lambda)} \end{aligned} \tag{6.2}$$

where

$$\delta_{g \in \mathcal{H}(\lambda)} = \begin{cases} 1 & \text{if } g \in \mathcal{H}(\lambda) \\ 0 & \text{otherwise} \end{cases}$$

using

$$\mathcal{H}(\lambda) := \{g \in \mathcal{G} \,|\, g \cdot \lambda \neq \lambda\} \cup \{id\}.$$

The second equality in Eq. 6.2 somehow considers that all the cubature nodes have a regular $\mathcal{G}$-orbit but omits the multiple entries of f evaluated on the same points. If all the $\mathcal{G}$-orbits are regular, the two lines of Eq. 6.2 involve the same number of non-zero terms.

Since the only structure we can use in the case of general group invariant rules is this invariance under the action of a certain group, we investigated the design of algorithms entirely based on the the symmetries, i.e. on the group structure. Actually, the product rules also are invariant rules and we can start our presentation by studying this first simple case.

6.2 Example of product rules

Before detailing our approach to deal with the general invariant cubature rules for any (possibly non-abelian) group $\mathcal{G}$, we present our ideas in the well-known case of product rules, showing first in Sect. 6.2.1 that these rules (see Sect. 3.1.1.3 and Sect. 6.1.1.3) are also invariant rules. We then propose a first purely algebraic approach to accelerate the interpolation on the sphere in Sect. 6.2.2, exhibiting in Sect. 6.2.3 the associated block-diagonalization. In Sect. 6.2.4 are finally discussed the links and differences between our approach and the existing algorithms dealing with product rules.

6.2.1 Product rules as invariant rules

To fix ideas, we will focus on the Gauss-Legendre product rules of Sect. 3.1.1.3 with order $2L$ (i.e. with $(L+1)(2L+1)$ nodes), but the content of this paragraph is true for any cubature rule on the sphere which is the product of an uniform one along the azimuthal angle and of a symmetric rule with respect to zero in the polar angle. Here, the polar angle θ is covered by the Legendre nodes of a given order while the azimuthal one ϕ is uniformly sampled. If we fix ϕ, we have a distribution of nodes on an arc on the sphere whose half can be deduced from the other one by means of a polar symmetry (i.e. a reflection by the plane $z = 0$). This symmetry is depicted in Fig. 6.3. The group behind this reflection is an abelian group with two elements (since reflection are involutive). This group is isomorphic to $\frac{\mathbb{Z}}{2\mathbb{Z}}$. The corresponding isometry in $SO(3)$ is

$$R_0 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

Notice that on a given arc, the only element that does not have a $\frac{\mathbb{Z}}{2\mathbb{Z}}$-orbit of size 2 is the one on the hyperplane $z = 0$, i.e. corresponding to $\theta = \pi/2$. Hence, the Gauss-Legendre rules of odd orders only have regular $\frac{\mathbb{Z}}{2\mathbb{Z}}$-orbits (since L is odd and $L+1$ nodes are chosen in the polar axis for each sample point on the azimuthal one, see Sect. 3.1.1.3) and the rules of even orders have a single $\frac{\mathbb{Z}}{2\mathbb{Z}}$-orbit corresponding to the root 0 of the associated Legendre polynomial.

Because the azimuthal angle is uniformly sampled (with $2L+1$ points, see Sect. 3.1.1.3), there exists a rotation matrix (i.e. $R_1 \in SO(3)$) expressed as

$$R_1 := \begin{bmatrix} cos\left(\frac{2\pi}{2L+1}\right) & -sin\left(\frac{2\pi}{2L+1}\right) & 0 \\ sin\left(\frac{2\pi}{2L+1}\right) & cos\left(\frac{2\pi}{2L+1}\right) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

such that the rest of the product cubature grid can be deduced from the arc in Fig. 6.3 by applying the different powers of R_1. This is depicted in Fig. 6.4. The set $\{R_1^q \mid q \in \mathbb{N}\} = \{R_1^q \mid q \in [\![0, 2L]\!]\}$ is an abelian finite group of rotations in $SO(3)$ with $2L+1$ elements and generated by R_1. Hence, this

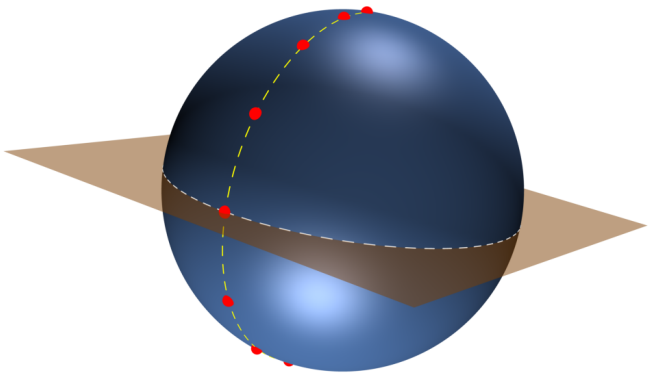


Figure 6.3: Polar symmetry on an arc of a Gauss-Legendre product rule with even order. The symmetry corresponds to a reflexion by the plane $z = 0$ (brown plan).

group is isomorphic to $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$. Because R_1 commutes with R_0, the overall group of rotation acting on the sphere and preserving the Gauss-Legendre product rule is isomorphic to $\frac{\mathbb{Z}}{2\mathbb{Z}} \times \frac{\mathbb{Z}}{(2L+1)\mathbb{Z}} \equiv \frac{\mathbb{Z}}{(2L+1)\mathbb{Z}} \times \frac{\mathbb{Z}}{2\mathbb{Z}} \equiv \frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$. The generator of this abelian cyclic group $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$ is given by R_0R_1. This last point is true only because the number of elements in the discretization along the azimuthal angle is odd[1]. See Fig. 6.5 for an intuitive representation. This figure also represents the regular $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbit of an element of the cubature grid that is not on the hyperplane $z = 0$.

Actually, this group is a subgroup of a larger one: the product between a dihedral group $\mathbb{D}_{2(2L+1)}$ and an abelian one with two elements $\mathbb{Z}/2\mathbb{Z}$. In Fig. 6.6 is represented a well chosen fundamental domain of the group $\mathbb{D}_{2(2L+1)} \times \mathbb{Z}/2\mathbb{Z}$ acting on the sphere as well as the corresponding Gauss-Legendre product rule on it. As one may notice, the original set generating the entire rule (the half arc we started with) is entirely located on the boundaries of this fundamental domain. As a consequence, the $(\mathbb{D}_{2(2L+1)} \times \mathbb{Z}/2\mathbb{Z})$-orbits of these nodes are singular, meaning that they are invariant under a proper subgroup of $\mathbb{D}_{2(2L+1)} \times \mathbb{Z}/2\mathbb{Z}$ which is isomorphic to $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$. This can be understood by exploiting the symmetry of the fundamental domain of $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$ acting on the sphere (see also Fig. 6.6 for a fundamental domain of $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$). Since this set is the projection on the unit sphere of an isosceles triangle, there exists a trivial symmetry on this set: a reflection by the plane passing through the poles and through the middle of the projected base of this triangle.

We have shown how the product Gauss-Legendre rules can be understood as $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-invariant (or even $(\mathbb{D}_{2(2L+1)} \times \mathbb{Z}/2\mathbb{Z})$-invariant) rules. The remaining question concerns the exploitation of this information in order to define efficient schemes for the **M2M**/**L2L** operators. The existing exact algorithms, at least those we are aware of, always use analytic knowledge on the interpolation operator on the sphere. However, these methods fail for non-product rules or even for non-uniformly sampled ones (along the azimuthal angle), so we came up with a novel approach, based on a property

[1] In the case of even discretization, the underlying rotation group is $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$ and can be generated by R_1 **or** R_0R_1 but with different orbits.

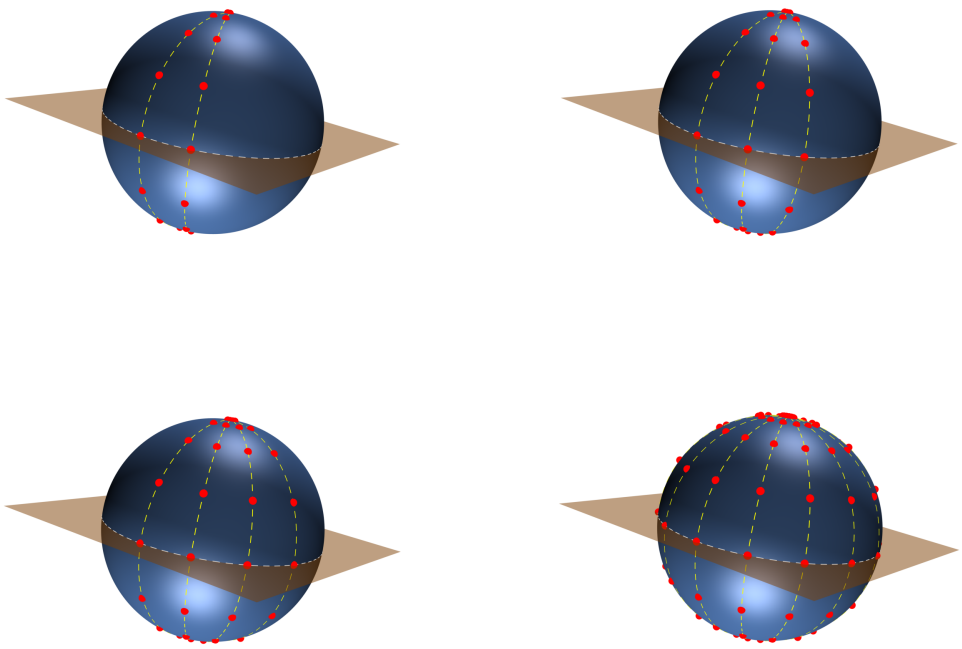


Figure 6.4: Schematic representation of the construction of the entire product Gauss-Legendre cubature rule, starting from a single arc sampled with Gauss-Legendre nodes repeated along the unit sphere by means of the rotation R_1. Single application of R_1 (top left), application of R_1 and R_1^2 (top right), application of R_1, R_1^2 and R_1^3 (bottom left), complete grid (bottom right).

of this operator that can be combined with the group invariant structure of the cubature grids. This approach can be expressed for the product rules as well as for the non-product ones.

6.2.2 Block-diagonalization

In this section, we give a first look of our approach for the fast evaluation of the polynomial interpolation over the sphere in the case of the product (Gauss-Legendre to fix ideas) rules on the sphere. Because the underlying group is abelian (cyclic), the method only involves standard tools in this particular case. This can be applied to a large variety of operators we are going to present. However, there is an important limitation in the choice of the source and target cubature grids when applying our approach to the product rules. We will describe and discuss this constraint.

First of all, we start by introducing a useful denomination.

Definition 6.2.1. *Let $\mathcal{G}$ be a (rotation) group acting on the unit sphere and $\mathcal{F}_{\mathbb{S}^2}(\mathcal{G})$ be a fixed choice of fundamental domain. Any element of $\mathcal{F}_{\mathbb{S}^2}(\mathcal{G})$ is named a germ.*

Hence, any $\mathcal{G}$-invariant cubature grid is generated by a set of germs and the action of $\mathcal{G}$ on these germs. The interpolation operator on the sphere (see Sect. 3.1.3.2)

$$\mathfrak{I}_{\mathbb{S}^2}(\mu,\lambda) = \sum_{l=0}^{L} \frac{2l+1}{4\pi} P_l(\langle \mu, \lambda \rangle),$$

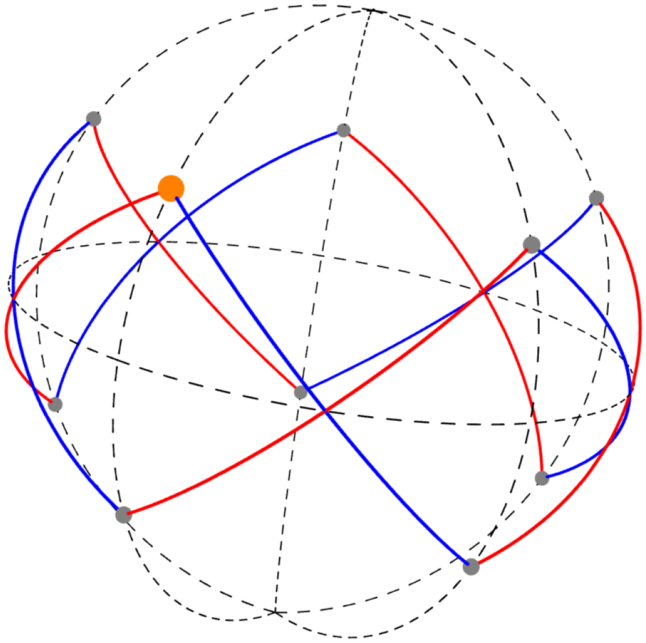


Figure 6.5: $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbit of the orange point with $L = 2$. The odd powers of the generator R_0R_1 are represented in blue and the even in red. The ordered set $\{(R_0R_1)^q \mid q \in \mathbb{N}\}$ alternates between the upper and lower hemispheres. The symmetry with respect to the hyperplan $z = 0$ of the orange point corresponds to the application of $(R_0R_1)^{2L+1} = (R_0R_1)^5$ to this point.

for $\mu, \lambda \in \mathbb{S}^2$, has an interesting form. Indeed, this is a rotationally invariant operator, meaning that for any $R \in SO(3)$, we have

$$\mathfrak{I}_{\mathbb{S}^2}(\mu, R \cdot \lambda) = \mathfrak{I}_{\mathbb{S}^2}(R^{-1} \cdot \mu, \lambda).$$

This property is a consequence of the scalar product in the argument of the Legendre polynomial in the expression of $\mathfrak{I}_{\mathbb{S}^2}$. This explicit expression of the interpolation operator will not be used anymore: we are only interested in its rotational invariance property. To summarize, the key idea is that in the interpolation problem using invariant grids, we are evaluating a rotationally invariant operator (i.e. $\mathfrak{I}_{\mathbb{S}^2}$) on rotationally invariant cubature grids. In the following sections, μ and λ will be nodes in different cubature grids. We shall refer to the cubature grid of λ as the source cubature grid and the one of μ as the target one.

6.2.2.1 Single germ case

Suppose that target and source cubature grids, respectively Ξ_t and Ξ_s, are generated by a single germ each time ($\mathbf{x}_0$ for the target grid and $\mathbf{y}_0$ for the source one) with regular orbits under the action of the same abelian cyclic group of rotations $\mathcal{A} = \{a_0, ..., a_{|\mathcal{A}|-1}\} = \{a_0, a_0^2, ...\}$. We express this in a compact form using the notations $\Xi_t = \mathcal{A} \cdot \mathbf{x}_0$ and $\Xi_s = \mathcal{A} \cdot \mathbf{y}_0$. The corresponding matrix

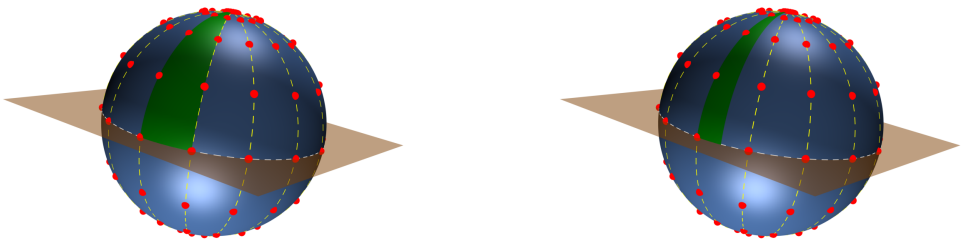


Figure 6.6: Fundamental domains (green) of the cyclic group (left) and of the dihedral group (right) with the corresponding Gauss-Legendre product rule.

$\mathbf{M} \in \mathbb{C}^{|\mathcal{A}|\times|\mathcal{A}|}$ for the interpolation step of the **M2M**/**L2L**, can be expressed as

$$\begin{aligned}
\mathbf{M}_{k,l} &= \mathfrak{I}_{\mathbb{S}^2}(a_k \cdot \mathbf{x}_0, a_l \cdot \mathbf{y}_0) \\
(\textit{Rotational invariance}) &= \mathfrak{I}_{\mathbb{S}^2}(a_l^{-1} \cdot (a_k \cdot \mathbf{x}_0), \mathbf{y}_0) \\
(\mathcal{A}\ \textit{abelian}) &= \mathfrak{I}_{\mathbb{S}^2}(a_{k-l} \cdot \mathbf{x}_0, \mathbf{y}_0) \\
(\mathcal{A}\ \textit{cyclic}) &= \mathfrak{I}_{\mathbb{S}^2}(a_{(k-l)\%|\mathcal{A}|} \cdot \mathbf{x}_0, \mathbf{y}_0).
\end{aligned}$$

This means that $\mathbf{M}$ is a circulant matrix. Hence, this matrix can be diagonalized explicitly in a Fourier basis (see Sect. 4.1.4). Actually, the elements of the Fourier basis are the matrix form of the *irreducible representations* of an abelian cyclic group. This notion will be detailed and extended to other types of groups in Sect. 6.3.

The main point is that there exists a diagonal matrix $D[\mathbf{M}]$ such that $\mathbf{M} = \mathbb{F}^* D[\mathbf{M}]\mathbb{F}$, where $\mathbb{F}$ is the matrix of the corresponding discrete Fourier transform. Such a matrix (and its inverse) can be efficiently computed using the FFT algorithm (see Sect. 4.1.4) with a linearithmic complexity instead of a quadratic one. However, this *toy example* is clearly not realistic since in practice more than one germ are involved in at least one of the target or source cubature grids.

6.2.2.2 Two-germ case

Suppose now that each grid is generated by two germs with regular orbits ($\mathbf{x}_0, \mathbf{x}_1$ for the target grid and $\mathbf{y}_0, \mathbf{y}_1$ for the source one) and still under the action of $\mathcal{A}$. In other words, $T = (\mathcal{A} \cdot \mathbf{x}_0) \cup (\mathcal{A} \cdot \mathbf{x}_1) =: \mathcal{A} \cdot \{\mathbf{x}_0, \mathbf{x}_1\}$ and $S = \mathcal{A} \cdot \{\mathbf{y}_0, \mathbf{y}_1\}$. Following the discussion on the single orbit case (see Sect. 6.2.2.1), the matrix form of the interpolation step in the corresponding **M2M**/**L2L** operator can be written as a block matrix composed of 4 circulant matrices. So there exist 4 diagonal

matrices $D^{0,0}, D^{0,1}, D^{1,0}, D^{1,1}$ such that:

$$\begin{aligned}\mathbf{M} &= \begin{bmatrix}\mathbb{F}^* D^{0,0}\mathbb{F} & \mathbb{F}^* D^{0,1}\mathbb{F} \\ \mathbb{F}^* D^{1,0}\mathbb{F} & \mathbb{F}^* D^{1,1}\mathbb{F}\end{bmatrix} \\ &= \begin{bmatrix}\mathbb{F}^* & 0 \\ 0 & \mathbb{F}^*\end{bmatrix}\begin{bmatrix}D^{0,0} & D^{0,1} \\ D^{1,0} & D^{1,1}\end{bmatrix}\begin{bmatrix}\mathbb{F} & 0 \\ 0 & \mathbb{F}\end{bmatrix} \\ &= \begin{bmatrix}\mathbb{F}^* & 0 \\ 0 & \mathbb{F}^*\end{bmatrix} P^{-1} \underbrace{\begin{bmatrix} D^{0,0}_{0,0} & D^{0,1}_{0,0} & 0 & 0 & \dots \\ D^{1,0}_{0,0} & D^{1,1}_{0,0} & 0 & 0 & \dots \\ 0 & 0 & D^{0,0}_{1,1} & D^{0,1}_{1,1} & \ddots \\ 0 & 0 & D^{1,0}_{1,1} & D^{1,1}_{1,1} & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix}}_{=:D[\mathbf{M}]} P \begin{bmatrix}\mathbb{F} & 0 \\ 0 & \mathbb{F}\end{bmatrix}\end{aligned}$$

where P is a well-chosen permutation matrix. This means that $\mathbf{M}$ can be block-diagonalized in a basis involving a combination of a permutation and of a block-diagonal matrix with non-zero blocks corresponding to Fourier matrices. Obviously, the number of diagonal blocks of $D[\mathbf{M}]$ is equal to the number of rows (or columns) of the circulant matrices (i.e. $|\mathcal{A}|$ since the orbits are supposed to be regular) and each of these blocks is a 2×2 matrix.

6.2.2.3 General case with regular orbits

If we suppose now that $\Xi_t = \underset{\mathbf{x}\in Q_t}{\cup} \mathcal{A}\cdot\mathbf{x}$ and $\Xi_s = \underset{\mathbf{y}\in Q_s}{\cup} \mathcal{A}\cdot\mathbf{y}$ for sets of germs (with regular orbits) Q_t, Q_s of the target and source cubature grids, the block-diagonalization result can be extended to a larger block-diagonalization of $\mathbf{M}$. The number of blocks is still equal to $|\mathcal{A}|$ but their size are $\#Q_t \times \#Q_s$, the products of the number of germs in the two grids. The block-diagonal basis is also still represented by the composition of a permutation matrix with a block-diagonal matrix with non-zero blocks equal to the Fourier matrix of size $|\mathcal{A}| \times |\mathcal{A}|$.

However, one has to keep in mind that we considered only the regular orbits, so only a single size for the Fourier matrices was used. The extension of our approach to the singular orbits is provided in Sect. 6.2.2.4.

6.2.2.4 Treatment of the singular orbit (single orbit case: first method)

In the case of product Gauss-Legendre rules with even order, we saw that the root 0 of the chosen Legendre polynomial or odd degree leads to a singular orbit (see Sect. 6.2.1). Indeed, a corresponding point on the sphere being in the hyperplane $z = 0$, the reflexion R_0 keeps it invariant. Let $\mathbf{z}$ be such a point. We then have

$$\begin{aligned}\langle\mathbf{z}\rangle_{\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}} &= \{(R_0R_1)^q \mid q \in \mathbb{N}\} \\ &= \{R_1^q \mid q \in \mathbb{N}\} \\ &= \langle\mathbf{z}\rangle_{\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}}\end{aligned}$$

and $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$ still is an abelian cyclic group. The *single orbit case* with singular orbits can thus be written in the same way but with a smaller Fourier matrix (i.e. $\mathbb{F}_{2L+1} \in \mathbb{C}^{(2L+1)\times(2L+1)}$ instead of $\mathbb{F}_{2(2L+1)} \in \mathbb{C}^{(2(2L+1))\times(2(2L+1))}$ since $\left|\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}\right| = 2L+1$). Now, still considering the *single orbit case*, if only one of the two target or source orbits is singular (for instance the orbit of the source germ), we can point out that the Fourier matrix $\mathbb{F}_{2L+1}$ is such that

$$\mathbb{F}_{2L+1} = I_{even}^T \mathbb{F}_{2(2L+1)} I_{even}$$

with I_{even} a matrix such that

$$(I_{even})_{k,l} = \begin{cases} 1 \ if \ k = 2l \\ 0 \ otherwise \end{cases} .$$

This can be checked through a quick computation:

$$\begin{aligned} \left(\mathbb{F}_{2L+1}\, q\right)_p &:= \sum_{k=0}^{2L} exp\left(\frac{2i\pi pk}{2L+1}\right) q_k \\ &= \sum_{k=0}^{2L} exp\left(\frac{2i\pi (2p)k}{2(2L+1)}\right) q_k \\ &= \sum_{k=0}^{4L} exp\left(\frac{2i\pi (2p)k}{2(2L+1)}\right) \tilde{q}_k \\ &=: \left(\mathbb{F}_{2(2L+1)}\, \tilde{q}\right)_{2p} \end{aligned}$$

with $\tilde{q} \in \mathbb{C}^{2(2L+1)}$ such that $\tilde{q}_k := \begin{cases} q_{k/2} \ if \ k \ is \ even \\ 0 \ otherwise \end{cases}$.

From an abstract viewpoint, I_{even} can be seen as a mapping from $\mathbb{C}\left[\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}\right]$ to $\mathbb{C}\left[\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}\right]$ that extends an application on a singular $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbit (i.e. here a $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$-orbit) into an application on a regular one, using zero padding. The main point is that there exists a circular embedding of the corresponding matrix $\mathbf{M} \in \mathbb{C}^{(2(2L+1))\times(2L+1)}$ of the interpolation step in the **M2M**/**L2L** operator, denoted by $\mathbf{C} \in \mathbb{C}^{(2(2L+1))\times(2(2L+1))}$, such that $\mathbf{M} = P\mathbf{C}I_{even}$ for a given permutation P. This circulant embedding can be explicitly expressed using:

$$\mathbf{C}_{k,l} := \mathfrak{I}_{\mathbb{S}^2}\left(\left(R_0^k R_1^{\lfloor \frac{k}{2} \rfloor}\right) \cdot \mathbf{x}_0, \left(R_0^l R_1^{\lfloor \frac{l}{2} \rfloor}\right) \cdot \mathbf{y}_0\right).$$

We thus obtain

$$\begin{aligned} \mathbf{M} &= P\mathbf{C}I_{even} \\ &= P\mathbb{F}^*_{2(2L+1)} D[\mathbf{C}] \mathbb{F}_{2(2L+1)} I_{even} \end{aligned}$$

with $D[\mathbf{C}] \in \mathbb{C}^{(2(2L+1))\times(2(2L+1))}$ a particular diagonal matrix. The methodology we just described, i.e. consisting in completing the singular orbits in order to recover regular ones, is related to the second line of Eq. 6.2, where the 'missing' nodes are replaced by zero weights. We may also have dealt with the singular orbit problem in another way, which is the purpose of Sect. 6.2.2.5.

6.2.2.5 Treatment of the singular orbit (single orbit case: second method)

The second methodology we can use to treat the singular orbits does not rely on an extension of $\mathbf{M}$ into a larger space. Actually, in this particular case, we can exploit the relation $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}} < \frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$, meaning that $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$ is a subgroup of $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$ (simple application of the chinese remainder theorem). In Fig. 6.7, we illustrate how a (regular) $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbit can be decomposed into two distinct $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$-orbits. A consequence of Fig. 6.7, in terms of orbits and sets, is that we have the decomposition

$$\langle \mathbf{z} \rangle_{\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}} = \langle \mathbf{z} \rangle_{\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}} \cup \langle R_0 \cdot \mathbf{z} \rangle_{\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}}$$

for any $\mathbf{z} \in \mathbb{S}^2$. If $\mathbf{z}$ is in the hyperplane $z = 0$, then these two $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$-orbits coincide but they are strictly disjoints if $\mathbf{z}$ is not in this hyperplane.

Let $\mathbf{x}_0$ be the germ of the target grid, which is supposed to have a regular $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbit (i.e. which is not on the hyperplane $z = 0$) and let $\mathbf{y}_0$ be the germ of the source grid with a singular

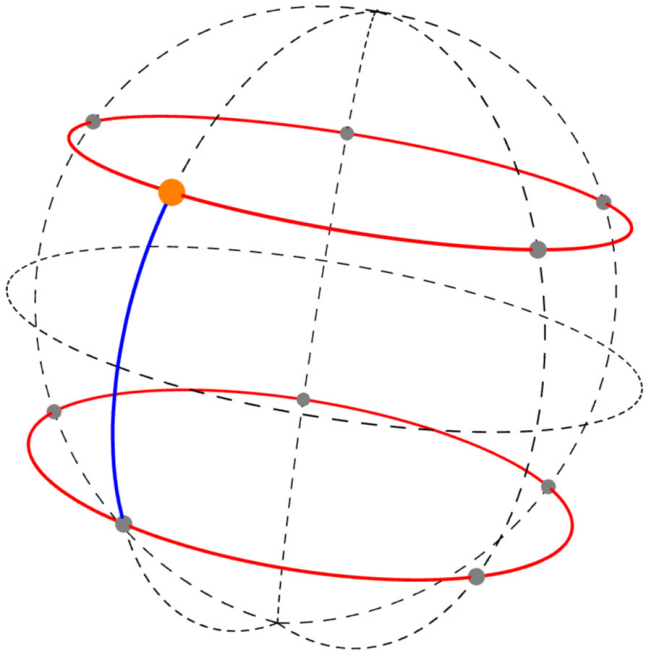


Figure 6.7: The two $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$-orbits (red) included in the orange germ $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbit with $L = 2$. The switching between the two orbits can be done by applying R_0 (blue).

$\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbit, i.e. with a regular $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$-orbit only. The matrix $\mathbf{M}$ of the interpolation step in the **M2M**/**L2L** operators can be decomposed (up to a permutation P) into a block matrix of the form $\mathbf{M} = P \begin{bmatrix} \mathbf{M}_0 \\ \mathbf{M}_1 \end{bmatrix}$, with $\mathbf{M}_0, \mathbf{M}_1 \in \mathbb{C}^{(2L+1)\times(2L+1)}$ and such that

$$\begin{cases} (\mathbf{M}_0)_{k,l} := \mathfrak{I}_{\mathbb{S}^2}\left(R_1^k \cdot \mathbf{x}_0, R_1^l \cdot \mathbf{y}_0\right) \\ (\mathbf{M}_1)_{k,l} := \mathfrak{I}_{\mathbb{S}^2}\left(R_1^k \cdot (R_0 \cdot \mathbf{x}_0), R_1^l \cdot \mathbf{y}_0\right) \end{cases}.$$

Hence, $\mathbf{M}_0$ and $\mathbf{M}_1$ can both be diagonalized in the same Fourier basis with the matrix $\mathbb{F}_{2L+1}$, giving

$$\begin{aligned} \mathbf{M} &= P \begin{bmatrix} \mathbf{M}_0 \\ \mathbf{M}_1 \end{bmatrix} \\ &= P \begin{bmatrix} \mathbb{F}_{2L+1}^* D[\mathbf{M}_0] \mathbb{F}_{2L+1} \\ \mathbb{F}_{2L+1}^* D[\mathbf{M}_1] \mathbb{F}_{2L+1} \end{bmatrix} \\ &= P \begin{bmatrix} \mathbb{F}_{2L+1}^* & 0 \\ 0 & \mathbb{F}_{2L+1}^* \end{bmatrix} \begin{bmatrix} D[\mathbf{M}_0] \\ D[\mathbf{M}_1] \end{bmatrix} \mathbb{F}_{2L+1} \end{aligned}$$

with $D[\mathbf{M}_0]$ and $D[\mathbf{M}_1]$ two particular diagonal matrices. Compared to the first method for the singular orbits (see Sect. 6.2.2.4), completing such orbits to regular ones, this second approach is less costly to apply. Notice that, applying left and right permutations on $\begin{bmatrix} D[\mathbf{M}_0] \\ D[\mathbf{M}_1] \end{bmatrix}$, we can still obtain a block-diagonal matrix with block size equal to 2×1.

6.2.2.6 Treatment of the singular orbit (cases with more than one orbit)

Depending on the chosen approach among the two ones we proposed, handling the singular orbits in the general case with more than one germ may need an adaptation of the general methodology we described in Sect. 6.2.2.3. Indeed, if the first approach (extanding the singular orbit into regular ones by means of zero padding in Sect. 6.2.2.4) is used, we end up with regular orbits only and the method of Sect. 6.2.2.3 can be directly applied. One only has to be able to find the singular orbits in order to extend them. The main drawback of this approach is that we lose the information about these singular orbits in the final block-diagonalization.

On the other hand, the second approach (see Sect. 6.2.2.5) keeps the information on the singular orbits and minimizes the size of the blocks of the final block-diagonalization. However, the structure is more complicated since we have to distinguish the interactions (with regard to $\mathfrak{I}_{\mathbb{S}^2}$) of the singular orbits with other singular orbits, singular ones with regular ones and regular ones with other regular ones.

6.2.3 The general approach

In this section, we provide the factorization we targeted for matrix of the interpolation over the sphere between two product cubature grids using an uniform sampling along the azimuthal axis and a symmetric one-dimensional rule with regard to 0 on the polar axis. We first extend the definition of group invariant operators to matrices.

Definition 6.2.2. *Let $\mathcal{G} = \{g_1, ..., g_\alpha\}$ be a group with $\alpha := |\mathcal{G}|$. A matrix $\mathbf{M} \in \mathbb{C}^{(M\alpha)\times(N\alpha)}$ is said to be $\mathcal{G}$-invariant if and only if $\exists \mathcal{X} = \{x_0, ..., x_{M-1}\}, \mathcal{Y} = \{y_0, ..., y_{N-1}\}$ two finite sets and an operator $\mathfrak{I} : \mathcal{X} \times \mathcal{Y} \to \mathbb{C}$ such that $\mathcal{G}$ acts on $\mathcal{X}$, $\mathcal{Y}$ and up to permutations, $\mathbf{M}$ is a block matrix with $(k,l)^{th}$ block $\tilde{\mathbf{M}}_{k,l}$ verifying:*

$$\left(\tilde{\mathbf{M}}_{k,l}\right)_{i,j} := \mathfrak{I}\left(g_i \cdot x_k, g_j \cdot y_l\right)$$

$\forall i, j \in [\![1, R]\!]$ and

$$\mathfrak{I}\left(g_i \cdot x, g_i \cdot y\right) = \mathfrak{I}\left(x, y\right)$$

for any $x \in \mathcal{X}$, $y \in \mathcal{Y}$.

In Thm. 6.2.1 is summarized the block-diagonalization described in Sect. 6.2.2. We simplify the result by assuming that all the orbits are regular, but as presented in Sects. 6.2.2.4, 6.2.2.5 and 6.2.2.6, one may easily adapt it to handle the singular orbits.

Theorem 6.2.1. *Let $\alpha \in \mathbb{N}^*$ and $\mathbf{M} \in \mathbb{C}^{(M\alpha)\times(N\alpha)}$ be a $\frac{\mathbb{Z}}{\alpha\mathbb{Z}}$-invariant matrix. Then $\mathbf{M}$ can be block-diagonalized with block-diagonal form $D[\mathbf{M}]$ with block size equal to $N \times M$ and there exist permutation matrices Q_0, Q_1, P_0, P_1 such that*

$$\mathbf{M} = Q_0 \, diag\left(\mathbb{F}_\alpha^*\right) P_1 D[\mathbf{M}] P_0 \, diag\left(\mathbb{F}_\alpha\right) Q_1$$

where $diag(\mathbb{F}_\alpha)$ is a block-diagonal matrix with diagonal blocks all equal to $\mathbb{F}_\alpha$, the matrix of the discrete Fourier transform on α entries.

The result of Thm. 6.2.1 is a straightforward generalization of the content of Sect. 6.3, so we do not provide a detailed proof of it. However, there is a direct application of Thm. 6.2.1 in the context of interpolation between product Gauss-Legendre rules on the sphere. We start by giving a general corollary of Thm. 6.2.1.

Corollary 6.2.1. *Let $\alpha \in \mathbb{N}^*$ and $\mathbf{M} \in \mathbb{C}^{(M\alpha)\times(N\alpha)}$ be a $\frac{\mathbb{Z}}{\alpha\mathbb{Z}}$-invariant matrix. A product by $\mathbf{M}$ can be evaluated in $\mathcal{O}\left(\alpha MN + \alpha log(\alpha)(N+M)\right)$ flops.*

Proof. All the permutation matrices in the decomposition of $\mathbf{M}$ provided in Thm. 6.2.1 can be applied with linear complexity by considering their sparse structure. The products by the block diagonal matrices $diag(\mathbb{F}_\alpha^*)$ and $diag(\mathbb{F}_\alpha)$ can be evaluated using the Fast Fourier Transforms with a complexity $\mathcal{O}(\alpha log(\alpha)(N+M))$. The complexity of the product by $D[\mathbf{M}]$ is equal to $\mathcal{O}(\alpha MN)$ because $\mathbf{M}$ is a block-diagonal matrix with α diagonal blocks of size $M \times N$, which dominates the overall cost. □

Regarding the **M2M**/**L2L** operator cost, Cor. 6.2.1 can be reformulated into Cor. 6.2.2.

Corollary 6.2.2. *Let $L' = 3L+1$. The **M2M**/**L2L** matrix interpolation matrix over the sphere $\mathbf{M}$ between product Gauss-Legendre rules of order L and L' can be evaluated in $\mathcal{O}\left(L^2L'\right)$ flops.*

Proof. The interpolation over the sphere dominates the cost of the **M2M**/**L2L** operators. We thus only focus on this step. Without loss of generality[2], we assume that L' is the order of the target grid and L the one of the source one. Since $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}} < \frac{\mathbb{Z}}{2(2L'+1)\mathbb{Z}}$ and following Sect. 6.2.2.4, the $\frac{\mathbb{Z}}{2(2L'+1)\mathbb{Z}}$-orbits can be decomposed into $(2L'+1)/(2L+1)$ (which is equal to 3 thanks to $L' = 3L+1$) $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbits. Hence, $\mathbf{M}$ is a block matrix with a single column of 3 blocks, each of size $(L'+1)(2L+1) \times (L+1)(2L+1)$.

Indeed, if $L+1$ is even, the source grid is composed of $\frac{L+1}{2}$ $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbits and if $L+1$ is even, this source grid is composed of $\frac{L}{2}$ $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbits and one single $\frac{\mathbb{Z}}{(2L+1)\mathbb{Z}}$-orbit. Hence, the number of columns in each of the 3 blocks of $\mathbf{M}$ is $\frac{L+1}{2} \times 2(2L+1) = (L+1)(2L+1)$ if $L+1$ is even and $\frac{L}{2} \times 2(2L+1) + (2L+1) = (L+1)(2L+1)$ if $L+1$ is odd. The same computation is done for L', giving the number of rows.

Suppose that $L+1$ is even, so that there only are regular $\frac{\mathbb{Z}}{2(2L+1)\mathbb{Z}}$-orbits in the source and target grids. Thus, the 3 blocks of $\mathbf{M}$ can be block-diagonalized (see Thm. 6.2.1) in a Fourier basis, each with $2(2L+1)$ blocks of size $(L'+1)/2 \times (L+1)/2$. The product with these blocks dominate the evaluation cost thanks to the Fast Fourier Transforms that can be used to apply the matrices of the discrete Fourier transforms. There are $3 \times 2(L+1)$ of these products by blocks of size $(L'+1)/2 \times (L+1)/2$. The overall cost is therefore $\mathcal{O}((2L+1)(L'/2)(L/2)) = \mathcal{O}\left(L'L^2\right)$.

If $L+1$ is odd, the same conclusions hold by means of zero-padding (see Sect. 6.2.2.4). □

This method imposes a strong constraint on the choice of the cubature rule orders (i.e. $L' = 3L+1$). Notice that in practice, a good choice of relation between the cubature rules of consecutive tree levels is a simple multiplication by a factor 2 of the rule orders. All the discussion we made in this section can be extended to this case by rounding the number of nodes in the discretization of the azimuthal angle to $2L+2$ instead of $2L+1$. With this rounding, the constraint on the orders in Corl. 6.2.2 can become $L' = 2L$ and we are able to cover this case of doubling the order between two consecutive tree levels. The result of Corl. 6.2.2 still holds in this last case, but the complexity has a better constant (hidden in the "$\mathcal{O}$" notation).

6.2.4 Positioning with regard to existing algorithms

Since we are interested in exact algorithms to perform interpolation over the sphere, we can compare the method we provided in Sect. 3.2, which is referred to as the *group invariant* method, to the other existing algorithms.

The method exploiting an uniform sampling along each spherical axis in Sect. 3.2.2.5 uses more cubature nodes than the group invariant method and allows to deal with any integration orders. The Jakob-Chien-Alpert's method (see Sect. 3.2.2.3) uses the same amount of nodes than the group invariant method for the same integration orders[3] with the same overall complexity, but has

[2] The other case is obtained by transposition.

[3] Actually, the cubature rules are the same in the Jakob-Chien-Alpert's method and in the group invariant one.

no restrictions on the links between the orders of the source and target cubature grids. Hence, our method may seem less interesting than the Jakob-Chien-Alpert's one.

However, our approach also has a lot of advantages. First of all, this can be applied to *any* rotationally invariant kernel (i.e. not only $\mathfrak{I}_{\mathbb{S}^2}$), as opposed to the Jakob-Chien-Alpert's method that uses the explicit form of $\mathfrak{I}_{\mathbb{S}^2}$ to derive a fast algorithm. In addition, the polar symmetry of the Gauss-Legendre rules is handled with the symmetries of the azimuthal discretization at the same time using the group invariant method. Finally, the full symmetries of the **M2L** operator can be exploited if and only if the cubature grids are invariant under the action of $\mathfrak{D}_3$, which is not true for a product rule. However, an invariance under the action of a subgroup of $\mathfrak{D}_3$ can be obtained on product cubature grids by imposing a condition: the discretization of the azimuthal angle has to be a multiple of 4 (see [197]). For an implementation of *hf-fmm* exploiting these symmetries as discussed in Sect. 5.3.3 (we emphasize that in this case, only a subgroup of $\mathfrak{D}_3$ can be used but not the entire group when dealing with product cubature rules), the choice of doubling orders between consecutive tree levels is relevant to preserve this structure if we restrict to this subgroup and these product rules. This means that the constraint of our approach is not a drawback in practice.

To propose our first group invariant method, we used a lot a strong assumptions on the invariance groups of the cubature grids. Thanks to their abelian properties and to the Fast Fourier Transforms, we were able to derive a fast general method for the **M2M**/**L2L** with an overall cost close to the one of the Jakob-Chien-Alpert's method. However, one can remember that the Gauss-Legendre cubature rules are not optimal with regard to the efficiency in Eq. 6.1. The exact methods we are aware of are not designed for the quasi-optimal rules. We thus want to extend our approach to any invariance group.

6.3 Explicit block-diagonalization of group-invariant matrices

We are targeting an explicit block-diagonalization of any $\mathcal{G}$-invariant matrix. The general case of an arbitrary (possibly non-abelian) group involves much more advanced tools than the abelian one. This section aims at presenting these tools and how the effective block-diagonalization is obtained with a general group $\mathcal{G}$. Therefore, we present in Sect. 6.3.1 the bases of the finite group representation theory. In Sect. 6.3.2, we illustrate on the abelian cyclic case of Sect. 6.2 how the objects we introduced allow to understand the block-diagonalization we already provided from the finite group representation theory. Then, the general case of arbitrary group is considered in Sect. 6.3.3, where we restrict ourselves to regular orbits. Finally, in Sect. 6.3.4, we describe how to take into account the singular orbits.

6.3.1 Generalities on group representations

The general abstract tools we work with are the representations of finite groups. Here we briefly recall the basic needed notions to use such tools. In all this section, $\mathcal{G}$ refers to an arbitrary finite group. For any vector space V, we denote by $GL(V)$ the *general linear* group over V, corresponding to the set of isomorphisms on V.

Definition 6.3.1. *A representation ρ of $\mathcal{G}$ on the (finite dimensional) vector space V over $\mathbb{C}$ is a group homomorphism $\mathcal{G} \to GL(V)$.*

In other words, ρ maps any $g \in \mathcal{G}$ on an element $\rho(g) \in GL(V)$ such that, for any $g, h \in \mathcal{G}$, $\rho(gh^{-1}) = \rho(g)\rho(h)^{-1}$. We will consider pairs (ρ, V) to denote a representation of $\mathcal{G}$ on V. If W is a vector subspace of V such that $\rho(g) \cdot W \subseteq W, \forall g \in \mathcal{G}$, then W is said to be $\mathcal{G}$-invariant (or to be a *proper* subspace of V). The restriction (ρ, W) of ρ on W is named a subrepresentation of (ρ, V). Fixing a basis for V, any representation can be expressed as a matrix. Hence, we often refer to the matrix form of a representation instead of the representation itself.

The key element is that a representation can be decomposed into a set of subrepresentations, up to a certain point.

Definition 6.3.2. *A representation (ρ, V) of $\mathcal{G}$ is said to be irreducible if and only if $V \neq \{0\}$ and there is no $\mathcal{G}$-invariant vector subspace of V except $\{0\}$ and V itself. If there is such a proper subspace, then (ρ, V) is said to be reducible.*

This concept of reducibility has a direct linear algebra interpretation. Suppose that there exists V_1 and V_2 two proper vector subspaces of V such that $V = V_1 \oplus V_2$. We denote by $M(g)$ the matrix form of $\rho(g)$ in an arbitrary fixed basis of V and by $M_1(g)$ and $M_2(g)$ those of (ρ, V_1) on g and (ρ, V_2) on g respectively. Then, there exists $P \in GL(V)$ with the equality :

$$P^{-1}M(g)P = M_1(g) + M_2(g) = \begin{bmatrix} \mathbf{M}_1(g) & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_2(g) \end{bmatrix} \tag{6.3}$$

where $\mathbf{M}_i(g)$ is a matrix of size $dim(V_i) \times dim(V_i), i \in \{1, 2\}$. This expression can be interpreted as a partial block-diagonalization of $M(g)$ and such a form exists if (ρ, V) is reducible. If at least one of the subrepresentations $(\rho, V_i), i \in \{1, 2\}$ is itself reducible, this procedure can be repeated until irreducible representations are reached.

We are thus searching a decomposition of V of the form :

$$V = V_1 \oplus ... \oplus V_p \tag{6.4}$$

where $p \in \mathbb{N}^*$, each V_i is a $\mathcal{G}$-invariant vector subspace of V, and we want the subrepresentation of $\mathcal{G}$ on V_i to be irreducible. The relation in Eq. 6.3 being a matrix form of the decomposition, we need to define a similar concept for representations themselves. This is given by the *sum* of two representations.

Definition 6.3.3. *Let (ρ, V_1) and (η, V_2) be two representations of G, V_1 and V_2 being two proper subspaces of V. The sum of those two representations is the representation $(\mu, V_1 \oplus V_2)$ such that $\forall g \in \mathcal{G}$*

$$\mu(g) = \begin{bmatrix} \rho(g) & \mathbf{0} \\ \mathbf{0} & \eta(g) \end{bmatrix}.$$

We use the notation $\mu =: \rho + \eta$.

The sum of representations on different vector subspaces of a same vector space in Def. 6.3.3 uses an abusive notation referring to the inclusion in Eq. 6.3. To be more precise, giving bases B_1 and B_2 of V_1 and V_2 seen as subspaces of V respectively and a basis on V such that $V_1, V_2 \subseteq V$, the sum $\rho(g) + \eta(g)$ refers to the sum of the matrix form of $\rho(g)$ and $\eta(g)$ in B_1 and B_2 respectively. Then, based on this definition, we can provide a first theorem that can be found in [189] and which is a consequence of the Maschke's theorem (see also [180] Chap. 7 Sect. 1.3 Prop. 1.32).

Theorem 6.3.1. *([189] Sect. 1.4, Thm. 2) Every representation is a sum of irreducible representations.*

Since the number of irreducible representations of a finite group is finite, Thm. 6.3.1 claims that any representation can be expressed as a *finite* sum of irreducible representations. Let $Irrep(\mathcal{G})$ be the set of irreducible representations of $\mathcal{G}$ with cardinal $p := \#Irrep(\mathcal{G})$. Considering a representation (ρ, V), there exist V_i, $i \in [\![1, p]\!]$, vector subspaces of V such that (ρ_i, V_i) is an irreducible representation of $\mathcal{G}$ with $\rho = \sum_{i=1}^{p} \rho_i$ and

$$V = \bigoplus_{i \in \{1,...,p\}} V_i. \tag{6.5}$$

The dimensions of the decomposition elements in Eq. 6.5 are connected to $\mathcal{G}$.

Definition 6.3.4. *Let (ρ, V) be a representation of $\mathcal{G}$. The degree $d(\rho)$ of this representation is the dimension of V.*

In the general case, $\sum_{\rho \in Irrep(\mathcal{G})} d(\rho)$ does not have to be equal to $|\mathcal{G}|$. The link between the degree $d(\rho)$ of elements $\rho \in Irrep(\mathcal{G})$ and the cardinal of $\mathcal{G}$ is expressed in Thm. 6.3.2.

Theorem 6.3.2. *([189], Sect. 2.4, Corl. 2)* $|\mathcal{G}| = \sum_{\rho \in Irrep(\mathcal{G})} d(\rho)^2$.

There is an important consequence of this result. Fixing a basis, the expression in Eq. 6.3 is a diagonal matrix if and only if the degree of the irreducible representations is always 1. This is not true for an arbitrary group. This notion of degree does not depend on the basis choice but only on the structure of $\mathcal{G}$.

The elements of the decomposition in Eq. 6.5 can actually be classified following the relations between them. The idea is to group together all the elements of the decomposition of V that are isomorphic.

Definition 6.3.5. *Two representations (ρ, V) and (η, W) of $\mathcal{G}$ are said to be isomorphic if there exists an isomorphism $\psi : V \to W$ such that for any $g \in \mathcal{G}$, $\psi(\rho(g) \cdot v) = \eta(g) \cdot \psi(v)$, $\forall v \in V$.*

Two representations are isomorphic if they are defined on vector spaces between which there exists an isomorphism preserving the action of $\mathcal{G}$. The irreducible representations can be grouped into equivalence classes: two isomorphic representations belong in the same equivalence class. The sum of all isomorphic irreducible representations in such a class is named an *isotypic component*. Let m be the number of isotypic components of the representation (ρ, V) of $\mathcal{G}$ and $(\rho_i, \tilde{V}_i)$ be the subrepresentation of (ρ, V) associated to the i^{th} isotypic component. Actually, m is equal to the number of conjugacy classes (see Def. 6.3.6) of $\mathcal{G}$ ([189] Sect. 2.5 Thm. 7).

Definition 6.3.6. *Let $h, g \in \mathcal{G}$. We say that h and g are in the same conjugacy class if there exists an element $s \in \mathcal{G}$ such that $h = sgs^{-1}$.*

We thus obtain a coarser decomposition of V than in Eq. 6.5 and that can be written:

$$V = \tilde{V}_1 \oplus ... \oplus \tilde{V}_m.$$

This decomposition is called the canonical decomposition of V. Each element $\tilde{V}_l$ of the canonical decomposition is itself a sum of the k irreducible representations in the associated isotypic component, with k the cardinal of this isotypic component, all isomorphic to any fixed element in this component. Denoting by $\bar{V}_l$ a choice of such element, we use the notation $V_l = k\bar{V}_l$ to emphasize that there exist a set of isomorphisms between the elements of an isotypic component.

The number of irreducible representations of $(\rho_i, \tilde{V}_i)$ is known ([189] Sect. 2.3 Thm. 2) and does not depend on the chosen decomposition ([189] Sect. 2.3 Corl. 1). Therefore, we have (expressing these numbers in terms of degrees of irreducible representations thanks to [189] Sect. 2.1 Prop 1) the following expression of Eq. 6.4:

$$\begin{aligned} V &= \bigoplus_{i \in \{1,...,m\}} \tilde{V}_i \\ &= \bigoplus_{i \in \{1,...,m\}} \bigoplus_{j \in \{1,...,d(\rho_i)\}} V_{i,j} \\ &= \bigoplus_{i \in \{1,...,m\}} d(\rho_i) V_{i,1} \end{aligned} \tag{6.6}$$

where $\tilde{V}_i = \bigoplus_{j \in \{1,...,d(\rho_i)\}} V_{i,j}$ and for a fixed i, all the $V_{i,j}$ are isomorphic with dimension $d(\rho_i)$. We can thus find some bases of $V_{i,j}$ and $V_{i,k}$ in which the matrix expressions of the irreducible representations on $V_{i,j}$ and $V_{i,k}$ are equal for any $g \in \mathcal{G}$. A crucial point is that the first line of Eq. 6.6 (i.e. the canonical decomposition) does not depend on the initial choice of decomposition of V into a sum of irreducible representations ([189] Sect. 2.6 Thm. 8) but the second line does.

Explicit formulas for the projectors on the $\tilde{V}_i$'s and the $V_{i,j}$'s are also provided by the group representation theory. This is the tool we are interested in for our applications. Since the canonical

decomposition does not depend on the initial decomposition of V, the projectors on the isotypic components can be expressed independently of the choice of basis (see [189] Sect. 2.6 Thm. 8). However, these projectors are not sufficiently fine for our purpose and we need the expression of the projectors on the $V_{i,j}$'s. Their general form is provided in Prop. 6.3.1.

Proposition 6.3.1. *([189] Sect. 2.7 Prop. 8) Let (η, V) be a representation of $\mathcal{G}$. Let $\omega^{(l)}(g)$ be a matrix expression on a chosen basis of $V_{l,1}$. Let $P^{(l)}_{i,j}[\eta]$ be the linear map between V and V given by the formula:*

$$P^{(l)}_{i,j}[\eta] = \frac{d(\rho_l)}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \omega^{(l)}_{i,j}(g^{-1}) \eta(g). \tag{6.7}$$

1. *For any $i = 1, ..., m$, $P^{(l)}_{i,i}[\eta]$ is a projector, $Im(P^{(l)}_{i,i}) \subset \tilde{V}_l$ and $P^{(l)}[\eta] := \sum_i P^{(l)}_{i,i}$ is a projector on $\tilde{V}_l$.*

2. *The linear map $P^{(l)}_{i,j}[\eta]$ vanishes on $\tilde{V}_k$, $k \neq l$, as well as on $V_{l,t}$, $t \neq j$. It defines an isomorphism from $V_{l,j}$ to $V_{l,i}$.*

3. *Let $x_1 \in V_{l,1}$, $x_1 \neq 0$ and let $x_i := P^{(l)}_{i,1}[\eta](x_1) \in V_{l,i}$. The x_i are linearly independent and generate a vector subspace $W(x_1)$ stable under $\mathcal{G}$ and of dimension $d(\rho_l)$. For each $s \in \mathcal{G}$, we have*

$$\eta(s)(x_i) = \sum_j \omega^{(l)}_{i,j}(s) x_j$$

and $W(x_1)$ is isomorphic to W_l.

4. *If $\left(x^{(1)}_1, ..., x^{(m)}_1\right)$ is a basis of $V_{l,1}$, the representation $\tilde{V}_l$ is the direct sum of the subrepresentations $W(x^{(1)}_1), ..., W(x^{(m)}_1)$ defined in 3).*

The next diagram summarizes the action of the projectors and isomorphisms of Prop. 6.3.1.

$$\begin{array}{ccc} V & \xrightarrow{id} & V \\ \Big\downarrow{\scriptstyle P^{(l)}_{j,j}[\eta]} & & \Big\downarrow{\scriptstyle P^{(l)}_{i,i}[\eta]} \\ V_{l,j} & \xrightarrow{P^{(l)}_{i,j}[\eta]} & V_{l,i} \end{array}$$

This formula relies on a choice of basis for V, i.e. explicit matrix forms for the $\omega^{(l)}$.

6.3.2 Application to the cyclic case

In the abelian cyclic group case, all the notions we presented in Sect. 6.3.1 are straightforward to define. Indeed, all the irreducible representations of an abelian cyclic group are one-dimensional (see [189] Sect. 5.1). Let (η, V) be a representation of $\frac{\mathbb{Z}}{N\mathbb{Z}}$. Thanks to Eq. 6.6, all the isotypic components of V are one-dimensional. Due to Thm. 6.3.2, we have $\left|\frac{\mathbb{Z}}{N\mathbb{Z}}\right| = N = \sum_{\rho \in Irrep(\frac{\mathbb{Z}}{N\mathbb{Z}})} d(\rho)$ which implies that there are N elements in $Irrep\left(\frac{\mathbb{Z}}{N\mathbb{Z}}\right)$. Since the canonical decomposition and the projectors on the isotypic components are independent of the initial decomposition of (η, V) into a sum of irreducible representations, their matrix forms do not depend on any basis and actually correspond to the Fourier modes, i.e. the $e^{2i\pi p \frac{k}{N}}$ (see [189] Sect. 5.1). Hence, the projectors of Prop. 6.3.1 in the abelian cyclic case can be simply written. The $\omega^{(l)}_{i,j}(g)$ in Eq. 6.7 corresponds to the only entry in the matrices $\omega^{(l)}(g)$ and using $g = g_0^k$ for a particular $k \in [\![0, N-1]\!]$, g_0 being a

generator of $\frac{\mathbb{Z}}{N\mathbb{Z}}$, we obtain a reformulation of Eq. 6.7:

$$\begin{aligned} P^{(l)}[\eta] &= \frac{1}{N}\sum_{g\in\mathcal{G}} \omega^{(l)}(g^{-1})\eta(g) \\ &= \frac{1}{N}\sum_{k=0}^{N-1} e^{-2i\pi l\frac{k}{N}}\eta(g_0^k). \end{aligned} \tag{6.8}$$

To link this expression with the block-diagonalization we presented in Sect. 6.2.2, we have to introduce the appropriate representations. Let $g \in \mathcal{G}$, Ξ_s be the source product Gauss-Legendre cubature grid on the sphere and $f \in \mathbb{C}[\Xi_s]$. For any $\mathbf{y} \in \Xi_s$, we define the representation $(\eta_s, \mathbb{C}[\Xi_s])$ of $\frac{\mathbb{Z}}{N\mathbb{Z}}$ as

$$\left(\eta_s(g)f\right)(\mathbf{y}) := f\left(g^{-1}\cdot\mathbf{y}\right). \tag{6.9}$$

Let $\mathcal{Y}$ be the set of germs of Ξ_s for the action of a group of rotations isomorphic to $\frac{\mathbb{Z}}{N\mathbb{Z}}$, i.e. $\Xi_s = \frac{\mathbb{Z}}{N\mathbb{Z}}\cdot\mathcal{Y}$ where N is the order of the abelian symmetry group of this grid and $\frac{\mathbb{Z}}{N\mathbb{Z}}\cdot\mathcal{Y} := \{z\cdot\mathbf{y}\,|\,z\in\frac{\mathbb{Z}}{N\mathbb{Z}}, \mathbf{y}\in\mathcal{Y}\}$. For the sake of clarity, we assume that all the orbits are regular. Defined in the same manner, let $\Xi_t = \frac{\mathbb{Z}}{N\mathbb{Z}}\cdot\mathcal{X}$ be the product Gauss-Legendre target cubature grid and let (η_t, Ξ_t) be the associated representation. Let also $f \in \mathbb{C}\left[\frac{\mathbb{Z}}{N\mathbb{Z}}\cdot\mathcal{Y}\right]$. A polynomial interpolation on the sphere of a function f defined on Ξ_s to a function $\tilde{f}$ defined on Ξ_t by means of $\mathfrak{I}_{\mathbb{S}^2}$ is thus a mapping $K : \mathbb{C}\left[\frac{\mathbb{Z}}{N\mathbb{Z}}\cdot\mathcal{Y}\right] \to \mathbb{C}\left[\frac{\mathbb{Z}}{N\mathbb{Z}}\cdot\mathcal{X}\right]$ defined by

$$\begin{aligned} \tilde{f}(\mathbf{x}) &:= K(f)(\mathbf{x}) \\ &:= \sum_{\mathbf{y}\in\Xi_s} \mathfrak{I}_{\mathbb{S}^2}(\mathbf{x},\mathbf{y})\, f(\mathbf{y}) \end{aligned}$$

for any $\mathbf{x} \in \Xi_t$.

The main point is that we have $P^{(l)}[\eta_t(g)K(f)] = K\left(P^{(l)}[\eta_s(g)f]\right)$ for any $l \in [\![0, N-1]\!]$. This can be verified with a direct computation using Eq. 6.8:

$$\begin{aligned} P^{(l)}\left[\eta_t(g)K(f)\right](\mathbf{x}) &= \frac{1}{N}\sum_{k=0}^{N-1} e^{-2i\pi l\frac{k}{N}}\left(\eta_t(g_0^k)K(f)\right)(\mathbf{x}) \\ &= \frac{1}{N}\sum_{k=0}^{N-1} e^{-2i\pi l\frac{k}{N}}\sum_{\mathbf{y}\in\Xi_s}\mathfrak{I}_{\mathbb{S}^2}\left(g_0^{-k}\cdot\mathbf{x},\mathbf{y}\right) f(\mathbf{y}) \\ &= \frac{1}{N}\sum_{k=0}^{N-1} e^{-2i\pi l\frac{k}{N}}\sum_{g_0^{-k}\cdot\mathbf{z}\in\Xi_s}\mathfrak{I}_{\mathbb{S}^2}\left(g_0^{-k}\cdot\mathbf{x}, g_0^{-k}\cdot\mathbf{z}\right) f\left(g_0^{-k}\cdot\mathbf{z}\right) \\ &= \frac{1}{N}\sum_{k=0}^{N-1} e^{-2i\pi l\frac{k}{N}}\sum_{\mathbf{z}\in\Xi_s}\mathfrak{I}_{\mathbb{S}^2}(\mathbf{x},\mathbf{z})\, f\left(g_0^{-k}\cdot\mathbf{z}\right) \\ &= \sum_{\mathbf{z}\in\Xi_s}\mathfrak{I}_{\mathbb{S}^2}(\mathbf{x},\mathbf{z})\left(\frac{1}{N}\sum_{k=0}^{N-1} e^{-2i\pi l\frac{k}{N}} f\left(g_0^{-k}\cdot\mathbf{z}\right)\right) \\ &= K\left(P^{(l)}\left[\eta_s(g)f\right]\right)(\mathbf{x}). \end{aligned} \tag{6.10}$$

In addition, thanks to the orthogonality of the $P^{(l)}$'s, we have for $k \neq l$

$$P^{(l)}\left[P^{(k)}\left[\eta_t(g)K(f)\right]\right] = 0. \tag{6.11}$$

In other terms, combining the results of Eqs. 6.10 and 6.11, the $P^{(l)}$'s block-diagonalize the interpolation matrix $\mathbf{M}$ corresponding to the matrix representation of K. Since the explicit form of the representation of $\frac{\mathbb{Z}}{N\mathbb{Z}}$ in Eq. 6.8 actually corresponds to the Fourier modes in the l^{th} line of a (scaled) Fourier matrix, we recovered the block-diagonal expression we provided in Sect. 6.2.2. The exact block-diagonal structure will be detailed in Sect. 6.3.3.

6.3.3 General case

In the general case of a (possibly non abelian) group, the projectors are a bit more complicated than in the abelian case. Their expression depends on the degree of the irreducible representations. The projector formula provided in Sect. 6.3.1 has been exploited in different forms in the literature [11–13, 45, 191], associated with different denominations. As in [11–13] we will use the matrix forms of the projectors with a different scaling than in Eq. 6.7. This matrix form is named *restriction matrices*.

6.3.3.1 Projector construction

Let (η, V) be a representation of $\mathcal{G}$ on V. Let

$$\begin{aligned} V &= \bigoplus_{l\in\{1,\dots,m\}} \tilde{V}_l \\ &= \bigoplus_{i\in\{1,\dots,m\}} d(\rho_i) V_{i,1} \end{aligned}$$

be the associated decomposition into a sum of irreducible representations $\eta = \sum_{i=1}^{m} d(\rho_i)\rho_i$. Actually, one may understand Eq. 6.7 as an improper matrix product:

$$\begin{aligned} P^{(l)}_{i,j}[\eta] &= \frac{d(\rho_l)}{|\mathcal{G}|} \sum_{g\in\mathcal{G}} \omega^{(l)}_{i,j}(g^{-1})\eta(g) \\ &= \frac{d(\rho_l)}{|\mathcal{G}|} \begin{bmatrix} \omega^{(l)}_{i,j}(g_1^{-1}) & \dots & \omega^{(l)}_{i,j}(g_{|\mathcal{G}|}^{-1}) \end{bmatrix} \begin{bmatrix} \eta(g_1) \\ \vdots \\ \eta(g_{|\mathcal{G}|}) \end{bmatrix} \end{aligned}$$

using $\mathcal{G} = \{g_1, \dots, g_{|\mathcal{G}|}\}$. Hence, the concatenation of the matrices involved in the $P^{(l)}_{i,j}$'s for fixed l and i (i.e. varying j) generates a matrix of the form (up to a normalization)

$$E^{(l)}_k := \sqrt{\frac{d(\rho_l)}{|\mathcal{G}|}} \begin{bmatrix} \omega^{(l)}_{k,1}(g_1) & \dots & \omega^{(l)}_{k,1}(g_{|\mathcal{G}|}) \\ \vdots & \ddots & \vdots \\ \omega^{(l)}_{k,d(\rho_l)}(g_1) & \dots & \omega^{(l)}_{k,d(\rho_l)}(g_{|\mathcal{G}|}) \end{bmatrix}.$$

These matrices are named *elementary restriction matrices* in [11–13]. As a consequence of the orthogonality of the projectors in Prop. 6.7, we obtain the following result.

Proposition 6.3.2. *Suppose that the matrices $\omega^{(l)}(g)$ are unitary. The following assertions hold:*

- $E^{(l)}_k \left(E^{(l)}_k\right)^* = Id_{d(\rho_l)}$ *where $Id_{d(\rho_l)}$ denotes the identity matrix.*
- $E^{(l)}_k \left(E^{(l')}_{k'}\right)^* = 0$ *for any $l \neq l'$ and any k, k'.*

Proof. The first point is a simple application of Prop. 6.7. The second point follows a corollary of Schur's lemma stating (see [189], Chap. 2 Corl. 3)

$$\sum_{g\in\mathcal{G}} \left(\omega^{(l)}(g)\right)_{p,i} \left(\omega^{(l)}(g^{-1})\right)_{q,j} = \begin{cases} \frac{|\mathcal{G}|}{d(\rho_l)} & \text{if } p=q, i=j \\ 0 & \text{otherwise} \end{cases}.$$

Since the $\omega^{(l)}(g)$ are unitary matrices, we have

$$\omega^{(l)}(g^{-1}) = \omega^{(l)}(g)^{-1} = \omega^{(l)}(g)^*.$$

We thus obtain the targeted result. □

Remark 6.3.1. *Through a suitable choice of basis, one can always recover unitary matrix representations of the irreducible representations (see [189] Sect. 1.3). Hence, the assumption "$\omega^{(l)}$ is unitary" of Prop. 6.3.2 is always verified in practice.*

Finally, by concatenating all elementary restriction matrices (i.e. for all l and corresponding i's), we end up with a projector of the form

$$\mathbb{E} = \begin{bmatrix} E_1^{(1)} \\ \vdots \\ E_{d(\rho_1)}^{(1)} \\ \hline \\ \vdots \\ \hline \\ E_1^{(m)} \\ \vdots \\ E_{d(\rho_m)}^{(m)} \end{bmatrix}. \tag{6.12}$$

Following Prop. 6.3.2, we obtain

$$\mathbb{E}\mathbb{E}^* = Id_{|\mathcal{G}|}. \tag{6.13}$$

The property in Eq. 6.13 is one of the key results in our block-diagonalization. To be used in practice, the effective representations have to be expressed as matrices. The matrix $\mathbb{E}$ in Eq. 6.12 has to be adapted in order to offer the projectors we need for a general $\mathcal{G}$-invariant matrix. This can be done after introducing the spaces we work on as well as the explicit representations we use, which is the purpose of Sect. 6.3.3.2.

6.3.3.2 Effective representations

Let $\mathcal{Z}$ be a (finite) set of germs and $\mathcal{G}\cdot\mathcal{Z}$ be the finite set generated by the action of $\mathcal{G}$ on $\mathcal{Z}$. Suppose that all $\mathcal{G}$-orbits in $\mathcal{Z}$ are regular. We have the decompositions of $\mathbb{C}[\mathcal{G}\cdot\mathcal{Z}]$:

$$\begin{aligned} \mathbb{C}[\mathcal{G}\cdot\mathcal{Z}] &= \bigoplus_{g\in\mathcal{G}} \mathbb{C}[g\cdot\mathcal{Z}] \\ &= \bigoplus_{\mathbf{z}\in\mathcal{Z}} \mathbb{C}[\langle\mathbf{z}\rangle_{\mathcal{G}}]. \end{aligned} \tag{6.14}$$

The first line of Eq. 6.14 decomposes $\mathbb{C}[\mathcal{G}\cdot\mathcal{Z}]$ in terms of group actions on the set of germs and the second line decomposes this space into $\mathcal{G}$-orbits of the set of germs. It is thus possible to derive a representation of $\mathcal{G}$ on $\mathbb{C}[\mathcal{G}\cdot\mathcal{Z}]$ through the representations of $\mathcal{G}$ in all the elements of the

decompositions in Eq. 6.14. We introduce the representation $(\mu_{\mathbf{z}}, \mathbb{C}[\langle \mathbf{z} \rangle_{\mathcal{G}}])$ for any $\mathbf{z} \in \mathcal{Z}$ such that for any $g \in \mathcal{G}$, $\mu_{\mathbf{z}}(g) \in Aut\left(\mathbb{C}[\langle \mathbf{z} \rangle_{\mathcal{G}}]\right)$ and

$$\left(\mu_{\mathbf{z}}(g)\right)_{\mathbf{z}_1,\mathbf{z}_2} := \begin{cases} 1 & \text{if } g^{-1} \cdot \mathbf{z}_1 = \mathbf{z}_2 \\ 0 & \text{otherwise} \end{cases} \tag{6.15}$$

for any $\mathbf{z}_1, \mathbf{z}_2 \in \langle \mathbf{z} \rangle_{\mathcal{G}}$. This corresponds to the definition we gave in Eq. 6.9. Representations of Eq. 6.15 can be exploited in order to recover a representation on $\mathbb{C}[\mathcal{G} \cdot \mathcal{Z}]$ by means of the decompositions in Eq. 6.14. Indeed, we have the isomorphism

$$\bigoplus_{\mathbf{z} \in \mathcal{Z}} \mathbb{C}[\langle \mathbf{z} \rangle_{\mathcal{G}}] \equiv \mathbb{C}[\mathcal{Z} \times \mathcal{G}]$$

where $\mathcal{Z} \times \mathcal{G}$ denotes the cartesian product between $\mathcal{Z}$ and $\mathcal{G}$, since all $\mathcal{G}$-orbits of elements of $\mathcal{Z}$ are assumed to be regular. To be more precise, we associate to $(\mathbf{z}, g) \in \mathcal{Z} \times \mathcal{G}$ the element $g \cdot \mathbf{z}$.

Let $Id_{\mathcal{Z}}(\mathbf{z})$ be the matrix form of the element in $Aut\left(\mathbb{C}[\mathcal{Z}]\right)$ such that

$$\left(Id_{\mathcal{Z}}(\mathbf{z})\right)_{\mathbf{z}_1,\mathbf{z}_2} := \begin{cases} 1 & \text{if } \mathbf{z} = \mathbf{z}_1 = \mathbf{z}_2 \\ 0 & \text{otherwise} \end{cases}.$$

The representation $(\mu, \mathbb{C}[\mathcal{G} \cdot \mathcal{Z}])$ such that for any $g \in \mathcal{G}$

$$\mu(g) := \sum_{\mathbf{z} \in \mathcal{Z}} Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \mu_{\mathbf{z}}(g) \tag{6.16}$$

is the one we are targeting. One may check that $(\mu, \mathbb{C}[\mathcal{Z} \times \mathcal{G}])$ actually defines a representation of $\mathcal{G}$ by observing that for any $g \in \mathcal{G}$, $\mu(g)$ is a block diagonal matrix with the $\mathbf{z}^{th}$ block equal to $\mu_{\mathbf{z}}(g)$. Thus, $\forall g, h \in \mathcal{G}$, we have

$$\begin{aligned}
\mu(g)\mu(h)^{-1} &= \left(\sum_{\mathbf{z} \in \mathcal{Z}} Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \mu_{\mathbf{z}}(g)\right) \left(\sum_{\mathbf{z} \in \mathcal{Z}} Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \mu_{\mathbf{z}}(h)\right)^{-1} \\
(\mu(h)\ \textit{block-diagonal}) &= \left(\sum_{\mathbf{z} \in \mathcal{Z}} Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \mu_{\mathbf{z}}(g)\right) \left(\sum_{\mathbf{z} \in \mathcal{Z}} Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \mu_{\mathbf{z}}(h)^{-1}\right) \\
(\mu_{\mathbf{z}}\ \textit{representation}) &= \left(\sum_{\mathbf{z} \in \mathcal{Z}} Id_{\mathcal{Z}}\left(\mathbf{z}\right) \mu_{\mathbf{z}}(g)\right) \left(\sum_{\mathbf{z} \in \mathcal{Z}} Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \mu_{\mathbf{z}}(h^{-1})\right) \\
(\textit{block-diagonal structures}) &= \sum_{\mathbf{z} \in \mathcal{Z}} \left(Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \mu_{\mathbf{z}}(g)\right) \left(Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \mu_{\mathbf{z}}(h^{-1})\right) \\
(\textit{product of tensor matrices}) &= \sum_{\mathbf{z} \in \mathcal{Z}} Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \left(\mu_{\mathbf{z}}(g)\mu_{\mathbf{z}}(h^{-1})\right) \\
(\mu_{\mathbf{z}}\ \textit{representation}) &= \sum_{\mathbf{z} \in \mathcal{Z}} Id_{\mathcal{Z}}\left(\mathbf{z}\right) \otimes \left(\mu_{\mathbf{z}}(gh^{-1})\right) \\
&= \mu(gh^{-1}).
\end{aligned}$$

We have obtained representations on $\mathbb{C}[\mathcal{G} \cdot \mathcal{Z}]$. The next step consists in introducing the general projectors on the proper subspaces of $\mathbb{C}[\mathcal{G} \cdot \mathcal{Z}]$ with regard to $\mathcal{G}$.

6.3.3.3 Invariant operator projection

Let $\mathcal{Z}$ be a (finite) set of germs and $\mathcal{G} \cdot \mathcal{Z}$ be the finite set generated by action of $\mathcal{G}$ on $\mathcal{Z}$. Suppose that all the $\mathcal{G}$-orbits in $\mathcal{Z}$ are regular. To build our general projectors on the proper subspaces of

$\mathbb{C}[\mathcal{G}\cdot\mathcal{Z}] \equiv \mathbb{C}[\mathcal{Z}\times\mathcal{G}]$ with regard to $\mathcal{G}$ (i.e. as in Eq. 6.6), we may have a similar reasoning than in Sect. 6.3.3.2. Indeed, let $\mathbb{E}_{\mathcal{Z}} \in Aut\left(\mathbb{C}[\mathcal{Z}\times\mathcal{G}]\right)$ be defined as

$$\mathbb{E}_{\mathcal{Z}} := Id_{\mathcal{Z}} \otimes \mathbb{E} \tag{6.17}$$

with $Id_{\mathcal{Z}}$ the identity in $Aut\left(\mathbb{C}[\mathcal{Z}]\right)$. As for $\mu(g)$ (see Eq. 6.16), $\mathbb{E}_{\mathcal{Z}}$ is a block-diagonal matrix with all diagonal blocks equal to $\mathbb{E}$ (and the number of diagonal blocks is equal to the cardinal of $\mathcal{Z}$). Thus, the result of Prop. 6.3.3 holds.

Proposition 6.3.3. *$\mathbb{E}_{\mathcal{Z}}\mathbb{E}_{\mathcal{Z}}^* = Id_{\mathbb{C}[\mathcal{Z}\times\mathcal{G}]}$, where $Id_{\mathbb{C}[\mathcal{Z}\times\mathcal{G}]}$ denotes the identity in $Aut\left(\mathbb{C}[\mathcal{Z}\times\mathcal{G}]\right)$.*

Proof. We simply check the result with a direct computation:

$$\begin{aligned}
\mathbb{E}_{\mathcal{Z}}\mathbb{E}_{\mathcal{Z}}^* &= \left(Id_{\mathcal{Z}} \otimes \mathbb{E}\right)\left(Id_{\mathcal{Z}} \otimes \mathbb{E}\right)^* \\
&= \left(Id_{\mathcal{Z}} \otimes \mathbb{E}\right)\left(Id_{\mathcal{Z}} \otimes \mathbb{E}^*\right) \\
&= Id_{\mathcal{Z}} \otimes \left(\mathbb{E}\mathbb{E}^*\right) \\
\textit{(Eq. 6.13)} \quad &= Id_{\mathcal{Z}} \otimes Id_{|\mathcal{G}|} \\
&= Id_{\mathbb{C}[\mathcal{Z}\times\mathcal{G}]}.
\end{aligned}$$

□

Now, let $\mathcal{X}$ and $\mathcal{Y}$ be two sets of points with regular $\mathcal{G}$-orbits. Let $\mathbf{M}$ be a linear mapping from $\mathbb{C}[\mathcal{G}\cdot\mathcal{Y}]$ to $\mathbb{C}[\mathcal{G}\cdot\mathcal{X}]$ and such that there exists a $\mathcal{G}$-invariant operator $\mathfrak{I} : (\mathcal{G}\cdot\mathcal{X})\times(\mathcal{G}\cdot\mathcal{Y}) \to \mathbb{C}$ with

$$\mathbf{M}_{\mathbf{x},\mathbf{y}} := \mathfrak{I}\left(\mathbf{x},\mathbf{y}\right) \tag{6.18}$$

for any $\mathbf{x} \in \mathcal{G}\cdot\mathcal{X}$ and $\mathbf{y} \in \mathcal{G}\cdot\mathcal{Y}$ (that is $\mathbf{M}$ is a $\mathcal{G}$-invariant matrix). Up to permutations, $\mathbf{M}$ defines a mapping from $\mathbb{C}[\mathcal{X}\times\mathcal{G}]$ to $\mathbb{C}[\mathcal{Y}\times\mathcal{G}]$. As a consequence of Prop. 6.3.3, we have

$$\mathbf{M} = \mathbb{E}_{\mathcal{X}} \underbrace{\left(\mathbb{E}_{\mathcal{X}}^*\mathbf{M}\mathbb{E}_{\mathcal{Y}}\right)}_{=:D[\mathbf{M}]} \mathbb{E}_{\mathcal{Y}}^*. \tag{6.19}$$

Since the $\mathbb{E}_{\mathcal{Z}}$, $\mathcal{Z} = \mathcal{X}, \mathcal{Y}$ are projectors, a certain amount of entries of the matrix $D[\mathbf{M}]$ vanishes. We exhibit in Sect. 6.3.3.4 a block-diagonal structure of $D[\mathbf{M}]$.

6.3.3.4 Block-diagonal form

In all this section we still suppose that all the orbits of the elements of $\mathcal{X}$ and $\mathcal{Y}$ are regular.

Index sets. According to the decomposition in Eq. 6.6, for any $\rho \in Irrep(\mathcal{G})$, there exist $d(\rho)$ elements $\eta \in Irrep(\mathcal{G})$ such that ρ and η are isomorphic and each of them is of course of degree $d(\rho)$. Hence, there are as many different projectors in Prop. 6.7 (i.e. rows in $\mathbb{E}$) as triplets in $\mathscr{B}$ defined by

$$\mathscr{B} := \bigcup_{\rho\in Irrep(\mathcal{G})} \bigcup_{(p,i)\in[\![1,d(\rho)]\!]^2} \{(\rho,p,i)\}.$$

This set has the same cardinal as $\mathcal{G}$ (as a consequence of Thm. 6.3.2). The main point is that the rows of $\mathbb{E}$ are each associated to a unique $b \in \mathscr{B}$, so that we have

$$\mathbb{E} : \mathbb{C}[\mathcal{G}] \to \mathbb{C}[\mathscr{B}].$$

Hence, thanks to Eq. 6.16, $\mathbb{E}_{\mathcal{Z}}$ can be interpreted as a mapping from $\mathbb{C}[\mathcal{Z}\times\mathcal{G}]$ to $\mathbb{C}[\mathcal{Z}\times\mathscr{B}]$. In other words, $D[\mathbf{M}]$ can itself be seen as a mapping from $\mathbb{C}[\mathcal{Y}\times\mathscr{B}]$ to $\mathbb{C}[\mathcal{X}\times\mathscr{B}]$, where $\mathcal{X}$ and $\mathcal{Y}$ are the target and source sets of germs respectively. We can thus identify the entries of $D[\mathbf{M}]$ through the pairs $(\mathbf{x},b) \in \mathcal{X}\times\mathscr{B}$ and $(\mathbf{y},c) \in \mathcal{Y}\times\mathscr{B}$.

Non-zero entries of $D[\mathbf{M}]$. Now, let $D[\mathbf{M}]_{b,c}$ be the block (b,c) of $D[\mathbf{M}]$ $(c,b \in \mathscr{B})$ defined by

$$(D[\mathbf{M}]_{b,c})_{\mathbf{x},\mathbf{y}} := (D[\mathbf{M}])_{(\mathbf{x},b),(\mathbf{y},c)}\,, \tag{6.20}$$

where $D[\mathbf{M}]_{b,c} \in \mathbb{C}^{\#\mathcal{X}\times\#\mathcal{Y}}$. Let $b=(\rho_l,p,i), c=(\rho_k,q,j) \in \mathscr{B}$, $\mathbf{x}\in\mathcal{X}$, $\mathbf{y}\in\mathcal{Y}$. We have, following [12]:

$$\begin{aligned}
(D[\mathbf{M}]_{b,c})_{\mathbf{x},\mathbf{y}} &= \frac{\sqrt{d(\rho_l)d(\rho_k)}}{|G|}\sum_{g\in G}\overline{\omega_{p,i}^{(l)}(g)}\sum_{g'\in G}\omega_{q,j}^{(k)}(gg')\mathfrak{I}(g\cdot\mathbf{x},(gg')\cdot\mathbf{y})\\
(\mathfrak{I}\ \textit{is}\ \mathcal{G}\textit{-invariant}) &= \frac{\sqrt{d(\rho_l)d(\rho_k)}}{|G|}\sum_{g\in G}\overline{\omega_{p,i}^{(l)}(g)}\sum_{g'\in G}\omega_{q,j}^{(k)}(gg')\mathfrak{I}(\mathbf{x},g'\cdot\mathbf{y})\\
&= \frac{\sqrt{d(\rho_l)d(\rho_k)}}{|G|}\sum_{g\in G}\overline{\omega_{p,i}^{(l)}(g)}\sum_{g'\in G}\underbrace{\Big[\sum_{t=0}^{d(\rho_k)-1}\omega_{q,t}^{(k)}(g)\omega_{t,j}^{(k)}(g')\Big]}_{\omega^{(k)}(gg')=\omega^{(k)}(g)\omega^{(k)}(g')}\mathfrak{I}(\mathbf{x},g'\cdot\mathbf{y})\\
&= \frac{\sqrt{d(\rho_l)d(\rho_k)}}{|G|}\sum_{g'\in G}\sum_{t=0}^{d(\rho_k)-1}\omega_{t,j}^{(k)}(g')\underbrace{\Big[\sum_{g\in G}\overline{\omega_{p,i}^{(l)}(g)}\omega_{q,t}^{(k)}(g)\Big]}_{=\begin{cases}|\mathcal{G}|/d(\rho_l) & \textit{if } k=l,p=q,i=t\\ 0 & \textit{otherwise}\end{cases}}\mathfrak{I}(\mathbf{x},g'\cdot\mathbf{y})\\
&= \begin{cases}\displaystyle\sum_{g\in G}\omega_{i,j}^{(l)}(g)\mathfrak{I}(\mathbf{x},g\cdot\mathbf{y}) & \textit{if } k=l,p=q\\ 0 & \textit{otherwise}\end{cases}.
\end{aligned} \tag{6.21}$$

Remark 6.3.2. *The last line of Eq. 6.21 does not depend on i (nor j). This means that the $d(\rho_l)$ blocks of the block-diagonalization corresponding to $\tilde{V}_l$ coincide (see [12]) for any $l\in[\![1,m]\!]$.*

To be more precise, if $b=(\rho_l,p,i), c=(\rho_k,q,j)\in\mathscr{B}$, then $D[\mathbf{M}]_{b,c}=0$ if $l\neq k$ or $p\neq q$. The final non-zero blocks are of size $(d(\rho_l)\#\mathcal{X})\times(d(\rho_l)\#\mathcal{Y})$ for each $\rho_l\in Irrep(\mathcal{G})$.

Theorem 6.3.3. *Let $\mathcal{X},\mathcal{Y}$ be finite sets of points with regular $\mathcal{G}$-orbits. Let $\mathbf{M}$ a $\mathcal{G}$-invariant matrix as in Eq. 6.18. We have*

$$\boxed{\mathbf{M}=\mathbb{E}_{\mathcal{X}}\,D[\mathbf{M}]\,\mathbb{E}_{\mathcal{Y}}^{*}}$$

where for any $b=(\rho_l,p,i), c=(\rho_k,q,j)\in\mathscr{B}$, $D[\mathbf{M}]_{b,c}=0$ if $l\neq k$ or $p\neq q$.

However, when $D[\mathbf{M}]$ is seen as a mapping from $\mathbb{C}[\mathcal{X}\times\mathscr{B}]$ to $\mathbb{C}[\mathcal{Y}\times\mathscr{B}]$, the entries of $D[\mathbf{M}]_{b,c}$ are not contiguous in $D[\mathbf{M}]$. We thus need additional permutations to recover a real block-diagonal structure.

Permutations. To represent $D[\mathbf{M}]$ as a block-diagonal matrix, one has to transform it into a mapping from $\mathbb{C}[\mathscr{B}\times\mathcal{X}]$ to $\mathbb{C}[\mathscr{B}\times\mathcal{Y}]$ by means of permutions. Indeed, the row and column entries of $D[\mathbf{M}]$ are ordered in the following way, using $\mathcal{Z}=\{\mathbf{z}_1,...,\mathbf{z}_{\#\mathcal{Z}}\}$ and $\mathscr{B}=\{b_1,...,b_{\#\mathscr{B}}\}$,

$$(\mathbf{z}_1,b_1),...,(\mathbf{z}_1,b_{\#\mathscr{B}}),(\mathbf{z}_2,b_1),...,(\mathbf{z}_{\#\mathcal{Z}},b_{\#\mathscr{B}})$$

but we want them to be indexed as in the following sequence

$$(b_1,\mathbf{z}_1),...,(b_1,\mathbf{z}_{\#\mathcal{Z}}),(b_2,\mathbf{z}_1),...,(b_{\#\mathscr{B}},\mathbf{z}_{\#\mathcal{Z}})$$

for $\mathcal{Z} = \mathcal{X}$ (rows) and $\mathcal{Z} = \mathcal{Y}$ (columns). Hence, introducing $P_{\mathcal{Z}} : \mathbb{C}[\mathcal{Z} \times \mathscr{B}] \to \mathbb{C}[\mathscr{B} \times \mathcal{Z}]$ such that for any $(b, \mathbf{z}) \in \mathscr{B} \times \mathcal{Z}$, $(b', \mathbf{z}') \in \mathcal{Z} \times \mathscr{B}$

$$(P_{\mathcal{Z}})_{(b,\mathbf{z}),(\mathbf{z}',b')} := \begin{cases} 1 & \text{if } \mathbf{z} = \mathbf{z}' \text{ and } b = b' \\ 0 & \text{otherwise} \end{cases},$$

we obtain the permutations we want.

Final result. We finally obtained the block-diagonalization we were looking at and which is summarized in Thm. 6.3.4, which specifies the result of Thm. 6.3.3.

Theorem 6.3.4. *Let $\mathcal{X}, \mathcal{Y}$ be finite sets of points with regular $\mathcal{G}$-orbits. Let $\mathbf{M}$ a $\mathcal{G}$-invariant matrix as in Eq. 6.18. We have*

$$\boxed{\mathbf{M} = \mathbb{E}_{\mathcal{X}}\, P_{\mathcal{X}}\, \bar{D}[\mathbf{M}]\, P_{\mathcal{Y}}^T\, \mathbb{E}_{\mathcal{Y}}^*}$$

where $\mathbb{E}_{\mathcal{X}}$, $\mathbb{E}_{\mathcal{Y}}^*$ and $\bar{D}[\mathbf{M}] := P_{\mathcal{X}}^T\, D[\mathbf{M}]\, P_{\mathcal{Y}}$ are block-diagonal.

Remark 6.3.3. *The result of Thm. 6.3.4 is a generalization of Thm. 6.2.1, corresponding to the block-diagonalization for the product rules using abelian cyclic groups.*

The assumption on the regular orbits in Thm. 6.3.4 can actually be relaxed. We discuss various techniques to do so in Sect. 6.3.4.

6.3.4 Singular orbits

We are now able to express the representations exploited in Eq. 6.12 in terms of matrices when all the orbits are regular. To adapt the formula in Eq. 6.16 to the possible singular orbits of a general set $\mathcal{Z}$ on which $\mathcal{G}$ acts, it is useful to introduce the stabilizer of a group.

Definition 6.3.7. *Let $\mathcal{Z}$ be a finite set and $\mathcal{G}$ be a group acting on $\mathcal{Z}$. The stabilizer $Stab(\mathbf{z})$ of $\mathbf{z} \in \mathcal{Z}$ is the set $Stab(\mathbf{z}) := \{g \mid g \cdot \mathbf{z} = \mathbf{z}, g \in \mathcal{G}\} \subseteq \mathcal{G}$.*

For any $\mathbf{z} \in \mathcal{Z}$, if $Stab(\mathbf{z})$ is normal[4], the set $\mathcal{H}(\mathbf{z}) := \frac{\mathcal{G}}{Stab(\mathbf{z})}$ is such that $\mathcal{H}(\mathbf{z}) \leq \mathcal{G}$. Assuming that $Stab(\mathbf{z})$ is normal for any $\mathbf{z}$, we have

$$\begin{aligned} \mathcal{G} \cdot \mathcal{Z} &= \bigcup_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z} \rangle_{\mathcal{G}} \\ &= \bigcup_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z} \rangle_{\mathcal{H}(\mathbf{z})} \end{aligned}$$

and the space $\mathbb{C}[\mathcal{G} \cdot \mathcal{Z}]$ is replaced in the general case by $\mathbb{C}\left[\bigcup_{\mathbf{z} \in \mathcal{Z}} \langle \mathbf{z} \rangle_{\mathcal{H}(\mathbf{z})}\right]$. Hence, the representations $\mu_{\mathbf{z}}$ (see Eq. 6.15) are defined on $\mathcal{H}(\mathbf{z})$ only instead of $\mathcal{G}$, i.e. $\forall g \in \mathcal{H}(\mathbf{z})$, $\mu_{\mathbf{z}}(g) \in Aut\left(\mathbb{C}[\langle \mathbf{z} \rangle_{\mathcal{H}(\mathbf{z})}]\right)$. With these modifications, the formula in Eq. 6.16 still holds in the case of possibly singular orbits.

The block sizes in the full block-diagonalization do not only depend on the group cardinal. Both the degree of the irreducible representations (that may be greater than one in the general case) and the singular orbits affect these sizes. There are two methods, as already explored in Sects. 6.2.2.4 and 6.2.2.5 in the abelian case, for the treatment of the singular orbits. The first approach consists in completing a singular orbit into a regular one by means of zero-padding and leads to a matrix embedding that can be block-diagonalized with Thm. 6.3.4. This is presented in Sect. 6.3.4.1. The second method takes into account the zero entries appearing when the padding method is used to reduce the size of the diagonal blocks and can be associated to optimal projectors in terms of size, as explained in Sect. 6.3.4.2.

[4] $Stab(\mathbf{z})$ is always a group and is normal if and only if $\forall s \in Stab(\mathbf{z}), \forall g \in \mathcal{G}, gsg^{-1} \in Stab(\mathbf{z})$.

6.3.4.1 Padding method

As for the abelian case, a naive way of taking into account the singular orbits is the padding method. Let $\mathbf{z}_0, \mathbf{z}_1 \in \mathcal{Z}$ with possibly singular orbits. For each of these points we can identify a set $\mathcal{H}(\mathbf{z}_i) =: \{h_1^i, ..., h_{\#\mathcal{H}(\mathbf{z}_i)}^i\}$, $i \in \{0,1\}$. Hence, the matrix block $\mathbf{M}$ such that, for a $\mathcal{G}$-invariant operator $\mathfrak{I}$ and $(k,l) \in [\![1, \#\mathcal{H}(\mathbf{z}_0)]\!] \times [\![1, \#\mathcal{H}(\mathbf{z}_1)]\!]$,

$$\mathbf{M}_{k,l} := \mathfrak{I}\left(h_k^0 \cdot \mathbf{z}_0, h_l^1 \cdot \mathbf{z}_1\right)$$

can be embedded into a larger matrix $\tilde{\mathbf{M}}$ (named a *group embedding* of $\mathbf{M}$) such that

$$\tilde{\mathbf{M}}_{i,j} := \mathfrak{I}\left(g_i \cdot \mathbf{z}_0, g_j \cdot \mathbf{z}_1\right) \tag{6.22}$$

where $\mathcal{G} = \{g_1, ..., g_{|\mathcal{G}|}\}$ and $(i,j) \in [\![1, |\mathcal{G}|]\!]^2$. This can be done by introducing the two matrices $I_{left} : \mathbb{C}[\mathcal{H}(\mathbf{z}_0)] \to \mathbb{C}[\mathcal{G}]$ and $I_{right} : \mathbb{C}[\mathcal{G}] \to \mathbb{C}[\mathcal{H}(\mathbf{z}_1)]$ such that

$$\left(I_{left}\right)_{k,i} := \begin{cases} 1 & \text{if } h_k^0 = g_i \\ 0 & \text{otherwise} \end{cases},$$

$$\left(I_{right}\right)_{j,l} := \begin{cases} 1 & \text{if } h_l^1 = g_i \\ 0 & \text{otherwise} \end{cases}.$$

Indeed, we obtain

$$\mathbf{M} = I_{left}\ \tilde{\mathbf{M}}\ I_{right}.$$

The difference between $\mathbf{M}$ and $\tilde{\mathbf{M}}$ is that the second can be considered as a mapping between two regular orbits. Hence, the padding method increases the size of the matrix $\mathbf{M}$ in order to recover only regular orbits, that can all be treated using the same restriction matrix. The extension of the padding method to any number of germs is straightforward. We have

$$\mathbb{C}[\mathcal{G} \cdot \mathcal{Z}] \equiv \bigoplus_{\mathbf{z} \in \mathcal{Z}} \mathbb{C}[\langle \mathbf{z} \rangle_{\mathcal{H}(\mathbf{z})}]. \tag{6.23}$$

Remark 6.3.4. *$\mathcal{H}(\mathbf{z})$ may not be a group in the general case (i.e. when Stab$(\mathbf{z})$ is not normal) so the notation $\langle \mathbf{z} \rangle_{\mathcal{H}(\mathbf{z})}$ is not rigourous. This notation actually refers to $\langle \mathbf{z} \rangle_{\mathcal{G}}$, but insisting on the cardinal deficiency compared to regular orbits. In the general case, when Stab$(\mathbf{z})$ is not a normal subgroup of $\mathcal{G}$, the relevant tools to work with are actually the cosets (see [189] Sect. 3.3)*

Let $I_{\mathcal{H}(\mathbf{z}),\mathcal{G}} \in \{0,1\}^{|\mathcal{G}| \times \#\mathcal{H}(\mathbf{z})}$ be the mapping from $\mathbb{C}[\mathcal{H}(\mathbf{z})]$ in $\mathbb{C}[\mathcal{G}]$ defined by

$$\left(I_{\mathcal{H}(\mathbf{z}),\mathcal{G}}\right)_{h,g} := \begin{cases} 1 & \text{if } h = g \\ 0 & \text{otherwise} \end{cases}.$$

Hence, $I_{\mathcal{H}(\mathbf{z}),\mathcal{G}}$ can be seen as a mapping from $\mathbb{C}[\langle \mathbf{z} \rangle_{\mathcal{H}(\mathbf{z})}] \equiv \mathbb{C}[\mathcal{H}(\mathbf{z}) \cdot \mathbf{z}] \equiv \mathbb{C}[\mathbf{z} \times \mathcal{H}(\mathbf{z})]$ to $\mathbb{C}[\mathcal{G}]$. Doing so for all the orbits, the decomposition in Eq. 6.23 allows to construct the padding matrix for the entire point cloud $\mathcal{G} \cdot \mathcal{Z}$. This padding matrix $I_{\mathcal{G},\mathcal{Z}} : \bigoplus_{\mathbf{z} \in \mathcal{Z}} \mathbb{C}[\mathbf{z} \times \mathcal{H}(\mathbf{z})] \to \mathbb{C}[\mathcal{Z} \times \mathcal{G}]$ for a general point cloud $\mathcal{Z}$ can be written as

$$\left(I_{\mathcal{G},\mathcal{Z}}\right)_{(\mathbf{z}',g),(\mathbf{z},h)} := \begin{cases} 1 & \text{if } h = g, \mathbf{z} = \mathbf{z}' \\ 0 & \text{otherwise} \end{cases}$$

for any $(\mathbf{z}', g) \in \mathcal{Z} \times \mathcal{G}$ and $\mathbf{z} \in \mathcal{Z}$, $h \in \mathcal{H}(\mathbf{z})$. The content of Thm. 6.3.4 can be adapted to the general case with singular orbits.

Theorem 6.3.5. *Let $\mathcal{X}, \mathcal{Y}$ be finite point clouds. Let $\mathbf{M}$ a $\mathcal{G}$-invariant matrix as in Eq. 6.18. We have*

$$\boxed{\mathbf{M} = I_{\mathcal{G},\mathcal{X}}^*\ \mathbb{E}_{\mathcal{X}}\ P_{\mathcal{X}}\ \bar{D}[\tilde{\mathbf{M}}]\ P_{\mathcal{Y}}^T\ \mathbb{E}_{\mathcal{Y}}^*\ I_{\mathcal{G},\mathcal{Y}}}$$

where $\bar{D}[\tilde{\mathbf{M}}]$ is block-diagonal and $\tilde{\mathbf{M}}$ is defined as in Eq. 6.22.

We refer to this treatment of the singular orbits as the *padding* method.

6.3.4.2 Impact of padding on diagonal blocks and minimal projection method

In practice, a certain amount of entries corresponding to singular orbits in the diagonal blocks of the block-diagonalization of the group embedding $\tilde{\mathbf{M}}$ of $\mathbf{M}$ (presented in Sect. 6.3.4.1) may vanish. This is explained by the following computation. For the sake of simplicity, we assume that $Stab(\mathbf{y})$ is normal in $\mathcal{G}$. Therefore using $b = (\rho_l, p, i), c = (\rho_l, q, j) \in \mathscr{B}$, we have

$$\begin{aligned}
\left(D[\tilde{\mathbf{M}}]_{b,c}\right)_{\mathbf{x},\mathbf{y}} &= \sum_{g\in\mathcal{G}} \omega_{i,j}^{(l)}(g)\mathfrak{I}(\mathbf{x}, g\cdot\mathbf{y}) \\
(g = hs) &= \sum_{s\in Stab(\mathbf{y})}\sum_{h\in\mathcal{H}(\mathbf{y})} \omega_{i,j}^{(l)}(hs)\mathfrak{I}(\mathbf{x}, \underbrace{(hs)\cdot\mathbf{y}}_{=h\cdot\mathbf{y}}) \\
&= \sum_{s\in Stab(\mathbf{y})}\sum_{h\in\mathcal{H}(\mathbf{y})} \Big[\sum_{t=1}^{d(\rho_l)} \omega_{i,t}^{(l)}(h)\omega_{t,j}^{(l)}(s)\Big]\mathfrak{I}(\mathbf{x}, h\cdot\mathbf{y}) \\
&= \sum_{t=1}^{d(\rho_l)} \left(\sum_{s\in Stab(\mathbf{y})} \omega_{t,j}^{(l)}(s)\right) \sum_{h\in\mathcal{H}(\mathbf{y})} \omega_{i,t}^{(l)}(h)\mathfrak{I}(\mathbf{x}, h\cdot\mathbf{y}).
\end{aligned}$$

If $\rho_{l|Stab(\mathbf{y})} \in Irrep(Stab(\mathbf{y}))$, with $\rho_{l|Stab(\mathbf{y})}$ the restriction of ρ_l to $Stab(\mathbf{y})$, then the term $\sum_{s\in Stab(\mathbf{y})} \omega_{t,j}^{(l)}(s)$ vanishes unless ρ_l is trivial (as a consequence of the Schur's lemma). However, in the general case, $\rho_l \notin Irrep(Stab(\mathbf{y}))$ (the Mackey's criterion allows to decide if the irreducibility of the restricted representation holds, see [189] Sect. 7.4 Prop. 23). We thus do not have a conclusion in the general case. Nevertheless, one can still obtain smaller projectors for singular orbits by summing along the columns, according to the stabilizers, of the projectors of the full group. This is used in [12] to derive new restriction matrices by applying a row-orthonormalization on extractions of the resulting (rectangular) matrix. This is referred to as the *minimal projection* method.

Compared to the padding method, the minimal projection one is more tricky to implement in the general case: one has to identify on which $\mathcal{G}$-invariant subspaces the new restriction matrices project. The block sizes in the non-zero blocks of the block-diagonalization of $\mathbf{M}$ directly depend on these chosen $\mathcal{G}$-invariant subspaces.

6.4 Lebedev rules in *hf-fmm*

We want to apply the block-diagonalization we presented in Thm. 6.3.5 to the existing quasi-optimal cubature rules in the context of *hf-fmm*. As we described in Sect. 6.1, these rules are built with a $\mathcal{G}$-invariance where $\mathcal{G}$ is a rotation group that preserves one of the Platonic solids. None of these groups is abelian, meaning that the projectors are not Fourier matrices anymore for these quasi-optimal cubature rules (so the approach of Sect. 6.2 is not sufficient to handle them and we need to rely on Sect. 6.3). A direct consequence is that the Fast Fourier Transforms exploited with the product rules cannot be used. In other words, the projection step of the block-diagonalization when dealing with quasi-optimal rules cannot be as efficient as in the case of the abelian cyclic group, i.e. of the product rules. However, since less cubature points are involved in the quasi-optimal rules than in the product ones, the projection matrices are smaller. A similar discussion has to be done on the efficiency of the block-diagonalization compared to the optimal case of an abelian cyclic group. In Sect. 6.4.1 we present the (quasi-optimal) Lebedev rules and numerical results on the interpolation on the sphere using our group theory based explicit block-diagonalization combined with such rules. In Sect. 6.4.2, we explain how the particularity in terms of group invariance of the Lebedev rules are important in a FMM context. Then, based on the asymptotical behavior of the different exact fast algorithms for the interpolation on the sphere, we discuss in Sect. 6.4.3 a switching between the Lebedev rules and the product ones *during the same FMM application*. A

Order	3	5	7	9	11	13	15	17	19	21	23	25	27	29
Number of nodes	6	14	26	38	50	74	86	110	146	170	194	230	266	302

Order	31	35	41	47	53	59	65	71	77	83	89
Number of nodes	350	434	590	770	974	1202	1454	1730	2030	2354	2702

Order	95	101	107	113	119	125	131
Number of nodes	3074	3470	3890	4334	4802	5294	5810

Table 6.1: Number of nodes in Lebedev cubature grids for the integration orders these grids are tabulated for.

general discussion on the applicability of the Lebedev rules combined with block-diagonalizations of the interpolation matrices on the sphere is done in Sect. 6.4.4.

6.4.1 Invariant rules in *hf-fmm*

In this section, we quickly motivate the use of tabulated Lebedev rules in the *hf-fmm* context (Sect. 6.4.1.1). We then provide an overview of the irreducible representations of the invariance group of the grids in such rules (Sect. 6.4.1.2). Implementation details on the way of evaluating the projectors are provided in Sect. 6.4.1.4, motivating the chosen treatment for the singular orbits. Finally, in Sect. 6.4.1.5 are presented numerical experiments to illustrate the effect of our block-diagonalization on the interpolation over the sphere.

6.4.1.1 Choice of the cubature rule

The first problem we face up when trying to use the quasi-optimal cubature rules is related to their computation. Such rules have to be tabulated to be used in practical applications (such as in *hf-fmm*). A construction method for the invariant rules under the action of the invariant group of the icosahedron has been provided in [10]. This corresponds to the same invariance group than the cubature proposed in [171] which has an efficiency greater than 1. In addition, the underlying group, the dodecahedral one (see Tab. 6.1) has the greatest cardinal among the invariance groups of the Platonic solids[5]. However, there is a natural symmetry in the octrees under the action of the octahedral group (see Sect. 5.3.3). This corresponds to the invariance group of the cube (or equivalently the octahedron). There exist invariant rules under such group, named after V. I. Lebedev who proposed various tabulations for some of them [153–158]. Not all the integration orders have lead to effective computations, but these rules are tabulated up to the 131^{th} order with a large amount of intermediate values. See Tab. 6.1 for the number of nodes in each of the tabulated Lebedev rules. In this thesis, we do not explore the numerical computation of group invariant cubature rules, so we restrict ourselves to the existing and tabulated Lebedev rules. Because they allow to exploit all the symmetries of the (*hf-fmm*) FMM operators while conserving a quasi-optimality in the **M2L** application cost, we decided to apply our block-diagonalization to the **M2M**/**L2L** matrices generated by these grids. The cardinal of $\mathfrak{D}_3$, the invariance group of the cube, is equal to 48 (using $\mathfrak{D}_3 \equiv \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^3 \times \mathfrak{S}_3$). Since $\mathfrak{D}_3$ is not abelian, we have to provide a matrix form of the irreducible representations of this group in well-chosen bases (the projectors of Prop. 6.7 relying on a basis choice).

[5] This group also has the largest irreducible representations in terms of degree (up to 5) among the invariance groups of the Platonic solids. This implies that its cardinal is not sufficient to compare the efficiency of the block-diagonalization with the other groups.

6.4.1.2 Irreducible representations of $\mathfrak{D}_3$

The irreducible representations of the invariance groups of the Platonic solids are well known in the group representation theory. See [191] for the details on the representations of $\mathfrak{D}_3$. There exist ten inequivalent irreducible representations of $\mathfrak{D}_3$: four degree one representations $\omega^{(1)},...,\omega^{(4)}$, two degree two representations $\omega^{(5)},\omega^{(6)}$ and four degree three representations $\omega^{(7)},...,\omega^{(10)}$. This means that there is a total of $4+2\times 2+3\times 4=20$ elementary restriction matrices (as defined in Sect. 6.3.3). In Fig. 6.8, we represent the shape of the block-diagonalization obtained in the single orbit case with regular orbits and using the irreducible representations of $\mathfrak{D}_3$.

Degree one representations. The first irreducible representation of $\mathfrak{D}_3$ is given by the *trivial representation* $\omega^{(1)}$ which has a degree equal to one. It is defined as

$$\omega^{(1)}(g) := 1$$

for any $g \in \mathfrak{D}_3$. The corresponding elementary restriction matrix is

$$E^{(1)} := \frac{1}{\sqrt{48}}(1,...,1) \in \mathbb{C}^{1\times 48}$$

because $\left|\mathfrak{D}_3\right| = 48$. The three other degree one irreducible representations of $\mathfrak{D}_3$ can be found for instance in [11, 191].

Degree two representations. The degree two irreducible representations of $\mathfrak{D}_3$ are the isometries of $SO(2)$ that preserve the square. There are 8 such matrices that are

$$\begin{bmatrix}1&0\\0&1\end{bmatrix},\begin{bmatrix}-1&0\\0&1\end{bmatrix},\begin{bmatrix}1&0\\0&-1\end{bmatrix},\begin{bmatrix}-1&0\\0&-1\end{bmatrix},\begin{bmatrix}0&1\\1&0\end{bmatrix},\begin{bmatrix}0&-1\\1&0\end{bmatrix},\begin{bmatrix}0&1\\-1&0\end{bmatrix},\begin{bmatrix}0&-1\\-1&0\end{bmatrix}. \tag{6.24}$$

The corresponding restriction matrices are built using the entries of the matrices in Eq. 6.24 and can be explicitly found for instance in [11].

Degree three representations. The irreducible representations of degree 3 of $\mathfrak{D}_3$ are the simplest to understand and correspond to the rotations of $SO(3)$ preserving the cube. Thanks to the decomposition $\mathfrak{D}_3 \equiv \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^3 \times \mathfrak{S}_3$, these rotation matrices are obtained by combining the following matrices (corresponding to $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^3$)

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix},\begin{bmatrix}-1&0&0\\0&1&0\\0&0&1\end{bmatrix},\begin{bmatrix}1&0&0\\0&-1&0\\0&0&1\end{bmatrix},\begin{bmatrix}1&0&0\\0&1&0\\0&0&-1\end{bmatrix},$$

$$\begin{bmatrix}-1&0&0\\0&-1&0\\0&0&1\end{bmatrix},\begin{bmatrix}1&0&0\\0&-1&0\\0&0&-1\end{bmatrix},\begin{bmatrix}-1&0&0\\0&1&0\\0&0&-1\end{bmatrix},\begin{bmatrix}-1&0&0\\0&-1&0\\0&0&-1\end{bmatrix}$$

with these matrices (corresponding to $\mathfrak{S}_3$)

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix},\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix},\begin{bmatrix}0&0&1\\0&1&0\\1&0&0\end{bmatrix},\begin{bmatrix}1&0&0\\0&0&1\\0&1&0\end{bmatrix},\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix},\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix}.$$

See [11] for the explicit form of the restriction matrices for the degree three irreducible representations of $\mathfrak{D}_3$.

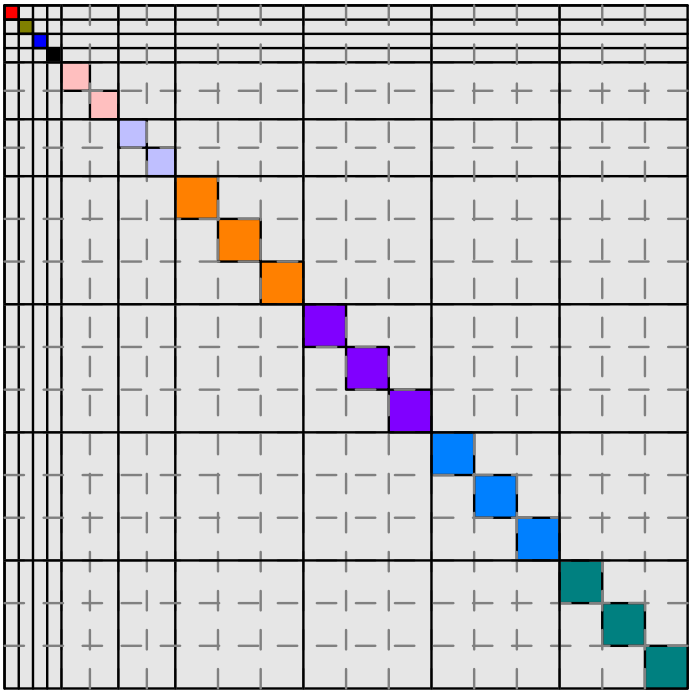


Figure 6.8: Shape of $D[\mathbf{M}]$ in the single orbit case using the representations of $\mathfrak{D}_3$. Different colors indicate that the blocks have different values. Thanks to Rem. 6.3.2, the $d(\rho_l)$ isomorphic representations to ρ_l lead to equal blocks. Continuous black lines represent delimitations between isotypic components (first line of Eq. 6.6) and dashed lines represent delimitations between isomorphic irreducible representations in a given isotypic component.

6.4.1.3 Singular orbits of $\mathfrak{D}_3$

In order to classify the elements of the cubature grids according to their stabilizers, we need to be able to determine the singular $\mathfrak{D}_3$-orbits on the sphere. Our choice of fundamental domain is the set $\{(z_1, z_2, z_3) \in \mathbb{S}^2 \,|\, z_1 \geq z_2 \geq z_3\}$. The germs $\mathbf{z} = (z_1, z_2, z_3) \in \mathbb{S}^2$ that lead to a singular $\mathfrak{D}_3$-orbit are those verifying at least one of the following conditions:

1. $\exists k \neq l$ such that $z_k = z_l$;
2. $\exists k$ such that $z_k = 0$.

The first point corresponds to a singular orbit with regard to a subgroup of $\mathfrak{S}_3$ using $\mathfrak{D}_3 \equiv \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^3 \times \mathfrak{S}_3$ and the second point to a singular orbit with regard to (at least) one $\frac{\mathbb{Z}}{2\mathbb{Z}}$. The combination of the two types of singular orbits lead to another type of singular orbits. There are only two germs on the sphere (up to group action, i.e. for a fixed fundamental domain) in the intersection of the two conditions:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{pmatrix}.$$

The first of these germs has a $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^3$-orbit and the second one has a stabilizer isomorphic to $\mathfrak{S}^2 \times \frac{\mathbb{Z}}{2\mathbb{Z}}$.

6.4.1.4 Implementation details

We decided to consider the padding method (presented in Sect. 6.3.4.1) instead of the minimal projection one (see Sect. 6.3.4.2) because this first method is better suited to low-level optimizations.

First, we recall that the factorization of the interpolation matrix on the sphere $\mathbf{M}$ in Thm. 6.3.5 writes

$$\mathbf{M} = I^*_{\mathfrak{D}_3,\mathcal{X}}\ \mathbb{E}_{\mathcal{X}}\ P_{\mathcal{X}}\ \bar{D}[\tilde{\mathbf{M}}]\ P^T_{\mathcal{Y}}\ \mathbb{E}^*_{\mathcal{Y}}\ I_{\mathfrak{D}_3,\mathcal{Y}}.$$

$\mathbb{E}_{\mathcal{Z}}$ is a block-diagonal matrix with a diagonal filled by the same single block (that is $\mathbb{E}$). Numerically, a product by $\mathbb{E}_{\mathcal{Z}}$ with a vector v can be seen as a sequence of products by $\mathbb{E}$ on restrictions of the input vector v. Splitting v into small parts of consecutive entries and of size $|\mathfrak{D}_3| = 48$ (that is the number of columns/rows of $\mathbb{E}$), this matrix-vector product can be written as a matrix-matrix one, that benefits from highly efficient implementations. This technique is refered to as *vector stacking* and is presented in App. D.2 in the FMM context. The same discussion is true for the conjugate of $\mathbb{E}_{\mathcal{Z}}$ because of its block-diagonal structure. The matrices $\mathbb{E}_{\mathcal{X}}$ and $\mathbb{E}_{\mathcal{Y}}$ in the factorization given by Thm. 6.3.5 are evaluated this way in our implementation (i.e. using matrix-matrix product reformulations).

The problem we face up is that the matrix $\bar{D}[\tilde{\mathbf{M}}]$ is also block-diagonal but not in the same basis than $\mathbb{E}_{\mathcal{X}}$, $\mathbb{E}_{\mathcal{Y}}$. Since the amount of non-zero entries of $\bar{D}[\tilde{\mathbf{M}}]$ is too high, the use of a sparse format to store $\bar{D}[\tilde{\mathbf{M}}]$ would be inefficient (both in terms of memory footprint and application timings). We thus apply the permutations $P_{\mathcal{X}}$ and $P^T_{\mathcal{Y}}$ (to switch between the bases on which the $\mathbb{E}_{\mathcal{Z}}$, $\mathcal{Z} = \mathcal{X}, \mathcal{Y}$ are block-diagonal to the basis on which $\bar{D}[\tilde{\mathbf{M}}]$ is) *on-the-fly* (with a linear cost). Each non-zero block of a block-diagonal matrix is stored and applied as a dense matrix. Hence, the product by $\bar{D}[\tilde{\mathbf{M}}]$ is divided into a sequence of dense matrix-vector products (i.e. with the diagonal blocks of $\bar{D}[\tilde{\mathbf{M}}]$).

6.4.1.5 Numerical results

In Fig. 6.9 we present application timings for various interpolations on the sphere using Lebedev rules with and without block-diagonalization. We also provide timings using the FFT-based technique of Jakob-Chien-Alpert (see Sect. 3.2.2.3) with Gauss-Legendre grids (without 1D FMM). Since not all the orders are available for the tabulated Lebedev rules, we choose the order of the target rule to be the smallest one greater than twice the one of the source rule in the available orders in Tab. 6.1. For the Gauss-Legendre rules, the order of the target rule is equal to twice the one of the source rule. Only the order of the source grid is given (in the abscissa). We emphasize that the matrices obtained using the Lebedev rules are smaller than the one obtained with Gauss-Legendre rules when no optimization is used, but the second matrices benefit from the FFT applications and from a more efficient block-diagonalization (using the product structure), which results in a complexity reduction. The block-diagonalization on the Lebedev rules only reduces the *constant* in the evaluation cost. We did not implemented our block-diagonalization in a complete *hf-fmm* code. Hence, the results provided in Sect. 6.9 correspond to sequential timings on the isolated interpolation step on the sphere. The numerical results we present are obtained on a Intel Xeon Gold 6152 CPU. We used FFTW3 for the FFT applications and the MKL for the matrix-vector and matrix-matrix products.

The results obtained with the block-diagonalization of the interpolation matrices using Lebedev rules have a comparable application cost than the FFT-based techniques on Gauss-Legendre grids for source grid orders lower or equal to $L \approx 30$ (and target one $L \approx 60$). For higher orders, the block-diagonalization method on Lebedev rules becomes clearly less efficient than the FFT-based techniques. We are limited (in terms of interpolation orders) to the results provided in Fig. 6.9 with the Lebedev rules, which is not the case for the Gauss-Legendre ones. However, since our block-diagonalization method on Lebedev rules does not reduce the complexity of the interpolation over the sphere (as opposed to the Jakob-Chien-Alpert's method), this tendency can be extrapolated to even higher orders than those presented.

Notice that these are not negative results. Indeed, an important remark concerns the costs presented in Fig. 6.9 compared to the overall cost of *hf-fmm*. The block-diagonalization we presented in this chapter aims at reducing the timings for the interpolation step on the sphere using non-product grids needed in the **M2M**/**L2L** operators (which is the more expansive step in these

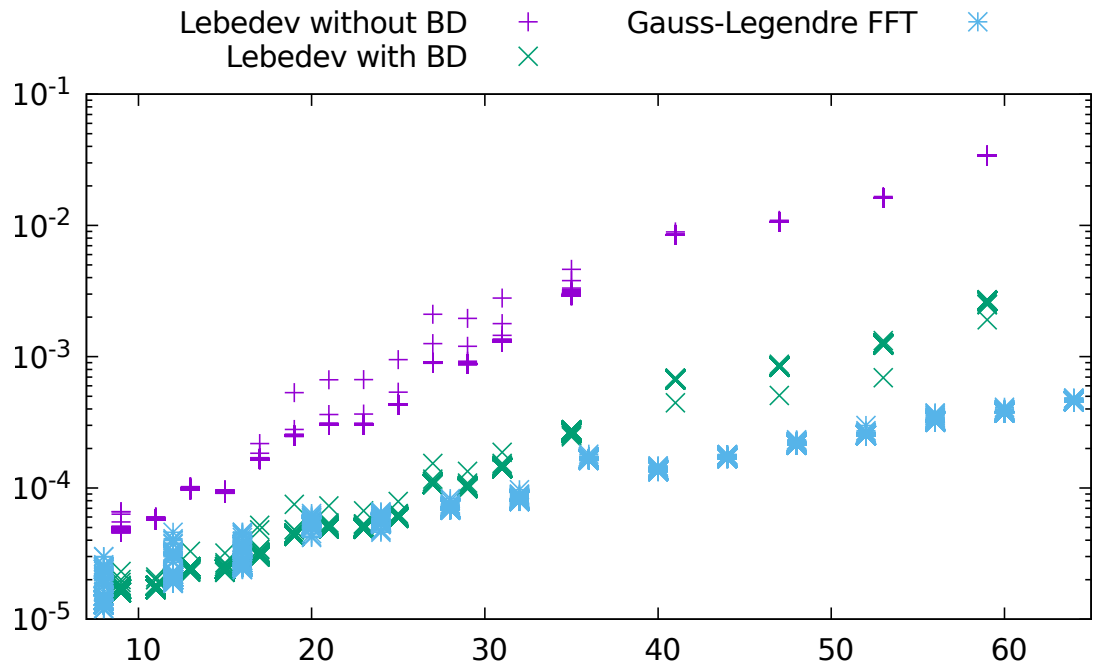


Figure 6.9: Sequential timings for the interpolation over the sphere using dense matrix-vector products and Lebedev rules (purple), block-diagonalization (BD) with Lebedev rules (green), indicative FFT techniques with Gauss-Legendre rules (blue). The abscissa refer to the integration order of the source cubature rule. 300 executions are performed for each tested order.

operators). However, the main motivation in the use of Lebedev rules is the quasi-optimality of such rules that trivially reduces the **M2L** operator cost. In addition, there are far more **M2L** applications than **M2M**/**L2L** ones in the deepest levels of the octrees (at least for most particle distributions), that is for the lowest interpolation orders. Hence, the overall performances of the Lebedev rules with block-diagonalization (**M2L** and **M2M**/**L2L** applications) in these deepest levels should be better than the performances obtained using Gauss-Legendre product rules.

Remark 6.4.1. *We are aware that our implementation is far from optimal. First, we considered linear algebra operations on complex data only, even if the interpolation operator on the sphere actually has a real range and the restriction matrices are composed of real entries. However, since they are applied to complex vectors, we decided to keep this choice of complex data. One can have written a slightly more complicated algorithm exploiting the real range of* $\mathfrak{I}_{\mathbb{S}^2}$ *while better benefiting from the BLAS 3 operations (see App. D.2) using a trick that will be presented in Sect. 7.5.4.3 but for another application. In addition, because a certain amount of diagonal blocks of* $D[\tilde{\mathbf{M}}]$ *are equal to others, with known positions (according to Rem. 6.3.2), other matrix-matrix reformulations of matrix-vector products could have been obtained (concatenating an amount of vectors equal to the degree of each irreducible representation).*

6.4.2 Full exploitation of $\mathfrak{D}_3$ symmetries in *hf-fmm*

As mentioned in Sect. 6.2.4, the use of product rules prevents the exploitation of the full symmetry group $\mathfrak{D}_3$ as in Sect. 5.3.3, that is from reducing the number of truly needed **M2L** matrices at each tree level. Notice that the computation cost of these matrices is quite important in *hf-fmm* compared to other FMM formulations, such as the 3D Laplace one (see Sect. 2.2.4.3), at least

for large cells in terms of wavelength. Hence, this optimization reduction has an impact on the precomputation cost of the method.

To apply the methodology described in Sect. 5.3.3, that is in order to ensure that only 16 **M2L** matrices need to be precomputed at each tree level, one still has to verify the group invariant property of the operator generating the entries of the **M2L** matrices, i.e. the operator introduced in Eq. 2.15:

$$\mathcal{T}_L(\mathbf{t},\lambda) := \sum_{l=0}^{L}(2l+1)h_l^{(1)}(\kappa|\mathbf{t}|)P_l(\langle \mathbf{t}/|\mathbf{t}|,\lambda\rangle)$$

where $\mathbf{t}$ denotes the **M2L** translation vector (using the terminology of Sect. 5.3.3) and $\lambda \in \mathbb{S}^2$ is a cubature node on the sphere. Suppose that the cubature rule is a Lebedev one and let $g \in \mathfrak{D}_3$. We have:

$$\begin{aligned}
\mathcal{T}_L(g\cdot\mathbf{t},\lambda) &= \sum_{l=0}^{L}(2l+1)h_l^{(1)}(\kappa|g\cdot\mathbf{t}|)P_l(\langle g\cdot\mathbf{t}/|g\cdot\mathbf{t}|,\lambda\rangle)\\
\textit{(g is an isometry)} &= \sum_{l=0}^{L}(2l+1)h_l^{(1)}(\kappa|\mathbf{t}|)P_l(\langle g\cdot\mathbf{t}/|\mathbf{t}|,\lambda\rangle)\\
\textit{(Lebedev rules invariant w.r.t. } \mathfrak{D}_3\textit{)} &= \sum_{l=0}^{L}(2l+1)h_l^{(1)}(\kappa|\mathbf{t}|)P_l(\langle \mathbf{t}/|\mathbf{t}|,g^{-1}\cdot\lambda\rangle)\\
&= \mathcal{T}_L(\mathbf{t},g^{-1}\cdot\lambda).
\end{aligned}$$

The last line indeed corresponds to the targeted result since the **M2L** matrices are diagonal in *hf-fmm*. The grids on the target and source cells are equal and are the same (Lebedev) cubature grid. Thus, for any λ,μ in this grid, the operator $\mathfrak{B}$ of Sect. 5.3.1.1 (Eqs. 5.3.1.1 and 5.19) corresponds to

$$\mathfrak{B}_{\mathbf{u}}(\lambda,\mu) := \begin{cases}\mathcal{T}_L(\mathbf{u},\lambda) & \textit{if } \lambda=\mu\\ 0 & \textit{otherwise}\end{cases}.$$

Hence, the use of Lebedev rules in *hf-fmm* allows to fully exploit the symmetries of the octree structure and the operator $\mathfrak{B}_{\mathbf{u}}$ in this context (see Sect. 5.3.3).

6.4.3 Switching between Lebedev and product grids

Since the use of Lebedev rules seems more promising at the deepest level than the Gauss-Legendre rules in terms of overall application cost while the contrary holds for the top tree levels (see Sect. 6.4.1.5), one may wonder if it is possible to fastly switch between Lebedev and product rules. Based on the group theoretical approach we proposed, this can be achieved thanks to Lem. 6.4.1.

Lemma 6.4.1. *Any product cubature rule on the sphere with uniformly sampled azimuthal angle using N points is invariant under the action of a subgroup of $\mathfrak{D}_3$ isomorphic to $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^3 \times \mathfrak{S}_2$ if*

- *4 divides N;*
- *The discretization of the polar angle is symmetric with regard to the hyperplane $z=0$.*

Proof. Let $\mathbb{D}$ be a product cubature grid on the sphere verifying the conditions of Lem. 6.4.1. Since 4 divides N, for any $\mathbf{z}=(z_1,z_2,z_3)\in\mathbb{D}$, the set $\{\mathbf{p}=(p_1,p_2,p_3)\in\mathbb{D} \mid p_3=z_3\}$ forms an uniform discretization of the circle invariant under the action of $\mathfrak{D}_2 \equiv \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^2 \times \mathfrak{S}_2$. Since there is a polar symmetry (i.e. an additional invariance under $\frac{\mathbb{Z}}{2\mathbb{Z}}$) due to the symmetry with regard to the hyperplane $z=0$, we obtain that $\mathbb{D}$ is invariant under the action of

$$\frac{\mathbb{Z}}{2\mathbb{Z}} \times \left(\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^2 \times \mathfrak{S}_2\right) \equiv \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^3 \times \mathfrak{S}_2.$$

□

Both the product Gauss-Legendre rules and the uniform ones verify the second condition listed in Lem. 6.4.1. As we discussed in Sect. 6.2.4, the condition "4 divides N" is needed in practice to exploit the symmetries of the octree structure and is an interesting choice. Hence, because the Lebedev rules are invariant under the action of $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^3 \times \mathfrak{S}_2 < \mathfrak{D}_3$, it is possible to exploit the block-diagonalization technique obtained in Thm. 6.3.5 to accelerate the interpolation between Lebdev rules and these product Gauss-Legendre or uniform rules. Notice that the restriction matrices of $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^3 \times \mathfrak{S}_2$ can be deduced from the restriction matrices of $\mathfrak{D}_3$.

6.4.4 Discussion on the practical applicability

The use of the Lebedev rules in *hf-fmm* allows to fully exploit the octree symmetries, which is not the case of the product cubature rules. This reduces the minimal number of **M2L** matrices to be precomputed from 34 (see [56, 197]) to 16 per level (on which the Lebedev rules are used). In addition, the application cost of a (diagonal) **M2L** operator is quasi-optimal using the Lebedev rules, contrary to the product ones. Actually, even the expansion sizes are reduced to their quasi-optimal size using the Lebedev rules, which has a beneficial impact on the memory footprint, still compared to the product rules.

However, even if the theoretical sizes of the **M2M**/**L2L** matrices are reduced using the Lebedev rules compared to the product ones, the theoretical complexity of these operators is greater using the Lebedev rules. Thanks to the block-diagonalization method, the application of these operators can be accelerated and practically becomes as costly as the **M2M**/**L2L** evaluations using the product Gauss-Legendre rules for small and mean integration orders.

To summarize, the cost of many the FMM operators (i.e. **P2M**/**M2L**/**L2P**) are lower using the Lebedev rules rather than the product Gauss-Legendre rules thanks to the quasi-optimality of the Lebedev rules. However, the cost of the **M2M**/**L2L** operators is not optimal using Lebedev rules on the top tree levels. The problem is that this last cost dominates the computations for large integration orders (i.e. at the top octree levels). Hence, discarding the full exploitation of symmetries on these top tree levels, the Gauss-Legendre rules may be more efficient than the Lebedev ones. In the end, we may distinguish two regimes such that the Lebedev rules should lead to more efficient FMM operator applications on the first one and the product rules should be better suited on the second one. Since our group theoretical approach allows to define a fast method to switch between Lebedev and product rules (see Sect. 6.4.3), an hybrid approach, combining the Lebedev rules for small and mean integration orders (i.e. on the deepest levels) with block-diagonalization of the **M2M**/**L2L** matrices and product rules for the highest levels should result in a highly efficient *hf-fmm* algorithm.

Part III

A new Fast Multipole Method library for oscillatory kernels

Chapter 7

defmm: Directional Equispaced interpolation-based Fast Multipole Method

Contents

The cost of the hierarchical methods for highly oscillatory kernels in the high-frequency regime is quite important. In this chapter, we present a new FMM library based on polynomial interpolations on equispaced grids and on a directional approach. The optimization of the operators of this library is detailed and comparisons with another directional interpolation-based FMM library are provided. These numerical experiments complement theoretical studies on the consistency of the proposed method and the way of combining Fourier techniques with the 2^d-tree symmetries.

7.1 A new directional FMM based on equispaced interpolation

In this section, we present the main high-level difficulties we face up when dealing with N-body problems and oscillatory kernels in the high-frequency regime (Sect. 7.1.1). Then, to handle these difficulties, we motivate the need of a new FMM library and the related high-level approach choices (Sect. 7.1.2). The principle particle distributions and the specific HPC architecture we work on are given in Sect. 7.1.3.

7.1.1 The high frequency challenge

The treatment of the high-frequency regime when dealing with oscillatory kernels with a FMM involves advanced methods that impact the overall complexity: modification of the MAC for kernel-independent directional methods (see Sect. 4.2.2.2) and diagonalization of large matrix blocks for kernel-explicit methods (see Sect. 2.2.4.3). The other kernel-independent alternatives are based on algebraic compression or diagonalization of large blocks in the high-frequency regime (see for instance [188]). The complexity of the kernel-explicit and non-directional methods strongly depends on the particle distributions: from $\mathcal{O}(N)$ on uniform distributions to $\mathcal{O}(N^2)$ on three-dimensional highly non-uniform ones (see [174]). Because of the ability of the directional kernel-independent methods to treat all types of particle distributions with the same overall complexity (see Sect. 4.2.2.2), this choice of approach seems promising for our applications. The directional methods appear as an extension of the kernel-independent method for low-frequency kernels themselves (see Sect. 4.2.2.4), meaning that the treatment of the low-frequency regime is straightforward. In addition, the capacity of considering general oscillatory kernels using a kernel-independent directional approach makes it very attractive. We therefore decided to focus on a directional kernel-independent approach.

Among the family of kernel-independent directional methods in more than one dimension, we have two options: the formulation based on *kifmm* presented in [38, 94–96] (see Sect. 4.2.1) or its adaptation to the interpolation-based FMM presented in [172–174] (see Sect. 4.2.2.4). Despite of the efficiency of *kifmm* in the low frequency regime, we opted for the interpolation-based approach for the following reasons:

- the polynomial interpolation tool is simple and can be applied to any (asymptotically) smooth kernel, without the need of potential theory;
- the compression scheme given in [96] is based on a random sampling of the wedges involved in the directional MAC (see Sect. 4.2.2.2), meaning that the compression scheme somehow depends on the MAC when using such compression.

The *dfmm*[1] library implementing the directional method described in [172–174], the only one we are aware of and which implements such a directional-interpolation-based FMM, is written following a list-based approach (see Sect. 2.2.4.1) and a tree construction using $MaxDepth$ (see Sect. 2.2.1.2). The interpolation rules used in *dfmm* are Chebyshev rules (see Sect. 4.1.1.3).

7.1.2 Motivations

In the polynomial interpolation-based FMMs, the interpolation process is used in practice with a fast evaluation scheme for the **M2L** operators (see Sect. 4.1.3). The authors of *dfmm* provide in [173] a set of methods for accelerating the (pre)computation and the evaluation of the **M2L** operators involved in their method. Each of these approaches is based on low-rank approximations. Among them, we are interested in the two best approaches: *dfmm-IAblk* (meaning *Individual*

[1] We will use this denomination since the code was named *dfmm* by the authors but one may keep in mind that this denomination was already used for the method proposed in [94].

Approximation with BLocKing and which is a combination of low-rank approximations and vector stacking using symmetries for BLAS 3 applications, see App. D.2) and *dfmm-SArcmp* (for *Single Approximation with ReCoMPression* and which corresponds to global low-rank approximations with additional recompressions, see Sect. 4.1.3.4). Notice that the efficiency of *dfmm-SArcmp* in terms of application timings has to be considered with its high precomputation cost.

The low-rank approximation methods are based on a parameter controlling the precision of the approximation which has to be tuned according to the interpolation rule. On the opposite, the methods based on interpolation grids composed of equispaced nodes (see Sect. 4.1.4) for non-oscillatory kernels, benefiting from FFT techniques to express the **M2L** matrices in diagonal forms, provide an efficient precomputation step with low memory requirements and a fast **M2L** operator evaluation. The precomputation step of a directional method being far more costly in a very high frequency regime than its low-frequency counterpart due to the increasing size of the interaction lists (see Sect. 4.2.2.2), the use of interpolation grids with equispaced nodes appears as a favorable choice. In addition, since FFTs are exact up to rounding errors, no additional threshold is needed to control the accuracy as in the low-rank methods. To our knowledge, there exists no directional-based FMM using FFT techniques for the **M2L** operator precomputation and evaluation. Because of the comparison results in [41, 64, 210] between the FFT techniques and low-rank methods, we expect an important computational gain by introducing the FFT techniques in the directional approach.

More precisely, the complexity of the application of such a diagonal **M2L** using the FFT-based methods is $\mathcal{O}\left((2L-1)^d\right)$, where L denotes the one-dimensional interpolation order (this notation will be widely used in the rest of this chapter). We provide in Tab. 7.1 a cost comparison of the storage, (pre)computation and application of a single **M2L** operator between *dfmm-IAblk* and the FFT-based techniques. *dfmm-SArcmp* does not benefit from a precise precomputation complexity analysis in the directional case, where the sizes of the interaction lists grow with the wavenumber.

	dfmm-IAblk	FFT-based
Precomputation	$\mathcal{O}\left(r^2L^d\right)$	$\mathcal{O}\left(d2^d(2L-1)^d log((2L-1))\right)$
Memory footprint	$2rL^d$	$(2L-1)^d$
Application	$2rL^d$	$(2L-1)^d$

Table 7.1: Complexity comparison of steps involving a single **M2L** operator between *dfmm-IAblk* and the FFT-based methods. L denotes the one-dimensional interpolation order, r the low-rank approximation for *dfmm-IAblk* and d the dimension.

Since $(2L-1)^d \leq 2^dL^d$, the application of a **M2L** using the FFT-based methods involves less operations than a product by a low rank matrix with rank 2^{d-1}, that is than the *dfmm-IAblk* method with $r = 2^{d-1}$ (which corresponds to a very small numerical rank in practice, associated to very low targeted accuracies). However, one has to keep in mind that using BLAS (see App. D.2) the matrix-vector products (*dfmm-IAblk*) are more efficient (because of a better use of cache memory) than vector-vector operations (FFT-based methods). Because of the vector stacking (see App. D.2), *dfmm-IAblk* may actually compute matrix-matrix products instead of matrix-vector products, depending on the particle distribution, which further increases the BLAS performance. This means that a comparison in terms of flops is not sufficient to represent the application timings and that *dfmm-IAblk* benefits from the uniformity of the particle distribution. Nevertheless, as the interpolation order L grows, the required rank r to preserve the targeted overall accuracy of the FMM using *dfmm-IAblk* increases and the theoretical complexity of the FFT-based methods quickly becomes better than the one of low-rank approaches. In addition, relying on the modern implementations of the FFT algorithm (such as in FFTW [107]), the precomputation step in the FFT-based methods can be efficiently performed. Our library thus relies on such FFT techniques for the efficient treatment of the **M2L** operators.

Finally, since we want to be able to apply our code to several non-uniform particle distributions,

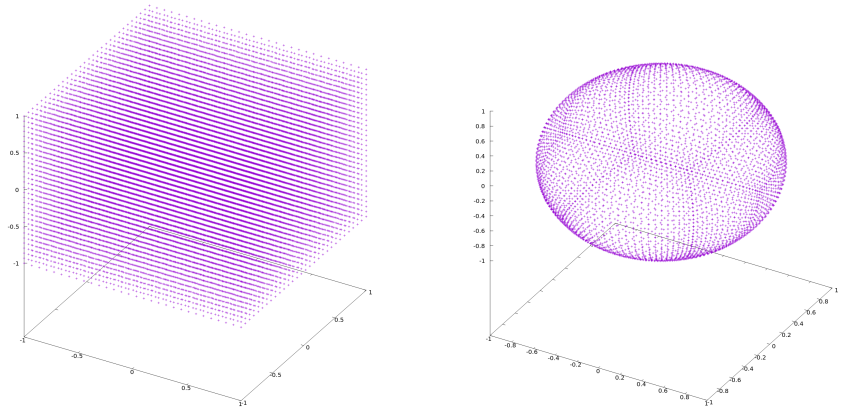


Figure 7.1: Particle distributions. Volume-uniform cube (left), surface-uniform sphere (right).

we choose to construct the trees with the *Ncrit* criterion (see Sect. 2.2.1.2). Indeed, this enables us to naturally adapt to the non-uniformity of the particle distributions (see Sect. 2.2.4.2). We will then use a relevant adaptive FMM strategy.

To summarize, on one hand the FFT-based methods on equispaced grids provide highly efficient performances (especially for the precomputation step), but these have not been investigated in a directional framework. Moreover they discard the need for a threshold controlling the error of the low-rank approximations. On the other hand the directional approach offers a kernel-independent way of handling the oscillations in the high-frequency regime with a linearithmic complexity for any particle distribution. We thus want to implement a directional-interpolation-based FMM using these FFT-based methods for the **M2L** operators treatment. We expect these choices to have important impacts on the performances.

To realize this new method, we developed our code, named *defmm* for "Directional Equispaced interpolation-based Fast Multipole Method". Our new directional FMM library uses quite different algorithmic choices than the implementations we are aware of. This chapter is dedicated to the theory on which our library is based, to its algorithmic design, to its implementation and to its optimization. To be used in practice, the consistency of the interpolation based FMM using grids with equispaced nodes has to be verified. This is the purpose of Sect. 7.2. In Sect. 7.3 are presented and motivated our design choices for *defmm*. We then provide in Sect. 7.4 how the symmetries are exploited in *defmm*. In Sect. 7.5 are presented several optimizations that we have introduced in *defmm*. We finally provide in Sect. 7.6 performance results of *defmm* and a comparison with the other directional interpolation-based FMM library we know, namely *dfmm*.

7.1.3 Particle distributions and test environment

To validate our code, we use a set of test cases presented here, starting from an uniform particle distribution in a cube with 10077696 elements. This corresponds to a volume distribution. This test case is referred to as the *cube* or as the *uniform cube* in the rest of this chapter.

The second test case we focus on is an almost uniform distribution over the surface of the sphere with 10140000 elements, in the sense that the particles are sampled over the unit sphere with no concentration around particular points. By considering such a distribution, the goal is to validate our method on surface meshes, i.e. on the distributions arising in the BIE context (see Sect. 2.1). Such a test case has been used to validate other FMM library, such as in [143]. This test case is referred to as the *sphere*.

These two distributions are depicted on Fig. 7.1.

The *defmm* library relies on other libraries widely used in an HPC context and that are listed in the following:

- BLAS for the matrix-vector and matrix-matrix products. To be more precise, we run our tests using the MKL (Math Kernel Library Version 2019.0.3).
- FFTW for the FFT computations. In practice, we use the MKL wrapper for FFTW3.

These are very standard dependencies in scientific computing. The *defmm* library will be compiled with:

- g++ (Version 8.2.0) with flags "-mprefer-vector-width=512 -Wall -O3 -g -ffast-math -mfma -fno-trapping-math -fopt-info-optall -march=native -mtune=native -fverbose-asm -fopenmp-simd -std=c++11".
- icpc (C++ Intel compiler, version 19.1.0.166) with flags "-Ofast -qopenmp-simd -fma -xhost -qopt-report=5 -qopt-report-phase=vec -qopt-zmm-usage=high -fPIC -g -DMKL_ILP64 -m64 -lpthread -lm -ldl -std=c++11".

The numerical results we present are obtained on an Intel Xeon Gold 6152 CPU with 384 GB of RAM.

7.2 Consistency

According to the numerical results given in [41,64,210], the FFT-based techniques using interpolation equispaced grids numerically lead to good accuracies, with a decreasing error as the interpolation order increases. However, because of the Runge phenomenon (see Sect. 4.1.1.2), this choice of grids is known to make the interpolation diverging in many applications, even when dealing with analytic functions. This problem was mentioned in [41]. In this section, we verify the consistency of the interpolation using equispaced grids in a FMM procedure for asymptotically smooth kernels. In Sects. 7.2.1, 7.2.2 and 7.2.3, the consistency proof is provided and in Sect. 7.2.4, the numerical stability of the method is discussed. We then give in Sect. 7.2.5 a few details on the way the consistency proof is extended to oscillatory kernels.

7.2.1 Main results

Because the FFT-based techniques for the **M2L** operators are only applied to radial kernels, we consider a radial kernel G and two cells t and s such that $G_{\mathbf{u}}(\hat{\mathbf{x}},\hat{\mathbf{y}}) := G(\mathbf{x},\mathbf{y})$ with $\mathbf{u} := ctr(t) - ctr(s)$, $\hat{\mathbf{x}} := \mathbf{x} - ctr(t)$, $\hat{\mathbf{y}} := \mathbf{y} - ctr(s)$. Suppose that $\hat{t} = \hat{s} = [-a,a]^d$, where $\hat{c} := \{\mathbf{z} - ctr(c) \mid \mathbf{z} \in c\}$ for any cell c. Hence, $\hat{t} \times \hat{s} = [-a,a]^{2d}$.

The consistency proof is based on Thm. 7.2.1, linking the MAC with the consistency of the interpolation process using equispaced grids.

Theorem 7.2.1. *Let $\mathcal{A}$ be a MAC such that*

$$\mathcal{A}(t,s) = 1 \Rightarrow \quad G_{\mathbf{u}} \text{ is analytic in each variable at any point in } [-a,a] \text{ with a convergence radius } R \text{ such that } R > \frac{2a}{e}.$$

The Lagrange interpolation of $G_{\mathbf{u}}$ on $t \times s$ using L equispaced points in each variable, denoted by $\mathcal{I}_L^{t\times s}[G_{\mathbf{u}}]$, verifies

$$\lim_{L\to+\infty} \left|\left|\mathcal{I}_L^{t\times s}[G_{\mathbf{u}}] - G_{\mathbf{u}}\right|\right|_{L^\infty(t\times s)} = 0$$

and the convergence is uniform.

The norm in Thm. 7.2.1 is defined as in Def. 7.2.1.

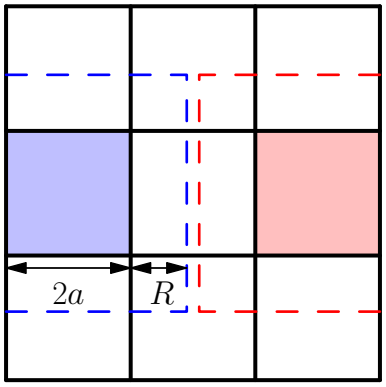


Figure 7.2: Target (blue) and source (red) cells with corresponding L^∞ balls (dashed lines) with radius strictly greater than $(1+\frac{2}{e})a$ and strictly smaller than $2a$.

Definition 7.2.1. *Let $f \in \mathcal{C}^\infty(t \times s)$. The $L^\infty(t \times s)$ norm of f is defined by*

$$||f||_{L^\infty(t\times s)} = \inf_{\mathbf{z}\in t\times s} |f(\mathbf{z})|.$$

In the FMMs using equispaced interpolation grids presented in [41, 42, 64, 188, 210], the chosen MAC is the strict MAC (see Sect. 2.2.4.1) of the uniform FMM that ensures, for two cuboid cells with side length $2a$ along each axis:

$$\forall\, R \in [0,a),\ G_{\mathbf{u}} \text{ is analytic on } B_\infty(ctr(t), a+R) \times B_\infty(ctr(s), a+R)$$

with $B_\infty(\mathbf{z}, r) := \{\mathbf{v} \in \mathbb{R}^d \mid ||\mathbf{v}-\mathbf{z}||_\infty \leq r\}$ the L^∞ ball of radius r centered in $\mathbf{z}$ (see Fig. 7.2). We can then provide the following corollaries of Thm. 7.2.1.

Corollary 7.2.1. *For any r such that $(1+\frac{2}{e})a < r < 2a$, we have:*

$$\left[\mathcal{A}(t,s) = 1 \Rightarrow G_{\mathbf{u}} \text{ analytic on } B_\infty(ctr(t),r)\times B_\infty(ctr(s),r)\right] \Rightarrow \left[\lim_{L\to+\infty} ||\mathcal{I}_L^{t\times s}[G_{\mathbf{u}}] - G_{\mathbf{u}}||_{L^\infty(t\times s)} = 0\right].$$

Notice that $[0, \frac{2a}{e}) \subset [0,a)$ (see Fig. 7.2). This means that the assumptions of Thm. 7.2.1 are verified in practice using the strict MAC (see 2.2.4.1) on 2^d-trees.

Remark 7.2.1. *Since the adaptive MAC (see Sect. 2.2.4.2) is such that two cells with a zero distance (i.e. sharing at least a point) are not well-separated, the result is also valid for such MAC on the same kernels. Because the strict MAC is the worst case for the proof of Thm. 7.2.1, we will consider this MAC in the rest of this section.*

Corollary 7.2.2. *Let $\mathcal{A}$ be the strict MAC. Then:*

$$\left[\mathcal{A}(t,s) = 1\right] \Rightarrow \left[\lim_{L\to+\infty} ||\mathcal{I}_L^{t\times s}[G_{\mathbf{u}}] - G_{\mathbf{u}}||_{L^\infty(t\times s)} = 0\right].$$

The presented results can be exploited in numerical methods only if the convergence is fast. At this stage, we have not discussed the speed of this convergence. To do so, we have to provide an effective proof of Thm. 7.2.1.

7.2.2 Preliminary results

The sketch of the proof of Thm. 7.2.1 is quite simple. The main idea consists in combining one-dimensional convergence estimates with the interpolation error results provided in [175]. Our

convergence proof differs from the one of [103, 174] because we cannot use the properties of the Chebyshev grids, which forces us to provide finer estimates, explicitly based on the analytic property of the interpolated function. We start by giving the theorem we are using to exploit one-dimensional estimates together to obtain an error estimate for the product multivariate interpolation. We denote by $\mathscr{C}^\infty(\Omega)$ 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 $\Omega \subset \mathbb{R}^{2d}$.

Theorem 7.2.2. *([175] Thm. 2.1) For $f \in \mathscr{C}^\infty([-a,a]^{2d})$, the product interpolation with the same one-dimensional rule in each variable is bounded in $[-a,a]^{2d}$ by*

$$\left|\left| f - \mathcal{I}_L^{[-a,a]^{2d}}[f] \right|\right|_{L^\infty([-a,a]^{2d})} \leq \sum_{\substack{\alpha=(\alpha_1,\dots,\alpha_{2d})\in\{0,1\}^{2d} \\ ||\alpha||_\infty = 1}} \omega_L^{\bar{\alpha}} \left|\left| \partial^{\alpha L} f \right|\right|_{L^\infty([-a,a]^{2d})}$$

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

Remark 7.2.2. *The results that can be found in [175] are quite more general: they can be applied to any cuboid and to different interpolation orders in each variable and to a larger class of functions (namely with essentially bounded weak derivatives). For the sake of simplicity, we restrict ourselves to $(2d)$-cubes and to a same interpolation order in each direction in the scope of this section. Since the kernels we are considering are in $\mathscr{C}^\infty(\Omega)$ for any domain Ω that do not contains the origin, which holds on any well-separated sets, the hypotheses we provided are sufficient.*

In the particular case we are interested in (i.e. equispaced grids), the constant ω_L can be bounded using the following lemma.

Lemma 7.2.1. *Let $\{x_k := -a + 2ak/(L-1) \mid k \in [\![0, L-1]\!]\} \subset [-a,a]$ an interpolation grid of equispaced point. We have $\omega_L \leq \left(\frac{2a}{L-1}\right)^L / (4L)$.*

Proof. Let $x \in [-a,a]$.

$$\begin{aligned}
\left| \prod_{j=0}^{L-1} (x - x_j) \right| &= (2a)^L \left| \prod_{j=0}^{L-1} \left(\frac{x}{2a} - \frac{x_j}{2a} \right) \right| \\
&= (2a)^L \left| \prod_{j=0}^{L-1} \left(\frac{x}{2a} - \frac{-a + 2aj/(L-1)}{2a} \right) \right| \\
&= (2a)^L \left| \prod_{j=0}^{L-1} \left(\frac{x-a}{2a} - \frac{j}{(L-1)} \right) \right|
\end{aligned}$$

and we have $y := \left(\frac{x-a}{2a}\right) \in [0,1]$. This leads to

$$\begin{aligned}
\left| \prod_{j=0}^{L-1} (x - x_j) \right| &\leq (2a)^L \left| \prod_{j=0}^{L-1} \left(y - \frac{j}{(L-1)} \right) \right| \\
&\leq \left(\frac{2a}{L-1} \right)^L \left| \prod_{j=0}^{L-1} y(L-1) - j \right|.
\end{aligned}$$

Because $y(L-1) \in [0, (L-1)]$, $\left| \prod_{j=0}^{L-1} y(L-1) - j \right|$ is maximal for $y \in (0, 1/(L-1))$. Majoring the product by a product of distances, choosing $y \in (0, 1/(L-1))$ for the two first terms of the product

and taking $y = 0$ for all the other terms, we get

$$\prod_{j=0}^{L-1} \Big| y(L-1) - j \Big| \ \leq |y||y-1| \prod_{j=2}^{L-1} j \ \leq \frac{1}{4} \prod_{j=2}^{L-1} j \ \leq \frac{(L-1)!}{4}$$

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 7.2.2. *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. 7.2.2.

Proof. Since f is analytic in all its variables, we can apply the Cauchy integral formula (see Thm. B.2.1), 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 Γ_j is a complex closed path encompassing $[-a, a]$ and verifying on any $p_j \in \Gamma_j$, $R > \min_{z \in [-a,a]} ||z - p_j||_\infty \geq r$. We thus have:

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

The term $\left(sup_{\mathbf{z} \in \Gamma_1 \times ... \times \Gamma_{2d}} |f(\mathbf{z})| \right)$ is bounded due to the analyticity of f on a neighborhood of $[-a, a]^{2d}$ encompassing the Γ_j's with a convergence radius equal to R, meaning that $\exists M(\Gamma_1, ..., \Gamma_{2d}) \in \mathbb{R}^{+*}$ such that, denoting by $C(\Gamma_1, ..., \Gamma_{2d}) := (2\pi)^{-2d} |\Gamma_1|...|\Gamma_{2d}| M(\Gamma_1, ..., \Gamma_{2d})$, where $|\Gamma_j|$ denotes the length of the path Γ_j, $j \in [\![1, 2d]\!]$,

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

where $\bar{\alpha} := \sum_{k=1}^{2d} \alpha_k$. □

7.2.3 Proof of the main theorem

It is now possible to give a proof for Thm. 7.2.1.

Proof. Following Lem. 7.2.1 and Thm. 7.2.2, we have

$$\begin{aligned}\left|\left|\mathcal{I}_L^{t\times s}[G_{\mathbf{u}}]-G_{\mathbf{u}}\right|\right|_{L^\infty([-a,a]^{2d})} &\leq \sum_{\substack{\alpha=(\alpha_1,\dots,\alpha_{2d})\in\{0,1\}^{2d}\\ ||\alpha||_\infty=1}}\left(\prod_{k=1}^{2d}\omega_L^{\alpha_k}\right)\left|\left|\partial^{\alpha L}G_{\mathbf{u}}\right|\right|_{L^\infty([-a,a]^{2d})}\\ &\leq \sum_{\substack{\alpha=(\alpha_1,\dots,\alpha_{2d})\in\{0,1\}^{2d}\\ ||\alpha||_\infty=1}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{L-1}\right)^L/(4L)\right)^{\alpha_k}\left|\left|\partial^{\alpha L}G_{\mathbf{u}}\right|\right|_{L^\infty([-a,a]^{2d})}\end{aligned}$$

which becomes, thanks to Lem. 7.2.2

$$\left|\left|\mathcal{I}_L^{t\times s}[G_{\mathbf{u}}]-G_{\mathbf{u}}\right|\right|_{L^\infty([-a,a]^{2d})}\leq C\sum_{\substack{\alpha=(\alpha_1,\dots,\alpha_{2d})\in\{0,1\}^{2d}\\ ||\alpha||_\infty=1}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{L-1}\right)^L\frac{L!}{4r^LL}\right)^{\alpha_k}r^{-1}.$$

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

$$\begin{aligned}\left|\left|\mathcal{I}_L^{t\times s}[G_{\mathbf{u}}]-G_{\mathbf{u}}\right|\right|_{L^\infty([-a,a]^{2d})} &\leq C\sum_{\substack{\alpha=(\alpha_1,\dots,\alpha_{2d})\in\{0,1\}^{2d}\\ ||\alpha||_\infty=1}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{L-1}\right)^L\frac{e^{-(L-1)}L^{L+1/2}}{4r^LL}\right)^{\alpha_k}r^{-1}\\ &\leq C\sum_{\substack{\alpha=(\alpha_1,\dots,\alpha_{2d})\in\{0,1\}^{2d}\\ ||\alpha||_\infty=1}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{L-1}\right)^L\frac{eL^{L+1/2}}{4(er)^LL}\right)^{\alpha_k}r^{-1}\\ &\leq C\sum_{\substack{\alpha=(\alpha_1,\dots,\alpha_{2d})\in\{0,1\}^{2d}\\ ||\alpha||_\infty=1}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{re}\right)^L\left(\frac{L}{L-1}\right)^L\frac{e}{4\sqrt{L}}\right)^{\alpha_k}r^{-1}.\end{aligned}$$

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

$$\begin{aligned}\left|\left|\mathcal{I}_L^{t\times s}[G_{\mathbf{u}}]-G_{\mathbf{u}}\right|\right|_{L^\infty([-a,a]^{2d})} &\leq C\sum_{\substack{\alpha=(\alpha_1,\dots,\alpha_{2d})\in\{0,1\}^{2d}\\ ||\alpha||_\infty=1}}\prod_{k=1}^{2d}\left(\left(\frac{2a}{re}\right)^L\frac{e}{\sqrt{L}}\right)^{\alpha_k}r^{-1}\\ &\leq Cr^{-2d}\sum_{\substack{\alpha=(\alpha_1,\dots,\alpha_{2d})\in\{0,1\}^{2d}\\ ||\alpha||_\infty=1}}\underbrace{\prod_{k=1}^{2d}\left(\left(\frac{2a}{re}\right)^L\frac{e}{\sqrt{L}}\right)^{\alpha_k}}_{=\left(\frac{2a}{re}\right)^{\bar{\alpha}L}\left(\frac{e}{\sqrt{L}}\right)^{\bar{\alpha}}}\end{aligned}$$

using $\bar{\alpha}=\sum_{k=1}^{2d}\alpha_k$. There is a finite number of terms in this sum depending only on the dimension: $2^{2d}-1$ terms (because $\alpha=(0,\dots,0)$ does not verify $||\alpha||_\infty=1$). This estimate 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 majored by $\left(\frac{2a}{re}\right)^L$ since $\frac{2a}{re}<1$. We

then have

$$\left|\left|\mathcal{I}_L^{t\times s}[G_{\mathbf{u}}] - G_{\mathbf{u}}\right|\right|_{L^\infty([-a,a]^{2d})} \leq \underbrace{\left(Cr^{-2d}(2^{2d}-1)\left(\frac{e}{\sqrt{2}}\right)^{2d}\right)}_{\text{Does not depend on } L} \underbrace{\left(\frac{2a}{re}\right)^L}_{\underset{L\to+\infty}{\to} 0}. \tag{7.1}$$

□

This proof has a geometric interpretation. In the inequality 7.1, the term $\left(\frac{2a}{re}\right)^L$ somehow corresponds to a MAC: a refers to the radius of an interacting cell (seen as a L^∞ ball) 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 get from the inequality 7.1 is that the convergence should be geometric in the one-dimensional interpolation order. This can be understood from the viewpoint of the adaptive MAC (see Sect. 2.2.4.2). Considering $\theta := \frac{rad(t)+rad(s)}{dist(t,s)}$, the estimate 7.1 somehow indicates that the error of the interpolation process is $\mathcal{O}\left(\left(\frac{\theta}{e}\right)^L\right)$.

7.2.4 Behavior at infinity with floating point arithmetic

There is still a problem arising when using floating point arithmetic and equispaced grids, even if the theoretical convergence is ensured. The ill-conditioning of the interpolation process on equispaced grids may cause an exponential amplification of rounding errors around the boundaries of the interpolation domain (see [181, 192]). We then cannot expect a numerical convergence for infinitely large orders. However, the question that remains is the following: for a given floating-point precision, above which interpolation order do the rounding errors cause divergence of the process?

We do not have a precise analysis of this error. Nevertheless, we present in Fig. 7.3 a numerical experiment that links interpolation on equispaced grids on well-separated sets and arithmetic precision. The numerical experiments illustrate our theoretical result and the ill-conditioning when the interpolation order becomes too large for the chosen arithmetic precision. To be more precise, the interpolation converges numerically up to a certain point, and then diverges in practice due to the accumulation of rounding errors. In practice, the interpolation order has to be chosen according to these considerations. Our applications usually require between 10^{-4} and 10^{-12} relative errors, which are correctly handled by the *double precision* format.

7.2.5 Oscillatory kernels

When the strict (or adaptive) MAC is used in the high-frequency regime, large degree polynomial interpolation are required to recover a prescribed accuracy [52]. This is a consequence of the difficulty of interpolating the complex exponential function for large arguments [202]. As explained in Sect. 7.2.4, too large orders cause numerical stability issues and would become too costly to be exploited in a fast method anyway. A directional approach [47, 50, 52, 94–96, 174] tackles this issue by combining directional approximations and specific MAC such that the modified interpolated kernel does not oscillate (see Sect. 4.2.1). In our case, this means that low order polynomials can still be used in the high-frequency regime when these hypotheses are verified. Thus, the numerical stability issue on large interpolation orders is not a problem. The directional MAC (see Sect. 4.2.2.2) used in this context also verifies the assumptions on the results provided in Sect. 7.2.1 and the interpolation of the directionally modified kernel on equispaced grids is correctly computed under these assumptions.

7.3 Design choices

In this section, we discuss the design choices and the high-level algorithmic approaches we opted for in *defmm* and their motivations. As opposed to *dfmm* that uses low-rank approximations

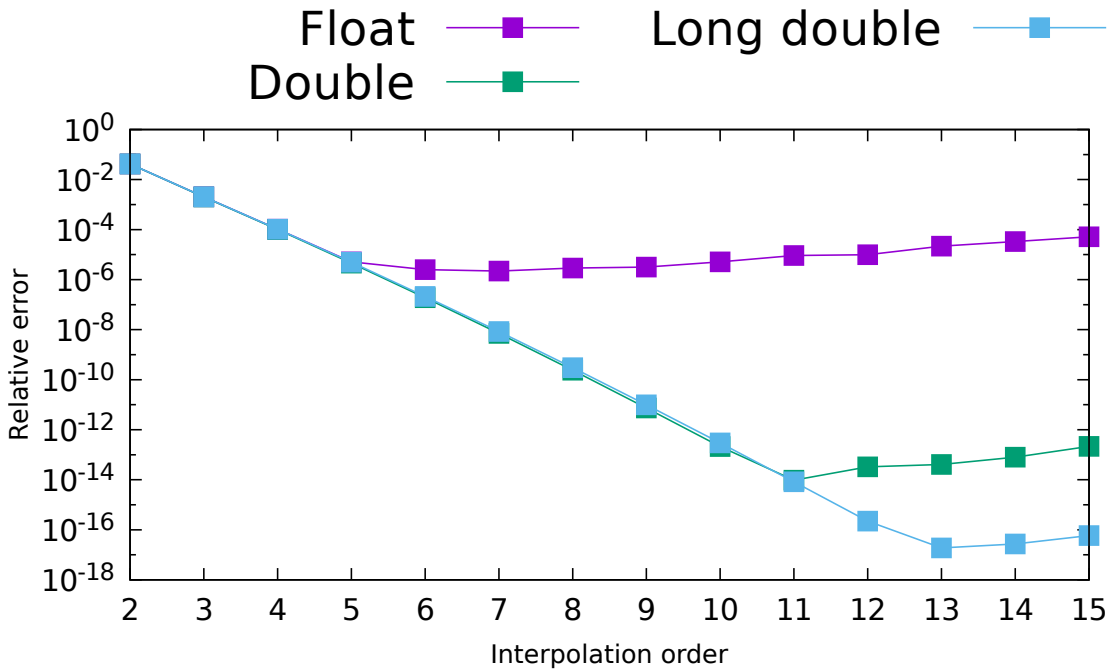


Figure 7.3: Relative maximum interpolation error of the one-dimensional kernel $\frac{1}{|x-y|}$ on well-separated sets according to the strict MAC (see Sect. 2.2.4.1) with respect to the interpolation order for different floating-point precisions.

to efficiently evaluate the **M2L** operators, we opted for FFT techniques in *defmm*, modifying the **M2L** operator expression (in the Fourier domain, see Sect. 4.1.4). The expression of the other operators actually remembers the definitions of the *dfmm* operators, except that we consider tensorized equispaced grids instead of tensorized Chebyshev ones. In Sect. 7.3.1, we provide the description of the procedures applying the FMM operators in *defmm*. In Sect. 7.3.2, we present the tree structure we use. In Sect. 7.3.3 are given the explicit tree traversals exploited in *defmm* and in Sect. 7.3.4 the tree construction algorithm is provided. In Sect. 7.3.5, we briefly detail how the directions are computed in our code. Finally, in Sect. 7.3.6 we described how the two frequency regimes communicate in our method.

7.3.1 *defmm* operators

As opposed to the FMMs in the low-frequency regime, the directional FMM associates to each high-frequency cell multiple expansions, themselves associated to different directions. Thanks to the direction tree (see Sect. 4.2.2.1), during the downward or upward passes, a given directional expansion associated to a cell c with direction u only communicates in the tree with directional expansions in $Father(c)$ associated to $u' \in Sons(u)$ and the directional expansions in $Sons(c)$ associated to $Father(u)$. Such links between expansions according to the direction tree are depicted in Figs. 7.4 and 7.5.

There are different ways for assembling the directional multipole expansions (and different ways for evaluating the local expansions): one may perform a single upward pass, assembling all the directional expansions of a given cell when all the directional expansions of its sons are known; or one may assemble a directional multipole expansion in a cell once the needed directional expansions of its sons are known (for instance, compute the directional expansion (R, d_{01}) in Fig. 7.5 when (A, d_{00}),(B, d_{00}),(C, d_{00}),(D, d_{00}) are computed but not (A, d_{01}),(B, d_{01}),(C, d_{01}),(D, d_{01})). The

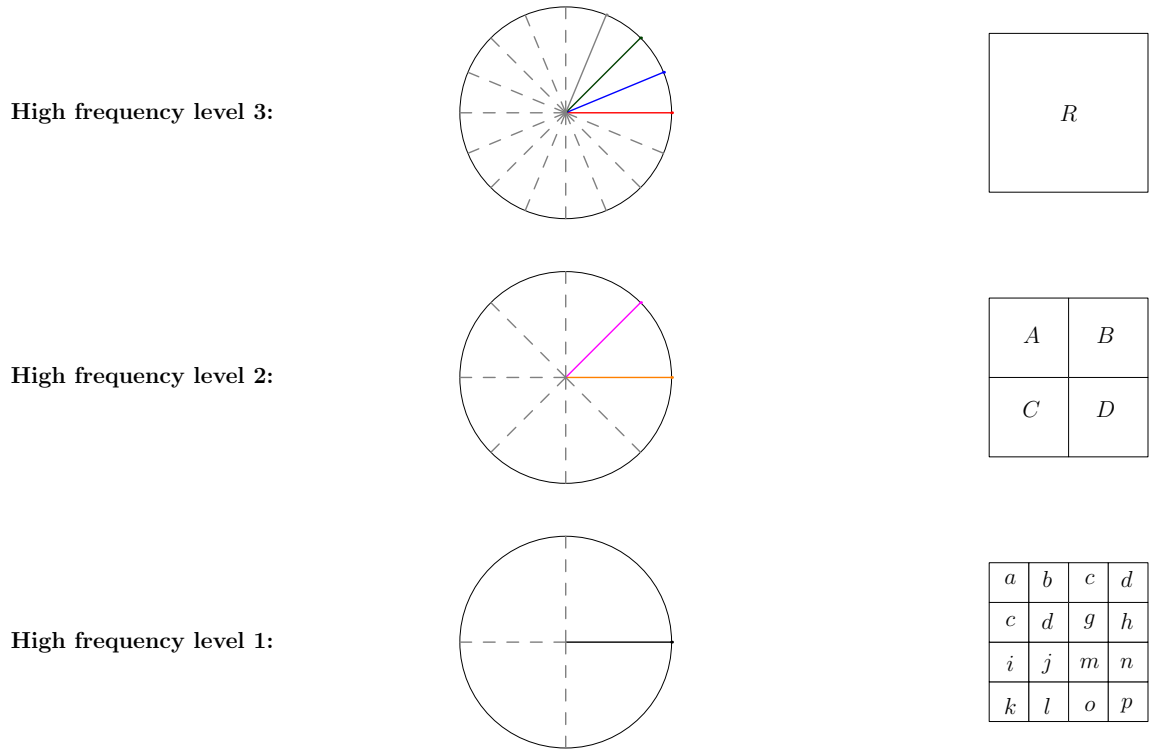


Figure 7.4: Quadtree splitting of $[-1,1]^2$ (right) and associated sets of directions (left) according to the high-frequency level.

second approach requires the upward pass to traverse the source tree according to the direction tree. On the opposite, the first approach does not impact the source tree traversal but the routines called during this upward pass, requiring to treat multiple directional expansions at each call. We decided to implement the first approach because this provides the same form of the upward pass in both the low- and high-frequency regimes. Hence, the definitions of the directional interpolation-based FMM operators as provided in Sect. 4.2.2.4 have to be further adapted into routines achieving this goal.

Since we use FFT techniques for the fast evaluation of the **M2L** operators (see Sect. 7.1.2), the multipole expansions used in Sect. 4.2.2.4 need to be transformed in the Fourier domain (and the local expansions in the Fourier domain to be transformed back) by applying Fourier matrices. In Sect. 7.3.1.1 we present two new operators that are introduced in *defmm* to perform these applications of Fourier matrices. In Sect. 7.3.1.2 are explicitly described the routines applying the FMM operators. Finally, in Sect. 7.3.1.3, we focus on the treatment of the **M2L** operators.

7.3.1.1 The M2F and F2L operators

In the FFT-based methods for the non-oscillatory kernels interpolated on equispaced grids (see Sect. 4.1.4) as presented in [41], each **M2M** or **P2M** application is followed by a FFT (i.e. the fast application of a Fourier matrix) and each **L2L** or **L2P** application is preceded by another FFT. Actually, these FFTs were included in the **P2M**, **M2M**, **L2L** and **L2P** applications but are independent of the (re)interpolation process.

Hence, to produce a general implementation of the directional interpolation-based FMM allowing to easily add other fast methods and for low-level optimizations in the specific context of the equispaced interpolation grids (see Sect. 7.5.6.1), we decided to separate the reinterpolation process (i.e. the **P2M**, **M2M**, **L2L**, **L2P** operators) from the conversion of the expansions into

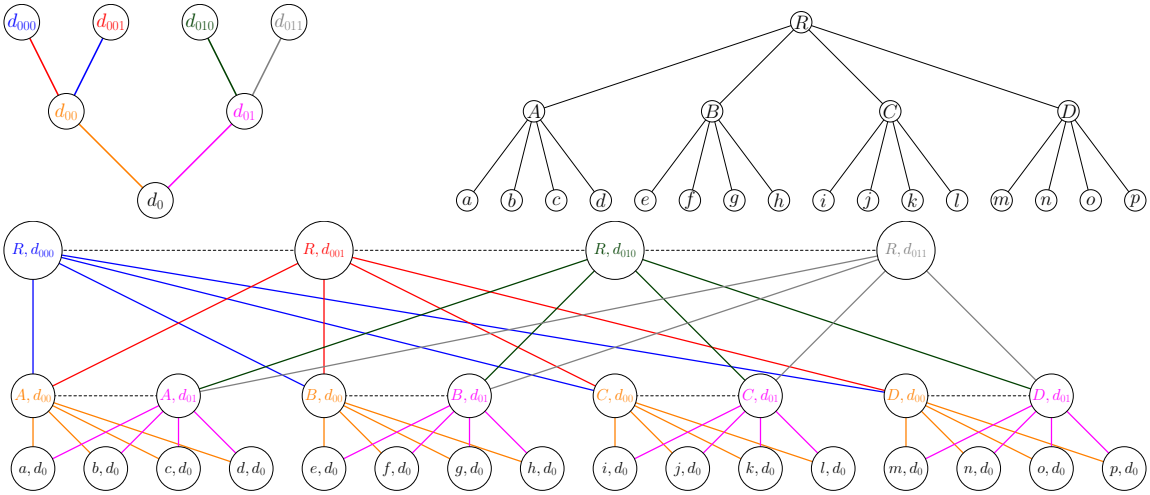


Figure 7.5: Tree representations of the cells in Fig. 7.4 (top right) and of non-dashed directions of Fig. 7.4 using the same colors (top left). Links between directional expansions according to the tree structures (bottom). There is one (directional) expansion per circle (that is per *cell-direction* pair at the same high-frequency level as in Fig. 7.4). Dashed lines link together the expansions associated to a same cell.

the Fourier domain (or into any other format allowing a fast **M2L** evaluation). This results in two new operators: the **M2F** and **F2L** ones (**M**ultipole-to-**F**ourier and **F**ourier-to-**L**ocal). Since we based our method on FFT techniques on equispaced grids, the **M2F** operator applies on a source cell and transforms its multipole expansion into a multipole expansion in the Fourier domain. This operator has to be applied *after* the **M2M** operator on this cell. In the same way, the **F2L** operator transforms a local expansion in the Fourier domain into a local expansion in the real domain and has to be performed *before* the evaluations of the local expansion (i.e. before the **L2L** evaluations on the corresponding cell).

Due to the wideband formulation in our approach, to the low- and high-frequency regime transitions, and to low-level implementation considerations, our FMM library *defmm* is structured in a specific way. Like for the other operators, two variants of the **M2F** and **F2L** operators exist in *defmm*, depending on the frequency regime. In Fig. 7.6 is depicted a schematic view of the links between all these operators.

7.3.1.2 Operator procedures

The procedures realizing the application of the operators in Fig. 7.6 are provided in Algs. 7-18. They use the FMM operators of the polynomial interpolation-based FMM described in Sect. 4.1.2.1 and their directional counterparts provided in Sect. 4.2.2.4.

The definition of the high-level procedure applying a **P2M** operator on a cell in the low-frequency regime directly follows the definition of this **P2M** as given in Sect. 4.1.2.1. This is detailed in Alg. 7.

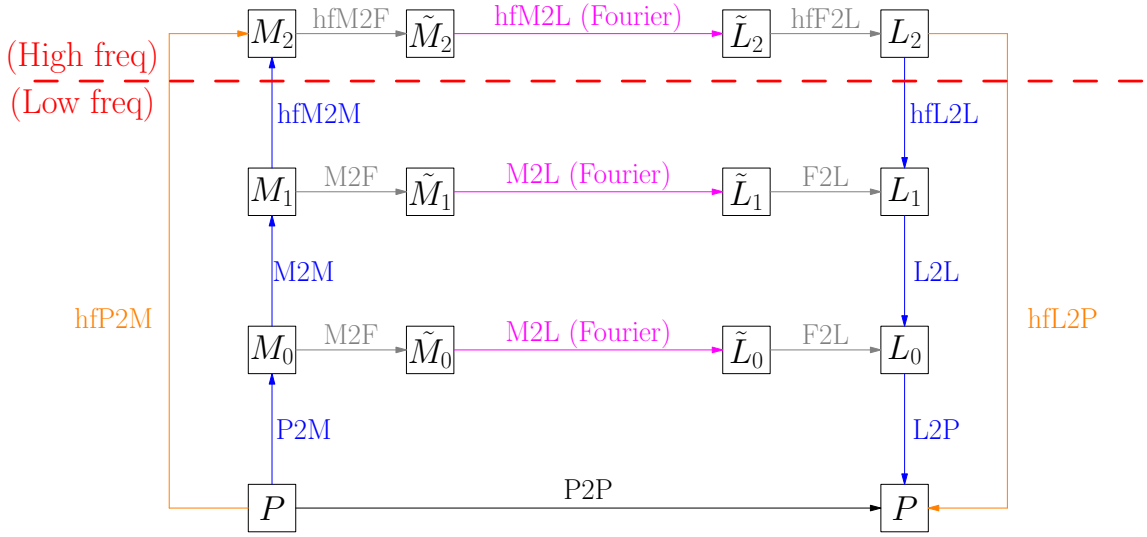


Figure 7.6: Links between expansions and operators in *defmm*. M: Multipole expansion, L: Local expansion, $\tilde{\text{M}}$: Multipole expansion in the Fourier domain, $\tilde{\text{L}}$: Local expansion in the Fourier domain, P: particles. An expansion D_i, $D \in \{M, L, \tilde{M}, \tilde{L}\}$, corresponds to an expansion at level $depth - i$, with *depth* the depth of the tree. The dashed red line corresponds to the boundary between high- and low-frequency regimes. The names of the high-frequency variants of the operators are preceded by "hf". In order of application: construction and aggregation of multipole expansions (bottom-up blue and orange arrows); conversion of the multipole expansions in the Fourier domain (left-right gray arrows from M_i's to $\tilde{M}_i$'s, *cell-cell* interactions (horizontal magenta arrows for **M2L** operators and black arrow for **P2P** one); conversion of local expansions in the Fourier domain into local ones (gray arrows between $\tilde{L}_i$'s and L_i's); forwarding and evaluations of local expansions (top-down blue and orange arrows).

Algorithm 7 P2M. $\mathfrak{q}_{|s}$ refers to the restriction of the vector of charges to the particles of s. $\mathcal{M}[s]$ refers to the multipole expansion in s.

1: // Input : cell $s \in \mathscr{S}$
2: // Output: $\emptyset$
3: **procedure** P2M(s)
4: $\quad \mathcal{M}[s] = \mathbf{P2M}[s] \cdot \mathfrak{q}_{|s}$ ▷ Assemble local expansion from charges and particles.
5: ▷ The **P2M** operator applied on s is defined as in Sect. 4.1.2.1.
6: **end procedure**

In the high-frequency regime, things are a bit more complicated. We want to determine *on-the-fly* which level of the direction tree $\mathbb{D}$ (see Sect. 4.2.2.1) is associated to a given cell (i.e. the high-frequency level indicated in Fig. 7.4). This is performed by the function $hflvl : \mathscr{H} \to \mathbb{N}^+$, $\mathscr{H} = \mathscr{T}, \mathscr{S}$, that associates to each cell the corresponding direction tree level plus one (since the level of the root of a tree is zero and since the first high-frequency level corresponds to the root of $\mathbb{D}$). Once this level is known, directional **P2M** operators are applied to the charges for all the directions associated to the considered high-frequency leaf. Such application is described in Alg. 8.

Algorithm 8 HFP2M. $\mathfrak{q}_{|s}$ refers to the restriction of the vector of charges to the particles of s. $\mathcal{M}_u[s]$ refers to the multipole expansion in s associated to the direction u.

// Input : cell $s \in \mathscr{S}$
// Output: $\emptyset$
procedure HFP2M(s)
  for $u \in \mathcal{D}_{hflvl(s)}$ **do** ▷ For each direction associated to $Level(s)$...
    $\mathcal{M}_u[s] = \mathbf{P2M}_u[s] \cdot \mathfrak{q}_{|s}$ ▷ ... assemble the *directional* multipole expansion.
    ▷ The **P2M** operator applied on s with
    ▷ direction u is defined as in Sect. 4.2.2.4
  end for
end procedure

In Algs. 9 and 10, we provide the procedures evaluating the **M2M** operators in the low- and high-frequency regimes, based on the same ideas than the **P2M** operators, except that the incoming multipole expansions are themselves directional, meaning that they have to be correctly selected. Thanks to the way the direction tree was constructed in Sect. 4.2.2.1, this can be easily done.

Algorithm 9 M2M. $\mathcal{M}[s']$ refers to the multipole expansion in $s' \in Sons(s)$. $\mathcal{M}[s]$ refers to the multipole expansion in s.

// Input : cell $s \in \mathscr{S}$, cell $s' \in Sons(s)$
// Output: $\emptyset$
procedure M2M(s, s')
  $\mathcal{M}[s] += \mathbf{M2M}[s, s'] \cdot \mathcal{M}[s']$ ▷ The result is added, not only computed!
end procedure

Algorithm 10 HFM2M. $\mathcal{M}_v[s']$ refers to the multipole expansion in $s' \in Sons(s)$ with direction v. $\mathcal{M}_u[s]$ refers to the multipole expansion in s with direction u.

// Input : cell $s \in \mathscr{S}$, cell $s' \in Sons(s)$
// Output: $\emptyset$
procedure HFM2M(s, s')
  for $u \in \mathcal{D}_{hflvl(s)}$ **do**
    $\mathcal{M}_u[s] += \mathbf{M2M}_u[s, s'] \cdot \mathcal{M}_{pred(u)}[s']$ ▷ $pred(u)$ is the father of u
    ▷ in the direction tree $\mathbb{D}$
  end for
end procedure

Recalling that, for any $\mathbf{z} = (z_1, ..., z_d) \in \mathbb{R}^d$, $|\mathbf{z}|_\infty := \max_{i=1,...,d} |z_i|$, the **M2F** operators in the low- and high-frequency regimes can be written as in Algs. 11 and 12 respectively. Their definition use the tensorized structure of the interpolation grids $\mathbb{G}$ (with the same order along each axis, that is $\mathbb{G} = H^d$, with H a one-dimensional grid composed of L equispaced nodes) and the representation of the multipole expansions as applications from $\mathbb{G}$ to $\mathbb{C}$. Once again, the high-frequency algorithm derives from the low-frequency one, only requiring a simple loop over the directions (since the Fourier matrix does not depend on the direction).

Algorithm 11 M2F. $\mathcal{M}[s]$ refers to the multipole expansion in s, $\tilde{\mathcal{M}}[s]$ is the Fourier multipole expansion in s. $\mathbb{F}$ refers to the d-dimensional Fourier matrix of size $(2L-1)^d \times (2L-1)^d$. $H := \{y_1, ..., y_L\}$ is the interpolation grid generating the grid $\mathbb{G}$ in s by means of tensorization.

1: // Input : cell $s \in \mathscr{T}$
2: // Output: $\emptyset$
3: **procedure** M2F(s)
4: Let $p \in \mathbb{C}^{(2L-1)^d}$
5: **for** $\mathbf{k} = (k_1, ..., k_d) \in [\![1, 2L-1]\!]^d$ **do**
6: $p_{\mathbf{k}} := \begin{cases} \mathcal{M}[s]((y_{k_1}, ..., y_{k_d})) & \text{if } |\mathbf{k}|_\infty \leq L \\ 0 & \text{otherwise} \end{cases}$ ▷ Zero-padding of multipole expansion
7: **end for**
8: $\tilde{\mathcal{M}}[s] = \mathbb{F} \cdot p$ ▷ Apply Fourier matrix (using FFT)
9: **end procedure**

Algorithm 12 HFM2F. $\mathcal{M}_u[s]$ refers to the multipole expansion in s associated to the direction u, $\tilde{\mathcal{M}}_u[s]$ is the Fourier multipole expansion in s associated to the same direction. $\mathbb{F}$ refers to the d-dimensional Fourier matrix of size $(2L-1)^d \times (2L-1)^d$. $H := \{y_1, ..., y_L\}$ is the interpolation grid generating the grid $\mathbb{G}$ in s by means of tensorization.

1: // Input : cell $s \in \mathscr{T}$
2: // Output: $\emptyset$
3: **procedure** HFM2F(s)
4: **for** $u \in \mathcal{D}_{hflvl(s)}$ **do**
5: Let $p \in \mathbb{C}^{(2L-1)^d}$
6: **for** $\mathbf{k} = (k_1, ..., k_d) \in [\![1, 2L-1]\!]^d$ **do**
7: $p_{\mathbf{k}} := \begin{cases} \mathcal{M}_u[s]((y_{k_1}, ..., y_{k_d})) & \text{if } |\mathbf{k}|_\infty \leq L \\ 0 & \text{otherwise} \end{cases}$
8: **end for**
9: $\tilde{\mathcal{M}}_u[s] = \mathbb{F} \cdot p$
10: **end for**
11: **end procedure**

As we mentioned in Sect. 4.2.2.4, the **M2L** operator in the (directional) polynomial interpolation-based FMM does not depend on the directionality. Actually, a high-frequency procedure is still needed in practice since the operators have different types of arguments depending on the frequency regime: non-directional or directional (Fourier) local/multipole expansions. In the high-frequency regime, the source multipole expansion has to be associated to the same direction in $\mathbb{D}$ than the target local expansion to which is added the result of the **M2L** operator application. This selection can be done *outside* the **M2L** operator application. In the low-frequency regime, only a single expansion exists in each cell, so there is no choice to do. The corresponding procedures are given in Algs. 13 and 14.

Algorithm 13 M2L. $\tilde{\mathcal{M}}[s]$ is the Fourier multipole expansion in s, $\tilde{\mathcal{L}}[t]$ is the Fourier local expansion in t.

// Input : cells $s \in \mathscr{T}$, $t \in \mathscr{T}$ such that $MAC(t,s) = 1$
// Output: $\emptyset$
procedure M2L(t, s)
    $D_{t,s}$:= **M2L** Fourier diagonal matrix between t and s
    for $\mathbf{z} \in [\![1, 2L-1]\!]^d$ **do** ▷ Perform the Hadamard product
        $\tilde{\mathcal{L}}[t](\mathbf{z}) += D_{t,s}(\mathbf{z})\tilde{\mathcal{M}}[s](\mathbf{z})$
    end for
end procedure

Algorithm 14 HFM2L. $u := \underset{v \in \mathcal{D}_{hflvl(s)}}{min} \left| v - \frac{ctr(t) - ctr(s)}{|ctr(t) - ctr(s)|} \right|$ $\tilde{\mathcal{M}}_u[s]$ is the Fourier multipole expansion in s associated to the direction u, $\tilde{\mathcal{L}}_u[t]$ is the Fourier local expansion in t associated to the same direction.

// Input : cells $s \in \mathscr{T}$, $t \in \mathscr{T}$ such that $MAC(t,s) = 1$
// Output: $\emptyset$
procedure M2L(t, s)
    $D_{t,s}$:= **M2L** Fourier diagonal matrix between t and s
    for $\mathbf{z} \in [\![1, 2L-1]\!]^d$ **do**
        $\tilde{\mathcal{L}}_u[t](\mathbf{z}) += \underbrace{D_{t,s}(\mathbf{z})}_{\text{Does not depend on } u} \tilde{\mathcal{M}}_u[s](\mathbf{z})$
    end for
end procedure

The **L2L** operators being the adjoint operators of the **M2M** ones and the **P2L** operators being the adjoint operators of the **P2M** ones, we directly obtain the definition of the corresponding procedures, that are listed in Algs. 15, 16, 17 and 18. The only difference with the already presented algorithms is that the directional expansions are aggregated together during the high-frequency **L2L** operators application procedures, according to the links in the direction tree. This leads to two loops instead of one when looping over the directions in the high-frequency **L2L** and **L2P** operators.

Algorithm 15 L2L. $\mathcal{L}[t']$ refers to the local expansion in $t' \in Sons(t)$. $\mathcal{L}[t]$ refers to the local expansion in t.

// Input : cell $t \in \mathscr{T}$, cell $t' \in Sons(t)$
// Output: $\emptyset$
procedure L2L(t', t)
    $\mathcal{L}[t'] += \mathbf{L2L}[t', t] \cdot \mathcal{L}[t]$
end procedure

Algorithm 16 HFL2L. $\mathcal{L}_v[t']$ refers to the local expansion in $t' \in Sons(t)$ with direction v. $\mathcal{L}_u[t]$ refers to the local expansion in t with direction u.

```
// Input : cell t ∈ 𝒯, cell t' ∈ Sons(t)
// Output: ∅
procedure HFL2L(t', t)
    for u ∈ 𝒟_{hflvl(t')} do
        for v ∈ Sons(u) do
            L_u[t'] += L2L_u[t', t] · L_v[t]
        end for
    end for
end procedure
```

Algorithm 17 L2P. $\mathfrak{p}_{|t}$ refers to the restriction of the vector of potentials to the particles of t. $\mathcal{L}[t]$ refers to the local expansion in t.

```
// Input : cell t ∈ 𝒯
// Output: ∅
procedure L2P(t)
    p_{|t} += L2P[t] · L[t]        ▷ The result is added, as opposed to the P2M!
end procedure
```

Algorithm 18 HFL2P. $\mathfrak{p}_{|t}$ refers to the restriction of the vector of potentials to the particles of t. $\mathcal{L}_u[t]$ refers to the multipole expansion in t associated to the direction u.

```
// Input : cell t ∈ 𝒯
// Output: ∅
procedure HFL2P(t)
    for u ∈ 𝒟_{hflvl(t)} do
        p_{|t} += L2P_u[t] · L_u[s]
    end for
end procedure
```

7.3.1.3 M2L operators treatment

The **M2L** treatment is done in *defmm* with FFT techniques based on equispaced interpolation grids. The theoretical efficiency of such scheme in terms of number of operations performed at each step involving a **M2L** matrix however has to be discussed.

Centered expansions. The FFT techniques require the source and target interpolation grids to be equal up to a translation in order to obtain a "small" circulant embedding of the **M2L** matrices (see Sect. 4.1.4). This requires the well-separated pairs of cells to be composed of cells with the same radius. This thus prevents the use of "decentering" techniques, consisting in shifting the cell centers and to adapt their radii to the local particle distribution (i.e. to adapt the boundaries of each shifting cell to the particles lying in it). Such technique has been used in [79]: its general idea is depicted on Fig. 7.7. One of the advantages of the decentering is that it may reduce the total number of **M2L** evaluations by maximizing the admissibility on the uppest tree levels.

On the other hand, when exploiting decentering techniques, the results of Sect. 5.3.3 on the total number of different **M2L** matrices per tree level do not hold. In the interpolation-based FMM, the **M2L** matrices are (far) more costly to compute than to apply. For instance, the computation of the diagonal **M2L** matrices obtained with the circulant embedding of Sect. 4.1.4 on equispaced grids

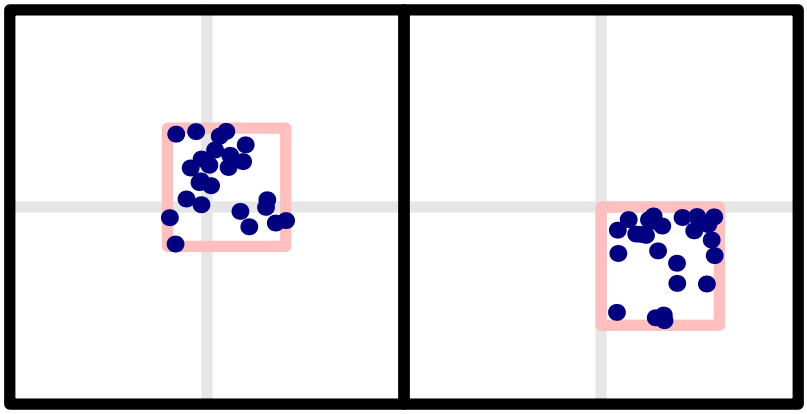


Figure 7.7: Adjacent cells (black) with clustered particles. These cells are not complying with the strict MAC (see Sect. 2.2.4.1) or with the adaptive MAC (see Sect. 2.2.4.2). Hence, discarding the gap between the smallest ball encompassing the particles in each cell, the resulting new cells (red) are admissible (considering the adaptive MAC). Because the nodes of a 2^2-tree have 4 sons (gray), this results in a single **M2L** call compared to 4 calls with centered expansions, the left cell having all its sons being non-empty and the right one having only one single non-empty child.

(i.e. the method used in *defmm*) requires $\mathcal{O}\left(dlog(L)(2L-1)^d\right)$ flops to be computed (see Eq. 4.10 where the Fourier matrix can be applied by using a FFT) but can be applied in $\mathcal{O}\left((2L-1)^d\right)$ flops (as a sparse matrix with $(2L-1)^d$ non-zero entries). In addition, the $(2L-1)^d$ kernel evaluations needed to assemble C_0 in Eq. 4.10 also have a non-negligible cost. Hence, for these reasons, the **M2L** (diagonal) matrices are precomputed in *defmm* without decentering techniques. This thus allows the results of Sect. 5.3.3 to be exploited. The constraint of admissible cells with same radius restrict ourselve to *cell-cell* interactions between cells of the same tree level (notice that *dfmm* uses the same constraint even with low-rank approximations instead of FFT techniques).

Hadamard products. The diagonal **M2L** matrices $D_{t,s}$ appearing in Algs. 13 and 14 are obviously not stored as matrices in *defmm* but as vectors composed of their diagonal entries. Hence, the diagonal matrix-vector product is a true Hadamard product in *defmm* (i.e. a vector-vector operation[2]). As a consequence, low performance gains can be expected with vector stacking (see App. D.2). We thus do not consider vector stacking for the **M2L** operator evaluations in our implementation.

M2L tables. The number of (possibly) precomputed vectors storing the diagonal **M2L** matrices $D_{t,s}$'s is bounded at each tree level thanks to the results of Sect. 5.3.3 (even in the high-frequency regime). Each of them can be uniquely identified using the vector $ctr(t) - ctr(s)$. Since we only have *cell-cell* interactions between cells at the same level, this is equivalent than considering the difference between the multi-indices locating t and s at their tree level. Hence, we referred to the d-dimensional array associating to each such difference the corresponding $D_{t,s}$ as the ***M2L** table* of the corresponding tree level. There are as many **M2L** tables as tree levels.

7.3.2 Adaptive method

Since we decided to exploit the directional approach, multiple expansions are associated to the same cell (but to different directions). We thus want to implement an adaptive FMM based on

[2]This operation actually does not have BLAS 1 (see App. D.2) implementation and then has to be written explicitly.

structures and traversals able to adapt to the (possibly highly non-uniform) particle distributions while efficiently handling the links of Fig. 7.5.

***Ncrit* criterion.** We decided to use a tree construction based on the *Ncrit* criterion (see Sect. 2.2.1.2) to easily adapt to possibly (highly) non-uniform particle distributions (see Sect. 2.2.4.2). Hence, the tree structure we are using does not store the empty cells: only the cells containing at least one particle are constructed. This is a strong difference with *dfmm*, that uses the criterion *MaxDetph* to build the 2^d-trees.

Dual Tree Traversal. In Sect. 2.2.4.2 we described the two main approaches to determine the interaction lists in the adaptive FMM: the list-based approach explicitly builds these interaction lists before evaluating them, whereas the Dual Tree Traversal one traverses them implicitly by finding the *cell-cell* interactions *on-the-fly*. Actually, because of the difficulty of finding the *cell-cell* interactions in the high-frequency regime, *dfmm* uses a DTT-like algorithm during a precomputation step to explicitly construct the interaction lists. Then, a list-based approach is used during the FMM application in *dfmm*.

On non-uniform distributions, the interaction lists may be sparse. Because the directional aspects strongly complicate these lists (multiple lists for each cell with varying sizes), we decided to implement a real DTT, meaning that the interaction lists are never built explicitly in *defmm*. The DTT approach also allows to easily handle the different MACs used in the wideband algorithm: the directional MAC (see Sect. 4.2.2.2) and the strict or adaptive one (see Sect. 2.2.4.1 and 2.2.4.2) provided that only the cell radii and distances are considered. This forces the directional MAC to be written *without* depending on any direction. This can be realized because the **M2L** matrices do not depend on the directions in a directional polynomial interpolation-based FMM (see [174] and Sect. 4.2.2.4).

Our DTT is quite different than the usual ones [79, 150, 220] because of the algorithmic choices we made. First, because the *cell-cell* interactions are efficiently treated with FFT techniques only for cells at the same level, we split both the target and source cells each time the MAC fails. Second, there are two frequency regimes with different MACs to take into account. We use a variant of the strict MAC (only considering a minimal distance criterion) in the low-frequency regime since this is the less restrictive MAC preserving the consistency of Thm. 7.2.1. In the high-frequency regime, we introduce the following new MAC derived from Eq. 4.15

$$\frac{max\{rad(s)^2\kappa, rad(t) + rad(s)\}}{dist(t,s)} \leq \eta \tag{7.2}$$

with, in practice, $\eta = 1$ and provided that $Level(t) = Level(s)$. If two cells in the high-frequency regime are well-separated according to this last MAC, the *cell-cell* interaction is performed on the best-suited directional expansions in t and s such that the MAC of Sect. 4.2.2.2 holds (i.e. the directional expansions associated to the closest direction at $Level(t) = Level(s)$ from $\frac{ctr(t)-ctr(s)}{|ctr(t)-ctr(s)|}$). Because the theoretical number of directions at each level is computed in order to ensure that at least one direction verifies the first condition of the directional MAC of Eq. 4.15, this is sufficient to guarantee a controlled global accuracy and to justify the MAC of the inequality 7.2. With such a formulation, the DTT does not depend on any direction.

In practice, since the search of the best suited-direction for a **M2L** operator application has a non-negligible cost, but since the number of different **M2L** operator is bounded at each tree level, we find these best-suited directions when precomputing the **M2L** (diagonal) matrices, independently of the interacting cells. This allows to use this information for each pair of well-separated cells complying with the MAC in the inequality 7.2.

Arithmetic intensity The Hadamard product we perform in the **M2L**s application are memory-bound (see App. D.1). In the *pvfmm* library [166, 167], a list-based method increasing the arithmetic

Pass	Traversal
Upward	Bottom-up
m2f	Any traversal (the order does not matter)
Horizontal	Dual Tree Traversal
j2l	Any traversal (the order does not matter)
Downward	Top-down

Table 7.2: Tree traversals for different passes.

intensity for a FMM with **M2L** operators performed by Hadamard products is implemented: this method groups in set of small matrices the entries of the expansions of the sons of a given cell (see App. D.2.4). The potential gain is limited for highly non-uniform distributions since the small created matrices are partially filled by zeros to handle the missing cells or expansions, at the cost of the expansion interleaving. Because we are interested in (highly) non-uniform particle distributions, we do not considered this particular technique in *defmm*.

In *dfmm*, the arithmetic intensity is also increased (using the *dfmm-IAblk* variant) by transforming BLAS 2 matrix-vector products into BLAS 3 matrix-matrix products [173]. Since we have Hadamard products instead of matrix-vector one, we do not rely in *defmm* on such techniques increasing the **M2L** arithmetic intensity. Hence, we do not need to complicate our DTT by imposing constraint grouping particular **M2L** operator applications.

7.3.3 The different algorithmic passes

The global algorithmic of *defmm* is splitted into five main passes, traversing the trees in different ways. Because of the choices we made (see Sect. 7.3), we are interested in the adaptive traversals based using recursive formulations (see 2.2.4.2). To the three passes used in the literature (upward pass in Alg. 19, horizontal pass in Alg. 21 and downward pass in Alg. 23), we added two other passes that respectively perform after the upward pass and before the downward one: the *m2f pass* and the *f2l pass*, respectively applying the **M2F** and **F2L** operators. These two passes are presented in Algs. 20 and 22. Each of these five passes corresponds to a specific tree traversal to respect the data dependencies. These traversals are summarized on Tab. 7.2. We provide below the high-level algorithms in their execution order, starting from the upward pass.

Algorithm 19 UPWPASS (Upward pass). This algorithm is called on the root of the source tree in order to be applied on all the source cells.

```
1: // Input : cell s
2: // Output: ∅
3: procedure UPWPASS(s)
4:     if isleaf(s) then
5:         if isHF(s) then
6:             hfP2M(s)
7:         else
8:             P2M(s)
9:         end if
10:    else
11:        for s' ∈ Sons(s) do
12:            UPWPASS(s')
13:            if isHF(Level(s)) then
14:                hfM2M(s, s')
15:            else
16:                M2M(s, s')
17:            end if
18:        end for
19:    end if
20: end procedure
```

The upward pass applies on the source tree using the **P2M**/**M2M** operators (i.e. construct the multipole expansion in the real domain), taking care on the frequency regime of each visited cell.

At the end of this pass, all the computed multipole expansions can be converted in the Fourier domain. This is the purpose of the *m2f* pass, also applied on the source tree.

Algorithm 20 M2FPASS.

```
1: // Input : tree 𝒮
2: // Output: ∅
3: procedure M2FPASS(𝒮)
4:     for s ∈ 𝒮 do
5:         if isHF(s) then
6:             hfM2F(s)
7:         else
8:             M2F(s)
9:         end if
10:    end for
11: end procedure
```

Because all the multipole expansions are known when the *m2f* pass is applied, the cells can be independently traversed in any manner. These Fourier multipole expansions are then transformed into Fourier local ones by the horizontal pass.

Algorithm 21 HRZPASS (Horizontal pass). This algorithm corresponds to a DTT and is called on the roots of the target and source trees.

```
1:  // Input : cells t and s
2:  // Output: ∅
3:  procedure HRZPASS(t, s)
4:      if isHF(t) and hfMAC(t, s) then          ▷ s is in the high-frequency regime if t is.
5:          hfM2L(t, s)
6:          return                  ▷ Stop the recursion when a cell-cell interaction is performed.
7:      else
8:          if not isHF(t) and MAC(t, s) then
9:              M2L(t, s)
10:             return
11:         end if
12:     end if
13:     if isLeaf(t) or isLeaf(s) then          ▷ The cell-cell interactions perform on cells
14:                                             ▷ at the same level, so if one of them is a leaf,
15:                                             ▷ we directly switch to the particle-particle interactions.
16:         P2P(t, s)
17:         return
18:     end if
19:     for t' ∈ Sons(t) do
20:         for s' ∈ Sons(s) do
21:             HRZPASS(t', s')                 ▷ Implies that Level(t') = Level(s').
22:         end for
23:     end for
24: end procedure
```

Only the *cell-cell* interactions involving cells at the same level are accelerated using FFT based techniques and we prohibit the interactions between cells of different levels (see Sect. 7.3.1.3). Hence, we apply the **P2P** operator (that is we switch to *particle-particle* interactions instead of *cell-cell* ones) when at least one of the two evaluated cells (line 13 of Alg. 21) is a leaf instead of the two cells (like in other DTTs: see e.g. *exafmm* [3]). This allows to group multiple **P2P** applications involving a same source or target cell and to maximize the size of this near field computation. This has an impact of the performances because of the arithmetic intensity of the **P2P** routine: a single application on large entries should perform better than multiple calls on smaller entries (see Sect. 7.5.5.1).

Once all the *cell-cell* interactions and *particle-particle* interactions are computed, we transform the Fourier local expansions in each target cell into local expansions in the real domain. This is done by the *f2l* pass.

Algorithm 22 F2LPASS.

```
1: // Input : tree 𝒯
2: // Output: ∅
3: procedure F2LPASS(𝒯)
4:     for t ∈ 𝒯 do
5:         if isHF(t) then
6:             hfF2L(t)
7:         else
8:             F2L(t)
9:         end if
10:    end for
11: end procedure
```

Finally, the downward pass properly evaluates the **L2P**/**L2L** operators on the target cells.

Algorithm 23 DWNPASS (Downward pass). This algorithm is called on the root of the target tree in order to be applied on all the target cells.

```
1: // Input : cell t
2: // Output: ∅
3: procedure DWNPASS(t)
4:     if isleaf(t) then
5:         if isHF(t) then
6:             hfL2P(t)
7:         else
8:             L2P(t)
9:         end if
10:    else
11:        for t' ∈ Sons(t) do
12:            if isHF(t) then
13:                hfL2L(t', t)
14:            else
15:                L2L(t', t)
16:            end if
17:            DWNPASS(t')
18:        end for
19:    end if
20: end procedure
```

7.3.4 Tree construction

The tree construction algorithm we use in our library is very similar to the one of Rio Yokota in *exafmm* [3]. Mainly, this algorithm is based on a particle sorting according to the Morton index (see App. D.3). All the source (resp. target) particles are stored in a single array and each cell contains a pointer on the first particle along with the number of particles in this cell. The particles of a given cell can then be directly accessed even for non-leaf cells. This particle sort is described in Alg. 24 and the process is illustrated on Fig. 7.8. The main differences between our implementation and the one of *exafmm* [3] are listed in the following.

- Our version is written for any dimension;
- The cells are allocated and created *after* the entire particle sorting.

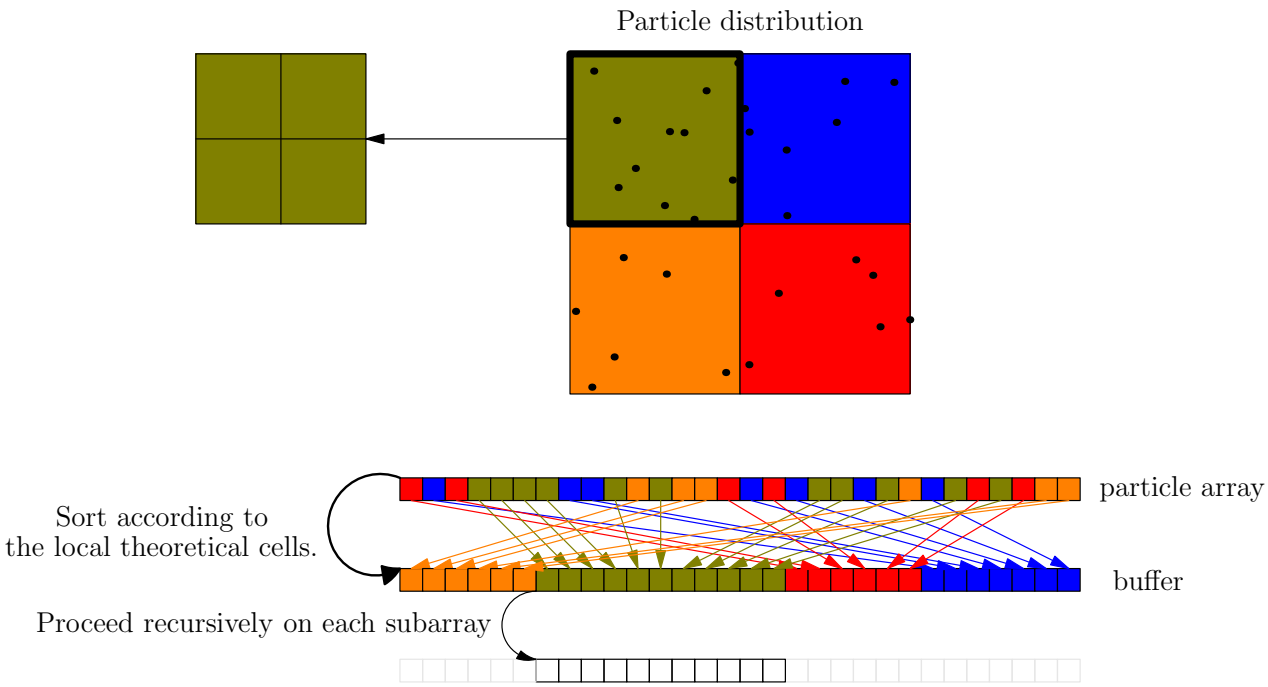


Figure 7.8: Schematic view of the recursive particle sort.

The particle sorting uses only two particle buffers that are switched at each recursive call (see Alg. 24). This algorithm actually traverses the non-empty cells that will be created after the particle sorting. The particles corresponding to a forthcoming cell are sorted in local subarrays on which the recursive calls are done. We use the stopping criterion $Ncrit$.

After the particle sorting, the exact number of cells is known (by counting the non-empty cells traversed by the sorting, i.e. the number of recursive calls to Alg. 24). We then allocate a cell array of this size and a traversal of the ordered particle array allows to fill this cell array but without introducing extra memory.

Timings for the entire tree building and particle sorting process are reported in Tab. 7.3.

Distribution	N	Time (s)	Time$\times 10^7$/N
Sphere	168931	0.08495	5.0
Sphere	1M	0.6054	6.1
Sphere	10M	6.40635	6.4
Cube	1M	0.5836	5.8
Cube	27M	17.4655	6.5
Cube	125M	87.891	7.0

Table 7.3: Timings of tree building (including the particle sorting) for several distributions.

By measuring the ratios between the tree construction timings and the number N of particles (last columns of Tab. 7.3), the practical $\mathcal{O}(N\,logN)$ complexity for the tree building is verified on both surface (the sphere) and volume distributions (the cube). Moreover, the tree building timings depend little on the particle distribution.

Algorithm 24 Recursive particle sorting.

1: // Input : a particle buffer Z storing the partially sorted particles, a particle buffer B with the same size (N), the number of theoretical levels lvl (set to zero for the first call) and a box C encompassing the particles in Z.
2: // Output: Stored particle buffer
3: **procedure** PARTICLESORT(Z, B, C, N, lvl)
4: **if** Stopping criterion is verified **then**
5: **if** lvl is odd **then**
6: Return B
7: **else**
8: Return Z
9: **end if**
10: **end if**
11: Divide C into 2^d equally-sized boxes C_i
12: Set to 0 the temporary number of particle n_i in C_i
13: **for** $\mathbf{z} \in Z$ **do**
14: Tag $\mathbf{z}$ with the index i such that $\mathbf{z} \in C_i$
15: Increment n_i
16: **end for**
17: Set $k_i := \sum_{j=0}^{i-1} n_j$ and $s_i = 0$ for any i
18: **for** $\mathbf{z} \in Z$ **do**
19: Let j be the tag of $\mathbf{z}$
20: Move the particle in the buffer: $B[k_j + s_j] = \mathbf{z}$
21: Increment s_j
22: **end for**
23: **for** $i \in [\![0, 2^d - 1]\!]$ **do**
24: **if** $n_i \neq 0$ **then** ▷ The empty sons are not considered.
25: PARTICLESORT($B + k_i, Z + k_i, C_i, n_i, lvl + 1$) ▷ Recursive calls on subarrays.
26: **end if**
27: **end for**
28: **end procedure**

7.3.5 Direction generation

To keep a fast algorithm, a nested structure in the direction sets of the different tree levels has to be ensured [52, 94]. Indeed, this limits the links between directional expansions to those depicted on Fig. 7.6, i.e. to the links between the directions of the direction tree. To construct such a nested set of direction, the algorithm we use in practice consists in recursively subdividing the faces of a d-cube and projecting the points obtained on the sphere. This corresponds to the algorithm described in [50, 94, 96]. Only the projected centers of each of the d-cube faces are used as possible directions for the deepest high-frequency level. The maximal number of directions is multiplied by 2^{d-1} on the next level. This choice of initial directions is sufficient to preserve the global targeted accuracy.

Remark 7.3.1. *As opposed to the directional method described in [94–96], the symmetries of the direction sets do not impact the symmetries in the **M2L** tables (that are only linked with the 2^d-tree structure and the interpolation grids since the **M2L** matrices do not depend on the directions in the interpolation-based FMM, see Sect. 4.2.2.4). Thus, any admissible nested set of direction such that each direction has the same number of sons can be used in defmm without modification of the code.*

As for the source and target trees, the direction tree (see Sect. 4.2.2.1) is computed once, during a precomputation step. The first high-frequency level uses $d!$ directions in our implementation, corresponding to the barycenters of the d-cube faces. Hence, since the number of directions is multiplied by a factor 2^{d-1} at each new high-frequency level, the l^{th} high-frequency level (corresponding to the $(l-1)^{th}$ level of the direction tree) have up to $d!2^{l(d-1)}$ possible directions.

In the high-frequency regime, the minimal distance between well-separated cells is proportional to the square of the cell length (multiplied by the wavenumber, see the MAC in Eq. 4.2.2.2). This implies that there may be no well-separated cells for the top levels: at least for the two top levels (as for the strict MAC of Sect. 2.2.4.1) and possibly for other top levels. In *defmm*, the directions are only computed for the high-frequency levels on which there may exist *cell-cell* interactions with well-separated cells (we still allow the *cell-cell* direct interactions using **P2P** applications on these upper levels). The number of upper tree levels with no far field interaction is found during the precomputation step, before creating the direction tree.

7.3.6 Transition between the two regimes

We distinguished the low- and high-frequency **M2M** and **L2L** operators in our implementation for efficiency reasons: any low-frequency level can be seen as a high-frequency level with a (false) direction equal to 0. Hence, all the evaluations of the planewaves directed by this direction are equal to 1 and the directional process has no effect. However, this introduces additional useless operations. We then considered that the low-frequency regime has no direction. There is a subtle consequence: when the limit between the low- and high-frequency is reached and the high-frequency FMM operators are applied, the expansions being non-directional, the formula in Sect. 4.2.2.4 have to be adapted. Our approach consists in considering the interpolation nodes on the low-frequency regime cells as particles from the viewpoint of the high-frequency cells. Doing so, the non-directional multipole expansions are transformed into directional ones by applying a high-frequency **P2M** operator and the directional local expansions are evaluated on low-frequency cells as **L2P** operations on the interpolation nodes of the low-frequency cell.

7.4 Symmetries

If we omit the diagonalization of the **M2L** operators in the Fourier domain, these operators, when applied to cells of the same levels of 2^d-trees with the strict MAC (see Sect. 2.2.4.1), verify the assumptions of Thm. 5.3.3. Hence, there exists a hyperoctahedral symmetry that can be exploited

to deduce some of them from a small set. This has a consequence on the precomputation cost of the overall FMM and can be used to obtain block algorithm using vector stacking (see [173] and App. D.2). In the Fourier domain, the vector stacking is no more justified since matrix-matrix products are performed with a left diagonal matrix, which does not lead to the same full exploitation of BLAS 3 routines (see Sect. 7.3.2). However, we may wonder if the hyperoctahedral symmetries are preserved in the Fourier domain that is if only a small set of **M2L** have to be precomputed. In Sect. 7.4.1, we illustrate the symmetries in the set of **M2L** matrices in the polynomial interpolation-based FMM. In Sect. 7.4.2, we are interested in extending these symmetries in the Fourier domain and an application to the **M2L** evaluation is provided in Sect. 7.4.3. In Sect. 7.4.4, we describe a new method able to handle the indirection problem appearing in this context.

7.4.1 Symmetries in M2L matrices

The content of this section is a direct illustration of Sect. 5.3.3 in the particular context of polynomial interpolation-based FMM. For the polynomial interpolation-based FMM applied to radial kernels, any **M2L** matrix $A_{t,s}$ between a target cell t and a source cell s of same radius in regular 2^d-trees with same roots is given by

$$(A_{t,s})_{k,l} = G(\mathbf{x}_k, \mathbf{y}_l) \tag{7.3}$$

where $\mathbf{x}_k$ is the k^{th} node of the interpolation grid $ctr(t) + \hat{\mathbb{G}}$ on t and $\mathbf{y}_l$ is the l^{th} node of the interpolation grid $ctr(s) + \hat{\mathbb{G}}$ on s with $\hat{\mathbb{G}}$ an interpolation grid obtained by product of the same one-dimensional grid with L nodes on the cube centered on zero with side length equal to the one of t and s. To be more precise, in the case of an equispaced grid:

$$\hat{\mathbb{G}} := \{\sum_{k=1}^{d} a(-1 + 2p_k/(L-1))\mathbf{e}_k \mid p \in [\![0, L-1]\!]^d\}$$

with $\mathbf{e}_k$ the k^{th} basis vector of $\mathbb{R}^d$ and a is the half of the side length of the cells $\hat{\mathbb{G}}$ applies on.

The two interpolation grids $ctr(t) + \hat{\mathbb{G}}$ and $ctr(s) + \hat{\mathbb{G}}$ are equal up to a translation $\mathbf{r} \in \mathbb{R}^d$, so we obtain

$$\begin{aligned}(A_{t,s})_{k,l} &= G(\mathbf{x}_k, \mathbf{y}_l)\\ &= G(\mathbf{x}_k - \mathbf{y}_l)\\ &= G(\hat{\mathbf{y}}_k - \hat{\mathbf{y}}_l + (ctr(t) - ctr(s)))\end{aligned}$$

where $\hat{\mathbf{y}} := \mathbf{y} - ctr(s)$ and $\hat{y}_p$ denotes the p^{th} node in $\hat{\mathbb{G}}$. Let $\mathfrak{D}_d$ be the hyperoctahedral group in dimension d, which is isomorphic to the group of rotations that preserve the d-cube and that can be represented by a set of $d \times d$ rotation matrices.

Lemma 7.4.1. *Let H be a one-dimensional interpolation grid on $[-1, 1]$ with a symmetry with regard to 0. Thus, the product rule $\hat{\mathbb{G}} := H^d$ is invariant under the action of $\mathfrak{D}_d$.*

Proof. Since $\hat{\mathbb{G}} := H^d$, for any $\mathbf{u} \in \hat{\mathbb{G}}$, for any permutation P of coordinates of $\mathbf{u}$, $P\mathbf{u}$ is still an element of $\hat{\mathbb{G}}$. Let D be a diagonal matrix with diagonal elements equal to ± 1. Since for any $h \in H$, $-h \in H$, thus $D\mathbf{u} \in \hat{\mathbb{G}}$. Since

$$\mathfrak{D}_d \equiv \{E \in \mathbb{R}^{d\times d} \mid \exists \text{ a permutation } P \text{ and a diagonal matrix } D \text{ with } \pm 1 \text{ on the diagonal such that } E = DP\},$$

the lemma holds. □

Remark 7.4.1. *Both the Chebyshev rule and the uniform rule on $[-1, 1]$ verify the assumption of Lem. 7.4.1.*

We suppose that $\hat{\mathbb{G}}$ is such that in Lem. 7.4.1. If the same rotation $g \in \mathfrak{D}_d$ is applied to these interpolation grids of t and s, we have

$$\begin{aligned} g \cdot \mathbf{x}_k - g \cdot \mathbf{y}_l &= g \cdot (\mathbf{x}_k - \mathbf{y}_l) \\ &= g \cdot (\hat{\mathbf{y}}_k - \hat{\mathbf{y}}_l + \mathbf{r}) \\ &= g \cdot (\hat{\mathbf{y}}_k - \hat{\mathbf{y}}_l) + g \cdot \mathbf{r} \end{aligned}$$

with $\mathbf{r} := ctr(t) - ctr(s)$. This means that the following holds, for any $g \in \mathfrak{D}_d$:

$$\left|(\hat{\mathbf{y}}_k - \hat{\mathbf{y}}_l) + g \cdot \mathbf{r}\right| = \left|g^{-1} \cdot (\hat{\mathbf{y}}_k - \hat{\mathbf{y}}_l) + \mathbf{r}\right|$$

since g can be interpreted as an isometry. Let $G_{\mathbf{r}}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) := G(|\hat{\mathbf{x}} - \hat{\mathbf{y}} + \mathbf{r}|)$, we have

$$(A_{t,s})_{k,l} = G_{ctr(t)-ctr(s)}(\hat{\mathbf{x}}_k, \hat{\mathbf{y}}_l)$$

using Eq. 7.3, where $\hat{\mathbf{x}}_k, \hat{\mathbf{y}}_l$ are respectively the k^{th} and l^{th} nodes of $\hat{\mathbb{G}}$. Hence, because the set

$$\mathcal{T} := \underset{t \in \mathscr{T}}{\cup} \{ctr(t) - ctr(s) \mid s \in \Lambda(t)\} \subset \mathbb{R}^d$$

using the strict, adaptive or directional MAC (see Sects. 2.2.4.1, 2.2.4.2 and 4.2.2.2 respectively) on perfect trees $\mathscr{T}$, $\mathscr{S}$ and with $\Lambda(t)$ the interaction list of t (see Sect. 5.3.3) is such that

$$\forall\, g \in \mathfrak{D}_d,\ g \cdot \mathcal{T} = \mathcal{T}$$

and there exists $t' \in \mathscr{T}, s' \in \mathscr{S}, g \in \mathfrak{D}_d$ with $ctr(t') - ctr(s') = g \cdot (ctr(t) - ctr(s))$ and

$$\begin{aligned} (A_{t',s'})_{k,l} &= G_{ctr(t')-ctr(s')}(\hat{\mathbf{x}}_k, \hat{\mathbf{y}}_l) \\ &= G_{g\cdot(ctr(t)-ctr(s))}(\hat{\mathbf{x}}_k, \hat{\mathbf{y}}_l) \\ &= G_{ctr(t)-ctr(s)}(g^{-1} \cdot \hat{\mathbf{x}}_k, g^{-1} \cdot \hat{\mathbf{y}}_l). \end{aligned}$$

Because g preserves the interpolation grids, there exist k', l' such that $\hat{\mathbb{G}} \ni \hat{\mathbf{x}}_{k'} = g \cdot \hat{\mathbf{x}}_k$, $\hat{\mathbb{G}} \ni \hat{\mathbf{y}}_{l'} = g \cdot \hat{\mathbf{y}}_l$. In addition:

$$\begin{aligned} (A_{t',s'})_{k,l} &= G_{ctr(t)-ctr(s)}(\hat{\mathbf{x}}_{k'}, \hat{\mathbf{y}}_{l'}) \\ &= (A_{t,s})_{k',l'}. \end{aligned}$$

Hence, by permuting the interpolation nodes (i.e. applying the transformations $k \to k'$ and $l \to l'$), $A_{t',s'}$ can be deduced from $A_{t,s}$ as well as all matrices A_{c_0,c_1} with c_0 and c_1 two cells of the same level than t and s and such that there exists $g \in \mathfrak{D}_d$ with $ctr(c_0) - ctr(c_1) = g \cdot (ctr(t) - ctr(s))$. An example is depicted on Fig. 7.9 in the case $d = 2$.

In [166], the possible symmetries of the **M2L** are exploited to increase the performances allowed by the BLAS 3 with vector stacking. This last technique on symmetries has been used in many implementations of many kinds of FMM (e.g. [151, 166, 173, 197]). Because the multipole and local expansions are converted in the Fourier domain when exploiting FFT-based techniques on equispaced grids, we have to determine the behavior of these permutations in the Fourier domain in order to use them in *defmm*.

7.4.2 Permutations in the Fourier domain

To be able to use the symmetries in the Fourier domain, we need to verify that the Fourier matrix (or its inverse) somehow commutes with any permutation matrix associated to a rotation of the hyperoctahedral group. The sense of this assertion has to be detailed: because the circulant embedding (see Sect. 4.1.4) increases the **M2L** matrix size, the corresponding permutation matrix has to be extended in a larger space. To be more precise, for any **M2L** matrix $U \in \mathbb{C}^{L^d \times L^d}$

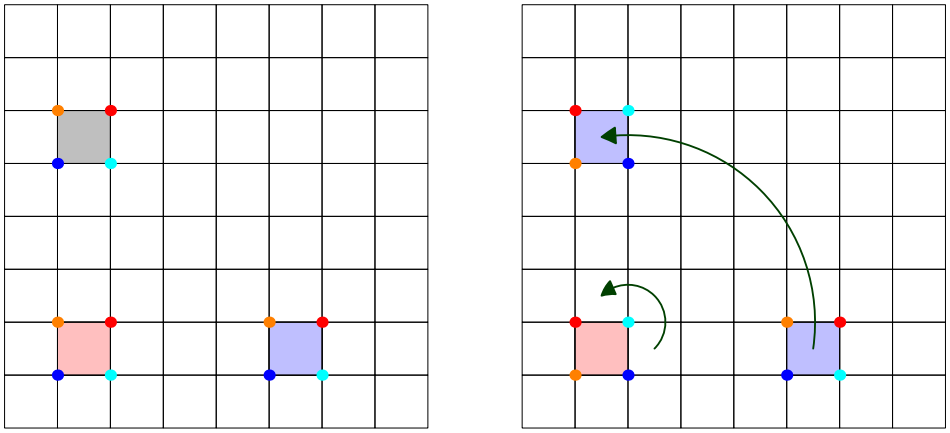


Figure 7.9: Goal: represent the interaction between the red and blue cells on the left figure using the data computed from the interaction between the red and gray ones. Left: target cell t (red) interacting with a source cell (gray) which is equal to a rotation centered in the center of t of another source cell (blue) with interpolation nodes (dots) on each cell. Right: Effective rotation of the cells. The positions of the interpolation nodes are permuted compared to the previous gray and red cells.

between two well-separated cells of the same level in the interpolation-based FMM, there exists $\chi \in \{0,1\}^{(2L-1)^d \times L^d}$ such that

$$U = \chi^T \mathbb{F}^* D[U] \mathbb{F} \chi$$

where $D[U] \in \mathbb{C}^{(2L-1)^d \times (2L-1)^d}$ is a diagonal matrix and $\mathbb{F} := \mathbb{F}_{2L-1,d}$ denotes the Fourier matrix in dimension d of size $(2L-1)^d \times (2L-1)^d$. Let V be another **M2L** matrix that can be obtained by permutation P from U, that is

$$\begin{aligned} V &= P^T U P \\ &= P^T \chi^T \mathbb{F}^* D[U] \mathbb{F} \chi P. \end{aligned}$$

Because the extensions (resp. restrictions) in the Fourier domain and the Fourier matrix are applied *before* (resp. *after*) the evaluation of the **M2L**s (in the Fourier domain), we want the permutation P to apply on $D[U]$ directly (notice that the two matrices have not the same size, so the definition of the permutation has to be adapted).

Before going into the details about this, we just provide a useful definition.

Definition 7.4.1. *Let $L \in \mathbb{N}$. $\mathfrak{I}$ denotes the bijection from $[\![0, L^d - 1]\!]$ to $[\![0, L-1]\!]^d$ such that $\mathfrak{I}^{-1}(\mathbf{I}) := \sum_{k=1}^{d} I_k L^{k-1}$, $\forall\, \mathbf{I} := (\mathfrak{I}_1, ..., \mathfrak{I}_d)$.*

Remark 7.4.2. *In practice, this mapping $\mathfrak{I}$ between indices and multi-indices being defined for any L such as in Def. 7.4.1, we never specify the space this function acts on.*

This mapping $\mathfrak{I}$ can be interpreted as a space filling curve and a representation of it is given in Fig. 7.10.

Basically, $\mathfrak{I}$ maps a multi-index into a one-dimensional one. This tool is particularly useful to refer to the multivariate interpolation nodes using the elements of the decomposition $\hat{\mathbb{G}} = H^d$. Because the entries of a multipole or local expansion correspond to interpolation nodes, $\mathfrak{I}$ maps a node to its corresponding entry in an expansion. The nodes of $H = \{h_k\}_{k \in [\![0, L-1]\!]}$ are supposed to be indexed such that $h_k < h_l \Leftrightarrow k < l$. Denoting by $\mathfrak{N}$ the mapping from the interpolation nodes in $\hat{\mathbb{G}}$ to their multi-indices, the following diagram can be drawn

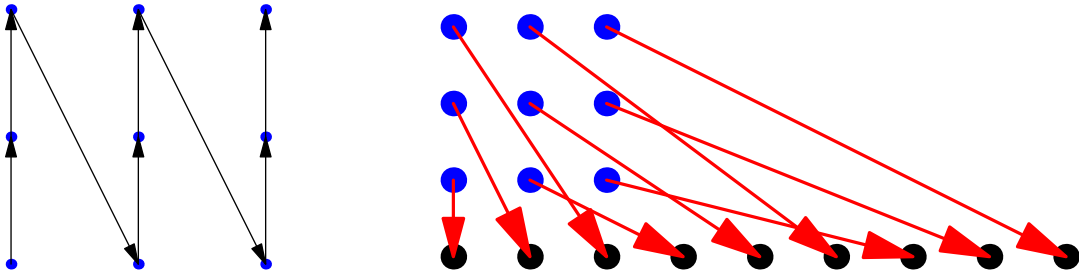


Figure 7.10: 2D mappings $\mathfrak{I}$ from indices to multi-indices(left) and $\mathfrak{I}^{-1}$ from multi-indices to indices (right). In the left picture, the first element (with no incident arrow) is indexed by 0 in $[\![0,2]\!]$ and $(0,...,0) \in Im(\mathfrak{I})$.

$$\begin{array}{ccc} \mathbb{C}[\hat{\mathbb{G}}] & \xrightarrow{\mathfrak{N}} & \mathbb{C}\left[\,[\![0,L-1]\!]^d\,\right] \\ & & \downarrow \mathfrak{I}^{-1} \\ & & \mathbb{C}\left[\,[\![0,(L-1)^d]\!]\,\right] \end{array}$$

and $\mathbb{C}[\hat{\mathbb{G}}]$ is the space in which the multipole/local expansions are. This allows to manipulate the terms of the expansions from a geometrical viewpoint (using the correspondances with the nodes of $\hat{\mathbb{G}}$) and in terms of vectors indexed by multi-indices at the same time.

We now show how to extend the permutations of Sect. 7.4.1 in the Fourier domain.

Theorem 7.4.1. *Let t, s be well-separated cells with the same interpolation grid $\hat{\mathbb{G}}$. Let U be the (Toeplitz) **M2L** matrix between t and s. Let $\mathbb{F}^* D[U] \mathbb{F}$ be the diagonalization of the circulant embedding of U. For any $g \in \mathfrak{D}_d$ with permutation representation $P_g \in \{0,1\}^{L^d \times L^d}$ in $Aut(\mathbb{C}[\hat{\mathbb{G}}])$, there exists $\mathbb{P}_g \in \{0,1\}^{(2L-1)^d \times (2L-1)^d}$ such that*

$$\begin{aligned} P_g^T U P_g &= P_g^T \chi^T \mathbb{F}^* D[U] \mathbb{F} \chi P_g \\ &= \chi^T \mathbb{P}_g^T \mathbb{F}^* D[U] \mathbb{F} \mathbb{P}_g \chi \\ &= \left(\chi^T \mathbb{F}^*\right) \left(\mathbb{P}_g^T D[U] \mathbb{P}_g\right) \left(\mathbb{F}\chi\right). \end{aligned}$$

Remark 7.4.3. *The permutations $\mathbb{P}_g$ can be explicitly computed by composing the representation of the permutations restricted to $\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d < \mathfrak{D}_d$ whose expression is provided in the proof of Thm. 7.4.1, see Eq. 7.4 or Eq. 7.5, and the permutations P_h, $h \in \mathfrak{S}_d < \mathfrak{D}_d$.*

Proof. First, we use $\hat{\mathbb{G}} = H^d$ with $H = \{h_0, ..., h_{L-1}\}$, $h_i < h_j \Leftrightarrow i < j$, $\#H = L$. A reflection with regard to 0 on $h_k \in H$ transforms h_k into h_{L-1-k}. We start with $d = 1$. The only possible symmetry (different than the identity) in dimension 1 is a reflection with regard to zero in H. The underlying group is isomorphic to $\frac{\mathbb{Z}}{2\mathbb{Z}} = \{id, g_0\}$ as an abelian group with two elements (the reflection of the reflection being the identity). Its action on H is such that

$$g \cdot h_k = \begin{cases} h_k & \text{if } g \equiv id \\ h_{L-1-k} & \text{otherwise.} \end{cases}$$

This can be trivially extended into an action on $\mathbf{N}_L := [\![0, L-1]\!]$ such that for any $k \in \mathbf{N}_L$

$$g \cdot k = \begin{cases} k & \text{if } g \equiv id \\ L-1-k & \text{otherwise.} \end{cases}$$

Since $Aut(\mathbb{C}[\hat{\mathbb{G}}]) \equiv Aut(\mathbb{C}[\mathbf{N}_L])$ by means of the bijection $\mathfrak{I}$ (trivial for $d = 1$), this gives an explicit expression for P_g, writting this permutation as a representation of $\frac{\mathbb{Z}}{2\mathbb{Z}}$ on $Aut(\mathbb{C}[\mathbf{N}_L])$:

$$(P_g)_{k,p} := \delta_{g\cdot k=p}$$

where $\delta_b := \begin{cases} 1 \text{ if } b \text{ is true,} \\ 0 \text{ otherwise} \end{cases}$.

Let $V = P_g^T U P_g$. We have

$$\begin{aligned} V_{k,p} &= U_{L-1-k,L-1-p} \\ &= U_{p,k} \end{aligned}$$

where the second equality is obtained using the Toeplitz structure of U. This means that $V = U^T$. Let $\mathbb{U} \in \mathbb{C}^{(2L-1)\times(2L-1)}$ be the circulant embedding of U such that for χ defined by

$$\chi_{k,p} := \delta_{k=p}\delta_{k<L}$$

we have $U = \chi^* \mathbb{U} \chi$.

Hence, there exist matrices $W_0 \in \mathbb{C}^{L\times(L-1)}, W_1 \in \mathbb{C}^{(L-1)\times L}, W_2 \in \mathbb{C}^{(L-1)\times(L-1)}$ such that

$$\mathbb{U} := \begin{bmatrix} U & W_0 \\ W_1 & W_2 \end{bmatrix}.$$

Since $\mathbb{U}$ is a circulant matrix, $\mathbb{U}^T =: \mathbb{V}$ also is and we have

$$\begin{aligned} \mathbb{V} &:= \begin{bmatrix} U^T & W_1^T \\ W_0^T & W_2^T \end{bmatrix} \\ &= \begin{bmatrix} V & W_1^T \\ W_0^T & W_2^T \end{bmatrix}. \end{aligned}$$

We obviously still have $\chi^* \mathbb{V} \chi = V$. Because $\mathbb{V}$ is circulant, its first row is sufficient to describe it. Such a first row can be interpreted as a vector $\mathbf{u} \in \mathbb{C}^{2L-1}$. Let $\mathbf{u}$ be this row, $\mathbf{u} \in \mathbb{C}[\mathbf{N}_{2L-1}]$, we are searching for $\mathbb{P}_{g_0} \in Aut\left(\mathbb{C}[\mathbf{N}_{2L-1}]\right)$ such that $\mathbb{P}_{g_0}\mathbf{u} = \mathbf{v}$.

Because $\mathbb{U}$ is a circulant matrix, we have

$$\begin{aligned} \mathbb{U}_{0,k} &= \mathbb{U}_{2L-1-k,0}\delta_{k\neq 0} + \mathbb{U}_{0,0}\delta_{k=0} \\ &= \mathbb{V}_{0,2L-1-k}\delta_{k\neq 0} + \mathbb{V}_{0,0}\delta_{k=0}. \end{aligned}$$

This gives an expression for $\mathbb{P}_{g_0}$:

$$\left(\mathbb{P}_{g_0}\mathbf{u}\right)_k = \begin{cases} \mathbf{u}_0 & \text{if } k = 0 \\ \mathbf{u}_{2L-1-k} & \text{otherwise.} \end{cases} \tag{7.4}$$

Notice that this allows to define the action of $g \in \frac{\mathbb{Z}}{2\mathbb{Z}}$ on $\mathbf{N}_{2L-1}$ as

$$g \cdot k := \frac{1}{2}\left(\left(\rho(g)+1\right)k + \left(1-\rho(g)\right)\left((2L-1-k)\delta_{k\neq 0} + k\delta_{k=0}\right)\right), \tag{7.5}$$

ρ being the sign representation of $\frac{\mathbb{Z}}{2\mathbb{Z}}$, i.e. $\rho(g) = \begin{cases} 1 & \text{if } g \equiv id \\ -1 & \text{otherwise} \end{cases}$. Indeed, the blue term in Eq. 7.5 vanishes when $g \equiv id$, which gives $g \cdot k = k$. On the contrary, when $g \neq id$, the red term vanishes and we have $g \cdot k = (2L-1-k)\delta_{k\neq 0} + k\delta_{k=0}$, corresponding to the expression in Eq. 7.4. We thus have $\left(\mathbb{P}_{g_0}\mathbf{u}\right)_k = \mathbf{u}_{g_0\cdot k}$. Notice that $g_0 \cdot (g_0 \cdot k) = k$, meaning that $g_0^{-1} = g_0$. This can be seen both from Eq. 7.5 or using that $\mathbb{Z}/2\mathbb{Z}$ is an abelian group with 2 elements.

One still has to verify that this permutation $\mathbb{P}_g$ commutes with the one-dimensional Fourier matrix $\mathbb{F}$ of size $(2L-1)\times(2L-1)$. Let $\mathbf{q}\in\mathbb{C}[\mathbf{N}_{2L-1}]$. We want to verify that $(\mathbb{F}\mathbb{P}_g\mathbf{q})_p=(\mathbb{P}_g\mathbb{F}\mathbf{q})_p$ for any p. Because the result is trivial for $g\equiv id$, we focus on the case $g=g_0$. For $p=0$, we have $\sum_{k=0}^{2L-2} e^{2i\pi\frac{p(g_0\cdot k)}{2L-1}}\mathbf{q}_k=\sum_{k=0}^{2L-2}\mathbf{q}_k$ that does not involve the action of g_0, so the result is true. We assume that $p\neq 0$. This gives:

$$
\begin{aligned}
(\mathbb{F}\mathbb{P}_{g_0}\mathbf{q})_p &= \sum_{k=0}^{2L-2} e^{2i\pi\frac{pk}{2L-1}}\left(\mathbb{P}_{g_0}\mathbf{q}\right)_k\\
&= \sum_{k=0}^{2L-2} e^{2i\pi\frac{pk}{2L-1}}\mathbf{q}_{g_0\cdot k}\\
(j:=g_0\cdot k) &= \sum_{j=0}^{2L-2} e^{2i\pi\frac{p(g_0^{-1}\cdot j)}{2L-1}}\mathbf{q}_j\\
(g_0^{-1}=g_0) &= \sum_{k=0}^{2L-2} e^{2i\pi\frac{p(g_0\cdot k)}{2L-1}}\mathbf{q}_k\\
(Eq.\ 7.5\ and\ g_0\not\equiv id) &= \sum_{k=0}^{2L-2} e^{2i\pi\frac{p(2L-1-k)}{2L-1}}\mathbf{q}_k\\
&= \sum_{k=0}^{2L-2} e^{2i\pi\frac{p(-k)}{2L-1}}\mathbf{q}_k\\
&= \sum_{k=0}^{2L-2} e^{2i\pi\frac{(-p)k}{2L-1}}\mathbf{q}_k\\
&= \sum_{k=0}^{2L-2} e^{2i\pi\frac{(2L-1)k}{2L-1}}e^{2i\pi\frac{(-p)k}{2L-1}}\mathbf{q}_k\\
&= \sum_{k=0}^{2L-2} e^{2i\pi\frac{(g_0\cdot p)k}{2L-1}}\mathbf{q}_k\\
&= (\mathbb{P}_{g_0}\mathbb{F}\mathbf{q})_p.
\end{aligned}
\tag{7.6}
$$

We thus obtained the result in dimension $d=1$. The extension of this result in more than one dimension (i.e. for arbitrary $d\in\mathbb{N}^*$) is based on the following facts:

- The multivariate Fourier matrices are tensorized matrices;
- The interpolation grid is a product grid $\hat{\mathbb{G}}=H^d$;
- $\mathfrak{D}_d\equiv\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d\times\mathfrak{S}_d\equiv\mathfrak{S}_d\times\left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d$ where each $\frac{\mathbb{Z}}{2\mathbb{Z}}$ corresponds to a possible reflection with regard to zero for a particular H of the decomposition of $\hat{\mathbb{G}}$.

Let $\mathbf{q}\in\mathbb{C}[\mathbf{N}_L^d]$ the mapping between the multi-indices of nodes of $\hat{\mathbb{G}}$ and the terms of the multipole expansion (i.e. the entries of vector in $\mathbb{C}^{L^d}$).

First, the multivariate definition of χ can be written as:

$$
\begin{array}{rccc}
\chi\ :\ & \mathbb{C}[\mathbf{N}_L^d] & \to & \mathbb{C}[\mathbf{N}_{2L-1}^d]\\
& \mathbf{v} & \mapsto & \chi\mathbf{v}
\end{array}
$$

such that for any $\mathbf{k}\in\mathbf{N}_{2L-1}^d$

$$
\left(\chi\mathbf{v}\right)_{\mathbf{k}}:=\begin{cases}\mathbf{v}_{\mathbf{k}} & if\ ||\mathbf{k}||_\infty<L\\ 0 & otherwise\end{cases},
$$

where $\mathbb{C}[\mathbf{N}_L^d] \overset{\mathfrak{I}^{-1}}{\equiv} \mathbb{C}[\mathbf{N}_{L^d}] \overset{\mathbf{q}}{\equiv} \mathbb{C}^{L^d}$ and $\mathbb{C}[\mathbf{N}_{2L-1}^d] \overset{\mathfrak{I}^{-1}}{\equiv} \mathbb{C}[\mathbf{N}_{(2L-1)^d}] \overset{\chi\mathbf{q}}{\equiv} \mathbb{C}^{(2L-1)^d}$.

Thanks to $\mathfrak{D}_d \equiv \mathfrak{S}_d \times \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d$, $\forall\, g \in \mathfrak{D}_d$, $\exists\, g_1 \in \mathfrak{S}_d,\ g_{(1)}, ..., g_{(d)} \in \frac{\mathbb{Z}}{2\mathbb{Z}}$ such that $g \equiv (g_1, g_{(1)}, ..., g_{(d)})$ and

$$P_g = P_{g_1} P_{g_{(1)}} ... P_{g_{(d)}}.$$

From a geometrical viewpoint, P_{g_1} permutes the coordinates of a node in $\hat{\mathbb{G}}$ and the $P_{g_{(j)}}$'s are possible reflexions with regard to 0 according to each axis (because $\hat{\mathbb{G}} = H^d$).

As for the one-dimensional case, we define $\mathbb{U}$ and $\mathbb{V}$ as the circulant embeddings of U and V respectively such that

$$\begin{cases} U = \chi^* \mathbb{U} \chi \\ V = \chi^* \mathbb{V} \chi \end{cases}.$$

The multivariate expressions of $\mathbb{P}_{g_{(1)}}, ..., \mathbb{P}_{g_{(d)}}$ are obtained similarly to the one-dimensional case using the decomposition $\mathbb{P}_{g_{(k)}} = \underbrace{Id \otimes ... \otimes Id \otimes \tilde{\mathbb{P}}_{g_{(k)}} \otimes Id \otimes ... \otimes Id}_{d\ terms}$ for any $k \in [\![1, d]\!]$ and where $\tilde{\mathbb{P}}_{g_{(k)}} \in Aut(\mathbb{C}[\mathbf{N}_{2L-1}])$ is defined as in Eq. 7.4 and is the k^{th} term of this decomposition. We easily check that $\mathbb{P}_{g_{(k)}} \chi = \chi P_{g_{(k)}}$. In addition, using $z \in \left(\frac{\mathbb{Z}}{2\mathbb{Z}}\right)^d$ so that $z := (g_{(1)}, ..., g_{(d)})$ with $\mathbb{P}_z := \prod_{k=1}^{d} \mathbb{P}_{g_{(k)}}$, we have $\mathbb{P}_z := \overset{d}{\underset{k=1}{\otimes}} \tilde{\mathbb{P}}_{g_{(k)}}$, thanks to the tensorized structure. The action of z on $\mathbf{N}_{2L-1}^d$ can be deduced from this tensorized expression and from Eq. 7.5.

χ preserves the coordinate exchanges of P_{g_1}, so $\mathbb{P}_{g_1}$ can be defined as

$$(\mathbb{P}_{g_1} \mathbf{v})_{\mathbf{p}} = \mathbf{v}_{\eta(g_1)\mathbf{p}}$$

where $\eta(g_1) \in \mathbb{C}^{d \times d}$ denotes the isometry representation of g_1 with $\mathfrak{S}_d$ acting on the d-cube as a set of rotations (exchanging the coordinates). This is equivalent to a permutation representation on a set of d element (which is an isometry). Let $\mathbb{F}_d := \overset{d}{\underset{k=1}{\otimes}} \mathbb{F}$, with $\mathbb{F}$ the one-dimensional Fourier matrix. We then have

$$\begin{aligned}
(\mathbb{F}_d \mathbb{P}_g \mathbf{q})_{\mathbf{p}} &= (\mathbb{F}_d \mathbb{P}_{g_1} \mathbb{P}_z \mathbf{q})_{\mathbf{p}} \\
&= \sum_{\mathbf{k} \in \mathbf{N}_{2L-1}^d} e^{2i\pi \frac{\langle \mathbf{p}, \mathbf{k} \rangle}{2L-1}} (\mathbb{P}_z \mathbf{q})_{\eta(g_1) \cdot \mathbf{k}} \\
&= \sum_{\mathbf{k} \in \mathbf{N}_{2L-1}^d} e^{2i\pi \frac{\langle \mathbf{p}, \eta(g_1)^{-1} \cdot \mathbf{k} \rangle}{2L-1}} (\mathbb{P}_z \mathbf{q})_{\mathbf{k}} \\
(\eta(g_1)\ isometry) &= \sum_{\mathbf{k} \in \mathbf{N}_{2L-1}^d} e^{2i\pi \frac{\langle \eta(g_1) \cdot \mathbf{p}, \mathbf{k} \rangle}{2L-1}} (\mathbb{P}_z \mathbf{q})_{\mathbf{k}} \\
&= (\mathbb{F}_d \mathbb{P}_z \mathbf{q})_{\mathbf{p}} \\
&= (\mathbb{P}_{g_1} \mathbb{F}_d \mathbb{P}_z \mathbf{q})_{\mathbf{p}}.
\end{aligned}$$

We conclude by exploiting the tensorized structure of the Fourier matrix:

$$\begin{aligned}
\mathbb{F}_d \mathbb{P}_z &= \left(\overset{d}{\underset{k=1}{\otimes}} \mathbb{F} \right) \left(\overset{d}{\underset{k=1}{\otimes}} \tilde{\mathbb{P}}_{g_{(k)}} \right) \\
&= \overset{d}{\underset{k=1}{\otimes}} \left(\mathbb{F} \tilde{\mathbb{P}}_{g_{(k)}} \right) \\
(see\ Eq.\ 7.6) &= \overset{d}{\underset{k=1}{\otimes}} \left(\tilde{\mathbb{P}}_{g_{(k)}} \mathbb{F} \right) \\
&= \mathbb{P}_z \mathbb{F}_d,
\end{aligned}$$

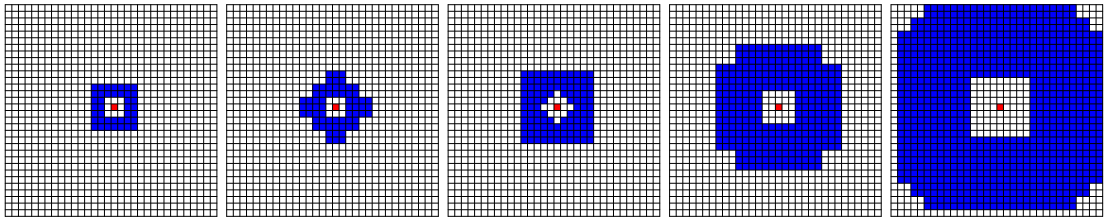


Figure 7.11: Evolution of the union of interaction lists (blue) of target cells (red) for increasing high-frequency tree levels (from left to right).

the third equality being obtained thanks to the one-dimensional case proof. Hence, this gives $\mathbb{F}_d\mathbb{P}_g = \mathbb{P}_g\mathbb{F}_d$.

□

7.4.3 Applying the permutations

The numerical exploitation of Thm. 7.4.1 allows to permute the expansions in the Fourier domain. However, thanks to the diagonal **M2L** structure in this domain, these manipulations can be done in a different way than in the standard methods exploiting the symmetries (e.g. [173]). In this section, we describe how the permutations are used in *defmm*.

7.4.3.1 Single permutation

Using these permutations, the multipole and local expansions are permuted respectively before and after the evaluation of a **M2L** operator in [173]. On the contrary, we can reduce here this number of applied permutations to only one thanks to the diagonal form of the **M2L** matrices in the Fourier domain in *defmm*. Let $\odot$ denote the Hadamard product and $\mathcal{U} \in \mathbb{C}^{(2L-1)^d}$. The permuted **M2L** evaluation, for a multipole expansion $\mathcal{M}$ in the Fourier domain can be written as

$$\begin{aligned} P_g^T diag(\mathcal{U}) P_g \mathcal{M} &= P_g^T \left(\mathcal{U} \odot (P_g \mathcal{M})\right) \\ &= \left(P_g^T \mathcal{U}\right) \odot \left(P_g^T P_g \mathcal{M}\right) \\ &= \left(P_g^T \mathcal{U}\right) \odot \mathcal{M}. \end{aligned}$$

Hence, by explicitly programming these Hadamard products, only the vector storing the diagonal values of the diagonalization of the **M2L** matrix has to be permuted. This reduces the number of permutations from two to only one.

7.4.3.2 Compressed M2L tables

As opposed to the low-frequency FFT based techniques, the number of possible admissible source cells with a fixed target one increases with the size of these cells in the high-frequency regime. Because the **M2L** matrices of the polynomial interpolation-based FMM do not depend on the direction (see Sect. 4.2.2.4), the directional MAC (see Sect. 4.2.2.2) somehow depends on the ratio between the interacting cells distance and radii in the high-frequency regime (see Sect. 7.3.2). This leads to a bounded set of theoretically different **M2L** operators at each level which is invariant under the action of $\mathfrak{D}_d$ (see Fig. 7.11). Hence, by storing the permutations P_g for any $g \in \mathfrak{D}_d$, the number of precomputed **M2L** in the Fourier domain is drastically reduced. This can be seen as the quotient of the set of possible **M2L** by the symmetries of the 2^d-trees, considering classes of **M2L** by means of permutations. In other words, this corresponds to a reduction of the **M2L** tables of Sect. 7.3.1.3. Because of the increasing size of the number of possible **M2L** matrices to compute in the high-frequency regime, the exploitation of the symmetries is important to

- Limit the precomputation time;
- Reduce the memory footprint.

In terms of application times, the use of symmetries does not increase the performance of the **M2L** processing: actually one may think that the performances can be deteriorated by the introduction of a permutation in the Hadamard product (see Sect. 7.4.3.1), especially when considering vectorization (see App. D.1). We thus propose a novel approach to obtain easy to vectorize code despite the use of these permutations.

7.4.4 Aligning data according to group permutation

We adopt in this section a general viewpoint, considering any group $\mathcal{G}$ even if in practice we limit ourselves to $\mathfrak{D}_d$. This is because the permutations we provided in Sect. 7.4.2 can be applied in a larger context than the one we looked at. Let $\mathcal{G}$ be a finite group acting on $\mathbb{E} := [\![0, N-1]\!]$ and for any $g \in \mathcal{G}$, let $p_g : \mathbb{E} \to \mathbb{C}$ be a bijective map such that $g \in \mathcal{G}$, $x \in \mathbb{E}$, $p_g(x) = g \cdot x$. Obviously, $p_g \in \mathbb{C}[\mathbb{E}]$ can be represented as a vector in $\mathbb{C}^N$. Let us consider three other arbitrary vectors in $\mathbb{C}^N$, namely x, y and u.

We are concerned by the fast evaluation of the permuted Hadamard products of Alg. 25.

Algorithm 25 Permuted Hadamard product

```
1: procedure HADP(x, u, p_g, y, N)
2:     for i ∈ E do
3:         x[i] += u[p_g[i]] * y[i]
4:     end for
5: end procedure
```

Due to the array indirections induced by p_g, the memory accesses are not predictable which can prevent the vectorization of the loop in Alg. 25. Fortunately, the group invariance can help to localize data in some cases we are going to describe. This could enable vectorization on subsets of the set $\mathbb{E}$.

7.4.4.1 $\mathcal{G}$-orbits

The basic idea of this localization is to sort the data of each vector according to the $\mathcal{G}$-orbits of the elements of $\mathbb{E}$. We recall the definition of $\mathcal{G}$-orbit already provided in Def. 5.3.4.

Definition 7.4.2. *Let $a \in \mathbb{E}$. The $\mathcal{G}$-orbit of a, denoted by $\langle a \rangle_{\mathcal{G}}$, is the set*

$$\langle a \rangle_{\mathcal{G}} := \{g \cdot a \mid g \in \mathcal{G}\}.$$

Their may exist different types of $\mathcal{G}$-orbits depending on the position of a on a fundamental domain (see Sect. 6.2.1). Each different orbit will have a specific treatment. If a has a singular $\mathcal{G}$-orbit, there exists at least an element $g \in \mathcal{G}$ that keeps a invariant: $g \cdot a = a$. Any kind of singular orbit can thus be characterized using this information.

Definition 7.4.3. *The set of invariants of $a \in \mathbb{E}$ is the set*

$$\mathscr{I}_{\mathcal{G}}(a) := \{g \in \mathcal{G} \backslash \{\mathbf{1}_{\mathcal{G}}\} \mid g \cdot a = a\}$$

where $\mathbf{1}_{\mathcal{G}}$ refers to the identity element in $\mathcal{G}$.

Notice that this definition differs from the stabilizers (see Sect. 6.3.4) since we excluded the identity element. The classification of orbits is done according to the set of invariants. We introduce the orbit class of an element of $\mathbb{E}$.

Definition 7.4.4. *$a, b \in \mathbb{E}$ are said to lie in the same orbit class if and only if $\mathscr{I}_{\mathcal{G}}(a) = \mathscr{I}_{\mathcal{G}}(b)$. The orbit class of an element $a \in \mathbb{E}$ is denoted by $\mathcal{C}(a)$. The set of orbit classes of $\mathcal{G}$ is denoted by $\mathcal{C}(\mathcal{G})$.*

Finally, for any orbit, we can choose a generator representing all the orbit using actions of $\mathcal{G}$.

Definition 7.4.5. *For any orbit $\langle a \rangle_{\mathcal{G}}$, a generator $[a]_{\mathcal{G}}$ of this orbit is a fixed choice of element of this orbit: $\forall a_0, a_1 \in \langle a \rangle_{\mathcal{G}}$, $[a_0]_{\mathcal{G}} = [a_1]_{\mathcal{G}}$.*

Remark 7.4.4. *A fundamental domain $\mathcal{F}_{\mathbb{E}}(\mathcal{G})$ of $\mathcal{G}$ on $\mathbb{E}$ can be defined as a (minimal) set of generators of $\mathcal{G}$-orbits of elements of $\mathbb{E}$.*

7.4.4.2 Removing the array indirections

Let $N(c)$ be the number of different orbits in the orbit class $c \in \mathcal{C}(\mathcal{G})$ and let c_k be a generator of the k^{th} orbit of this class. We can rewrite Alg. 25 according to the classes and orbits as shown in Alg. 26.

Algorithm 26 Permuted Hadamard product

```
1: procedure HADP(x, u, p_g, y)
2:     for c ∈ C(G) do                                   ▷ For each class...
3:         for h ∈ G\I_G(c) do          ▷ ...for each group action not keeping the class invariant...
4:             for i ∈ [[0, N(c) − 1]] do                ▷ ...and for each orbit in this class...
5:                 x[h · c_i] += u[p_g[h · c_i]] ∗ y[h · c_i]   ▷ ... perform the permuted Hadamard product.
6:             end for
7:         end for
8:     end for
9: end procedure
```

The key idea is that for any $g, h \in \mathcal{G}$ and any $a \in \mathbb{E}$, due to the group property of $\mathcal{G}$, $\exists f := (gh) \in \mathcal{G}$ such that:

$$\begin{aligned} p_g[h \cdot a] &= g \cdot (h \cdot a) \\ &= (gh) \cdot a \\ &= f \cdot a. \end{aligned}$$

The Hadamard product line 5 in Alg. 26 can be expressed as

$$x[h \cdot c_i] += u[f \cdot c_i] * y[h \cdot c_i].$$

Now, for $c \in \mathcal{C}(\mathcal{G})$ and $g \in \mathcal{G}$, let $z_{c,g}$ be the vector such that

$$(z_{c,g})_i := z[g \cdot c_i],$$

$i \in [\![0, N(c) - 1]\!]$, $z = u, x, y$. In other terms, $z_{c,g}$ fixes the class c and the group action g and concatenates the entries of z corresponding to the generators in c permuted by g. This set can be interpreted as the set $g \cdot (\mathcal{F}_{\mathbb{E}}(\mathcal{G}) \cap c)$, i.e. the permutation of the restriction of a fundamental domain to the elements in the class c. This is depicted on Fig. 7.12 and all the other orbits types are depicted on Fig. 7.13. We obtain, using these notations, the Alg. 27.

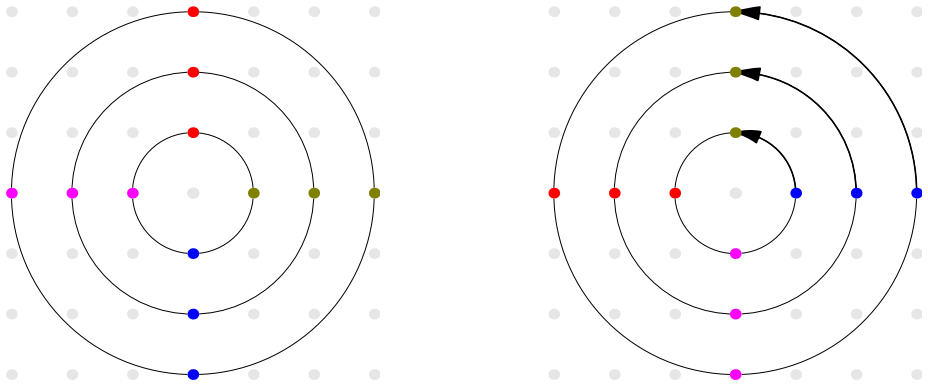


Figure 7.12: c is the class invariant under the action of $\mathfrak{D}_{0,1} \times \mathfrak{D}_{0,2}$ (horizontal and vertical symmetries). Each color represent a $(\mathcal{F}_{\mathbb{E}}(\mathcal{G}) \cap c)$ for x and y (left). Example of $g \cdot (\mathcal{F}_{\mathbb{E}}(\mathcal{G}) \cap c)$ for u (right). The Hadamard product without indirection groups the nodes of new same positions in the two grids, one color set at a time.

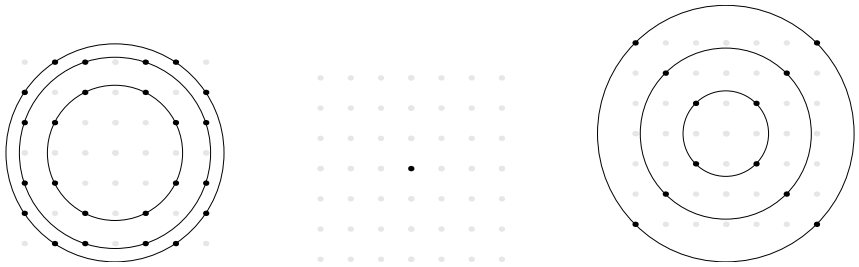


Figure 7.13: All the other orbits (different from the one of Fig. 7.12) for $\mathbb{E}$ under the action of $\mathfrak{D}_2$. Regular orbits (left); singular orbits of the identity element (middle); singular orbits of the elements invariant under the action of $\mathfrak{D}_{(d),1}$, which are the diagonal symmetries (right).

Algorithm 27 Permuted Hadamard product

```
1: procedure HADP(x, u, p_g, y)
2:     for c ∈ C(G) do
3:         for h ∈ G\I_G(c) do
4:             for i ∈ [[0, N(c) − 1]] do
5:                 x_{c,h}[i] += u_{c,gh}[i] * y_{c,h}[i]
6:             end for
7:         end for
8:     end for
9: end procedure
```

Clearly, in Alg. 27, the indirections have been removed, provided that the vectors are well sorted according to the orbits and the group actions. This allows a straightforward vectorization by using SIMD instructions to perform the line 5 of Alg. 27. Our method thus transforms the Hadamard product with indirections of Alg. 25 into a sequence of smaller Hadamard products of varying sizes (depending only on the underlying group structure) with no indirection (see Alg. 27). Actually, the indirections of Alg. 25 are converted in Alg. 27 in the application order of the small Hadamard products.

7.4.4.3 Performance results

We applied such permutation without array indirection to the group appearing in our **M2L** applications in dimension three with the hyperoctahedral group $\mathfrak{D}_d$. The results are reported in Fig. 7.14. We only compared the application timings of the Hadamard product with indirections to the product with permutations (without the indirections). For each of them, we ran in sequential 10000 Hadamard products on different vectors randomly initialized at each application (to prevent the data reuse in the cache memory). Our code is executed on the architecture described in Sect. 7.1.3, that is on a CPU supporting AVX-512 SIMD units. These AVX-512 units can offer an up to 8x speedup on our double precision computations. Notice however that the Hadamard products are memory-bound (see App. D.1.1.1), so the expected gain on such application should be limited. To measure the performance impact of the vectorization (see App. D.1.2.2), we first use as reference scalar timings the performances obtained by disabling the compiler vectorization. We then consider the compiler auto-vectorization that relies only on the C++ code. Recalling that the compiler may not be able to detect that the loop in the Hadamard product of Alg. 25 can be parallelized due to the indirections, which may prevent the auto-vectorization, we also provide the results with an OpenMP vectorized version (using a simple #**pragma omp simd** OpenMP directive). This last version makes the compiler (correctly) assume that the loop is parallelizable and hence try to enforce the vectorization. We may mention that we checked the assembly codes to verify that the right SIMD registers were used as well as FMA instructions.

As shown in Fig. 7.14, the code with indirections is never vectorized using compiler auto-vectorization (with none of the tested compilers presented in Sect. 7.1.3). By adding OpenMP directives, the icpc compiler is able to vectorize the Hadamard product with indirections, and hence offers performance gains, but the g++ compiler cannot. Concerning our method with permutation (and no indirection), the two compilers are able to auto-vectorize the inner loop of Alg. 27. This seems to confirm our hypothesis: indirections prevent the compiler from understanding that the loop can be parallelized. We also tested the OpenMP vectorized version of this inner loop, leading to similar performances. Indeed, this inner loop vectorization is straightforward since our method removes the indirections. Hence, relying on the auto-vectorization with the permuted Hadamard product is sufficient for both compilers to obtain a vectorized code.

Actually, the code with indirections using OpenMP directives and the icpc compiler performs quite well. Below order $L = 12$, the OpenMP vectorized Hadamard product with indirections (Alg. 25) with the icpc compiler is faster than our vectorized method without indirection (Alg. 27). Above this order, our method starts being more efficient and the permuted products (using g++ or icpc) start being as efficient as the OpenMP vectorized products with indirections using icpc. For extremely high orders (see Fig. 7.14), the permuted Hadamard products are always faster. However, in practice, high orders like $L = 10$ are related to very high precisions and lead to high computation times for the FMM. We believe that it is not relevant to consider such higher orders in our application context.

The reasons of this limited efficiency for low orders is that the vectors on which we apply the Hadamard products with no indirection in Alg. 27 are too small. Some of them cannot fully benefit from the (AVX-512) SIMD units. Hence, we discarded the use of permutations without indirections, and we rely in our final code on the Hadamard products with indirections and OpenMP directives, using preferalby the *icpc* compiler. According to Fig. 7.14, this version offers, with respect to the Hadamard products with indirections but without OpenMP (and using *icpc*), performance gains up to $\approx 16\%$.

Another important point to mention is that our permutation algorithm is independent of the application context and can be applied to any group. Moreover, the data localization idea we presented is not limited to the FMM scope.

Remark 7.4.5. *The results given in Fig. 7.14 also show that the icpc compiler combined with OpenMP directives is way better than the g++ one at vectorizing the Hadamard products with indirections. This motivates the use of the icpc compiler (still combined with OpenMP directives)*

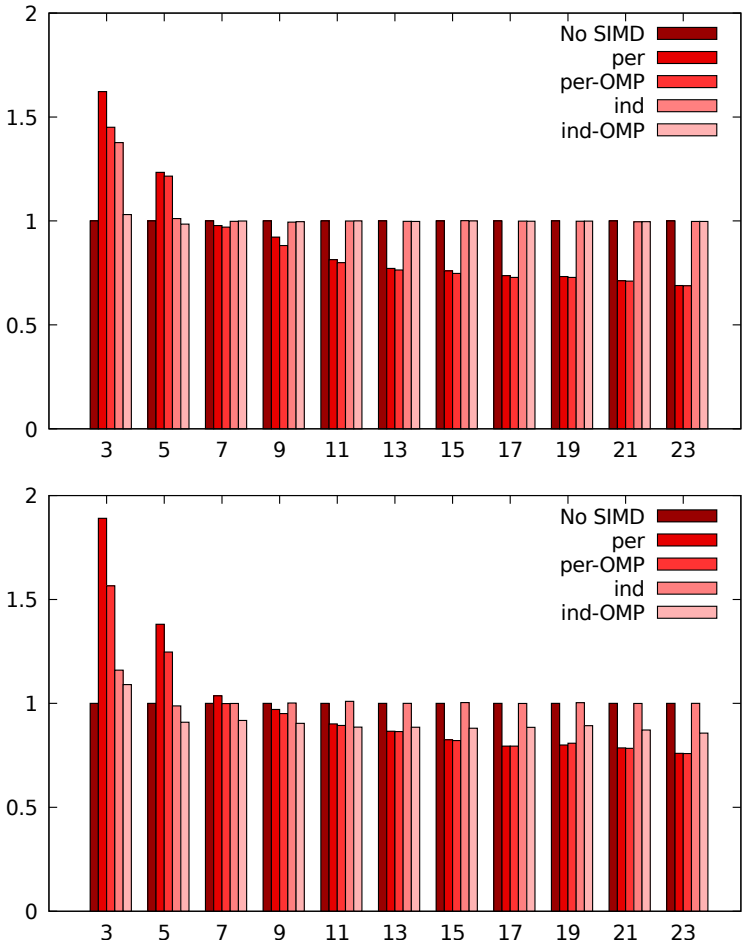


Figure 7.14: Relative timings of the different approaches for the Hadamard products with regard to the one-dimensional interpolation order L (i.e. the vectors are of size L^3) and using different compilers: g++ (top) and icpc (bottom). The timings are all given relatively to the performances obtained when disabling the compiler auto-vectorization ('No SIMD'). 'per': permutation method; 'ind': indirection method. OMP: relying on OpenMP vectorization.

in our tests on defmm.

7.5 Optimizations

In this section, we describe several optimizations we provide in *defmm*. Here is a summary of the new optimizations we present:

- A way of efficiently (pre)computing the **P2M** and **L2P** operators (Sect. 7.5.1)
- A storage strategy for the expansions taking into account the potential non-uniformity of the particle distribution reducing all the accesses to $\mathcal{O}(1)$ operations (Sect. 7.5.2)
- A set of blank tree traversals to limit the precomputation and application costs by minimizing the number of computed directional expansions according to the particle distribution (Sect. 7.5.3)

- A set of fast algorithms for the **M2M** and **L2L** operators based on vector stacking and the particular expression of the matrix form of these operators (Sect. 7.5.4)
- An efficient SIMD vectorized code for the near field in the Helmholtz kernel case and a strategy based on precomputation, BLAS 2 and particle sorting for applications of *defmm* to iterative solvers (Sect. 7.5.5)
- An efficient way of handling the numerous small FFTs involved in *defmm*, i.e. to speed up the **M2F**/**F2L** evaluations (Sect. 7.5.6)
- A refined DTT with **M2P** and **P2L** operators for a better handling of highly non-uniform particle distributions (Sect. 7.5.7)

7.5.1 Leaf operators optimizations

The interpolation-based tools and FFT techniques aim at providing a fast evaluation of most of the FMM operators. However, the positions of the particles having non-predictable values, expansive computations have still to be done on the leaves when applying the **P2M**/**L2P** operators. Their efficient treatment is the purpose of this section.

The **M2M**/**L2L** operators benefit from a complexity reduction thanks to the nature of the multivariate interpolation uniform grids, which are products of the same one-dimensional one (details are provided in Sect. 7.5.4). For the **P2M**/**L2P** operators, the product structure can also be exploited. Let s be a source cell and let $\mathbb{M} \in \mathbb{R}^{L^d \times N(s)}$ be the matrix form of the **P2M** operator on s, $N(s)$ being the number of source particles in s. The same discussion, up to a transposition, can be done for the **L2P** operators on target cells. We have

$$(\mathbb{M})_{k,l} := \mathfrak{L}_k^{(s)}(\mathbf{y}_l)$$

where $\mathfrak{L}_k^{(s)}$ is the Lagrange polynomial associated to the k^{th} nodes in the interpolation grid of s and $\mathbf{y}_l$ is the l^{th} particle in s. Then we have the decomposition $\mathfrak{L}_k^{(s)} = \Pi_{p=1}^{d} S^{(s)}_{\mathfrak{I}(k)_p}\left((\mathbf{y}_l)_p\right)$ thanks to the product grid, $S^{(s)}_{\mathfrak{I}(k)_p}$ being a one-dimensional interpolation polynomial and $\mathfrak{I}$ being as in Def. 7.4.1. Hence, the naive computation of $\mathbb{M}$ requires $\mathcal{O}\left(N(s)L^d d\right)$ one-dimensional polynomial evaluations, each of them having a cost of $\mathcal{O}\left(L\right)$ flops, resulting in an overall cost for the computation of a single **P2M** equal to $\mathcal{O}\left(N(s)L^{d+1}d\right)$. However, precomputing first the d sets $\mathcal{S}_p$ such that

$$\mathcal{S}_p := \{S_q^{(s)}\left((\mathbf{y}_l)_p\right) \mid q \in [\![0, L-1]\!], l \in [\![1, N(s)]\!]\},$$

each entry of $\mathbb{M}$ is the product of one element of each $\mathcal{S}_p$, $p \in [\![1, d]\!]$. The computation of these sets costs $\mathcal{O}\left(dL^2N(s)\right)$ and allows to compute $\mathbb{M}$ in $\mathcal{O}\left(dN(s)L^d\right)$ flops. If the FMM is supposed to be applied many times, the precomputation of the **P2M**/**L2P** matrices can thus offer an interesting performance gain for the evaluation of $\mathbb{M}$. This is the default method in our *defmm* since we are supposed to include it in an iterative solver.

Notice that a similar method, exploiting the tensorized structure of the **P2M** / **L2P** matrices, has been described in [126], but for storage optimization only.

Remark 7.5.1. *The interpolation grid computation has a non-negligible cost and are needed for both the **M2L** and **P2M**/**L2P** precomputations. Hence, because all the grids have the same interpolation order and all the cells are d-cubes (whose radii depend only on the tree level, see Sect. 7.3.1.3), they can be deduced from a single grid by means of scaling. We then precompute this unique interpolation grid. Following the same idea, the bijection $\mathfrak{I}$ (see Def. 7.4.1) is intensively used in our implementation of the precomputation step, so we also precompute the correspondences between indices and multi-indices.*

7.5.2 Expansion storage

In the low-frequency regime, there exists at most one single expansion in each cell. This is no longer valid in the high-frequency domain, where the number of expansions associated to a cell depends on the number of directions used in the directional method. In this section, we explain how these expansions are stored in *defmm* for fast accesses. We also present a global storage format mainly designed to lower the precomputation cost of the global FMM.

7.5.2.1 Directional expansions

In the high-frequency regime, we only store the directional expansions that appear in at least one effective *cell-cell* interaction. These directional expansions are called *effective* directional expansions. This strongly reduces the number of stored expansions for non-uniform distributions. Indeed, the maximal number of possible directional expansions in each cell corresponding to the E^{th} direction tree level (see Sect. 4.2.2.1) is of order $\mathcal{O}\left(2^{E(d-1)}\right)$ while only a few effective directional expansions may exist, depending on the distribution.

Remark 7.5.2. *In terms of memory footprint and upward/downward pass costs, the uniform distribution case is the worst one in the high-frequency regime due to the important number of effective directional expansions in most cells. On the other hand, elongated geometries (such as ellipsoids) fully benefit from the storage of only the effective directional expansions.*

The directions, for a given cell size, are uniquely indexed using a *global direction index* defined in accordance with to the elements of the corresponding direction tree level. For any cell, we thus want to be able to recover in $\mathcal{O}(1)$ operations the effective directional expansion corresponding to a given direction index. In the same time, we also want to efficiently enumerate the directions involved in a given cell in the **M2M**/**L2L** operators, i.e. without having to test all possible directions. These two points are achieved by associating an array and a list to each cell in the high-frequency regime. The array has a size equal to the number of effective directional expansions for this cell c and its k^{th} entry is an integer l which is non-negative if and only if the directional expansion with direction k is the l^{th} expansion stored for c. This array is denoted by dir. The list, denoted by rid has a size equal to the number of directional expansions stored in c and associates to each direction kept in c its global direction index. Hence, we have

$$\begin{cases} rid[dir[k]] = k \text{ if the direction } k \text{ is stored in } c; \\ dir[k] < 0 \text{ otherwise.} \end{cases}$$

The directional expansions used in a cell are stored in the same single array whose size is equal to the number of positive elements in its rid array. This allows to compute the starting addresses of the different directional expansions of a cell in the high-frequency regime using only a single pointer and the number of directions kept in this cell. Hence, we can use the same cell structure in both the low- and high-frequency regime thanks to the reduction of directional expansions to store. It is important to also discuss the way these arrays are allocated.

7.5.2.2 Global expansion arrays

The multipole and local expansions (and their conversions in the Fourier domain) are arrays of complex numbers and were originally allocated independently in each cell and stored in non-contiguous memory areas. However, this approach prohibited the design of certain batch algorithms (see Sect. 7.5.6.1) and generated numerous allocations, with an impact on the precomputation cost. We then decided to group the expansions of all cells of a given tree into one single global array instead of individual arrays for each cell. Two consecutive cells in the cell array (i.e. the array storing the cells of a given tree, see Sect. 7.3.4) are ensured with our construction to have consecutive expansions in the corresponding global array. Because both the multipole/local expansions and their conversion

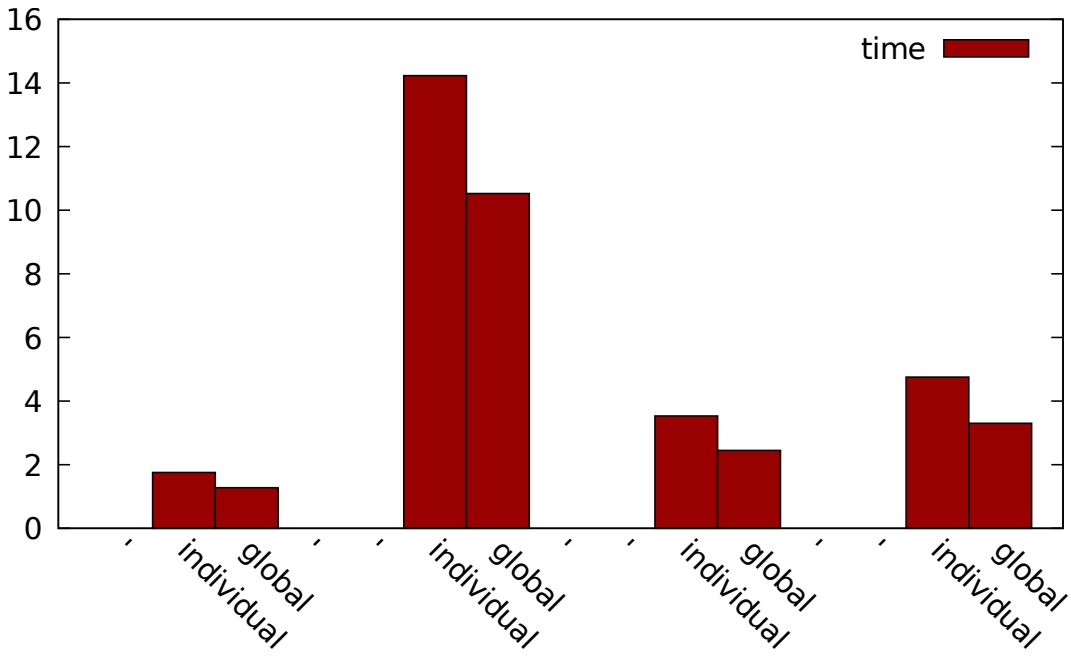


Figure 7.15: Timings (in seconds) of the allocation and zeroing of expansions for a one-dimensional interpolation order equal to 4 without global arrays (*individual*) or with global arrays (*global*). From left to right, denoting by K the geometry length multiplied by the wavenumber: cube $K = 0$, cube $K = 64$, sphere $K = 0$, sphere $K = 64$, with $10M$ particles each, see Sect. 7.1.3.

into the Fourier domain are stored, we decided to declare one global array for each of the four types of expansion.

The comparative timings for the allocation and initialization of the expansions depending on the particle distribution with or without global arrays are reported in Fig. 7.15.

This modification of the storage results in a substantial time reduction (up to 32%) of the expansion allocations and zeroing with all the tested distributions. The only needed informations to initialize the global arrays are the number of cells in a FMM 2^d-tree, this being computed explicitly during the tree construction (see Sect. 7.3.4), and the number of effective directional expansions in each cell (see Sect. 7.5.2.1). This last element is computed during the blank horizontal pass (see Sect. 7.5.3).

There is however a drawback with this approach in the case of moving particles, modifying the structure of the FMM 2^d-trees at each application. In this case, the number of effective directional expansions in each cell may change, modifying the size of the global expansion arrays. Nevertheless, in our application case the effective directional expansions do not change between two FMM applications.

7.5.3 Blank passes

As presented in Sect. 7.4.3.2, the size of the **M2L** tables can be reduced by "quotienting them" by the rotations of the d-cube. However, in the high-frequency regime, because of the increasing size of these tables and because only a small number of **M2L** matrices may be used with non-uniform particle distributions, we precompute only the **M2L** (diagonal) matrices required by the horizontal pass.

7.5.3.1 Blank horizontal pass

To find the **M2L** matrices that are effectively used in the horizontal pass, we perform a "blank" DTT during the precomputation step. To be more precise, by applying such a blank DTT, each time a far field *cell-cell* interaction in the high-frequency regime should be realized with a relative index in the corresponding **M2L** table associated to an element not already precomputed, the blank DTT calls a routine that precomputes this **M2L** matrix. The algorithm only traverses the high-frequency levels.

Because the symmetries presented in Sect. 7.4.1 also apply for the **M2L** matrices in the high-frequency regime, we precompute such a matrix if and only if no other rotation of it has already been precomputed. However, these **M2L** matrices are computed independently of the direction of the *cell-cell* interaction. This information on the direction is needed to recover the directional multipole and local expansions involved in the interaction. This corresponds to the *best cone index* search for each *cell-cell* interaction.

7.5.3.2 Best cone index

When applying the blank DTT, the relative positions of the interacting cells are computed, meaning that the *best* theoretical direction u for this interaction is known. This *best* direction corresponds to the optimal theoretical interaction direction between the two cells, i.e. the normalized center difference. Let $\mathcal{D}(E)$ be the set of directions at level E, E being the interacting cell level. We thus find $\tilde{u} \in \mathcal{D}(E)$ such that

$$\tilde{u} = \underset{v \in \mathcal{D}(E)}{argmin} |u - v|. \tag{7.7}$$

Notice that such a $\tilde{u}$ may not be unique. We obtain $\tilde{u}$ by using an exhaustive and naive search in $\mathcal{D}(E)$, keeping the first direction minimizing Eq. 7.7 during the blank DTT. Then, we mark the source and target interacting cells with the index of $\tilde{u}$ in $\mathcal{D}(E)$. This corresponds to an update of the list *rid* presented in Sect. 7.5.2.1. In the same time, we mark the **M2L** precomputed matrix with this $\tilde{u}$. If a permutation of this matrix has already been computed, we store $\tilde{u}$ in an array storing these indices for all the permutations. This allows to use the same precomputed **M2L** matrix in the high frequency regime for each *cell-cell* interaction that involves a permutation of this matrix while recovering the global direction indices of the directional expansions in $\mathcal{O}(1)$ each time (a given permutation of this matrix being associated to a single global direction index). This means that during the FMM application, the high-frequency **M2L** operators only execute Hadamard products as in low-frequency regime.

At the end of the blank DTT, each cell knows all the directions this cell is interacting with. The set of effective directions (see Sect. 7.5.2.1) found by this process is not sufficient to run the overall directional FMM scheme. Indeed, the effective directional multipole expansions of the ancestors of a cell c are assembled from particular directional expansions in c which could be not among its effective directions. Hence, we have to handle these missing directions by vertically communicating this information in the source and target 2^d-trees.

7.5.3.3 Blank downward pass

To complete the set of all required directions involved in a computation for a given cell, we added two other phases in the precomputation step. This consists in two blank downward passes, one on the source and one on the target 2^d-trees, transmitting the information of the needed directions from the root to the leaves. This can be performed efficiently by correctly indexing the directions of $\mathcal{D}(E)$ with regard to $\mathcal{D}(E+1)$. Each $v \in \mathcal{D}(E)$ having an ancestor in $\mathcal{D}(E+1)$ and each $w \in \mathcal{D}(E+1)$ having 2^{d-1} possible sons, if k denotes the index of v in $\mathcal{D}(E)$, thus the index of the ancestor of v in $\mathcal{D}(E+1)$ is supposed to be $\lfloor k/2^{d-1} \rfloor$. Once again, the algorithm is only applied on the cells in the high-frequency regime.

After the application of all these blank passes, the entire and minimal[3] set of required directional expansions in each cell to perform the directional FMM is known.

7.5.4 High frequency M2M and L2L

Because of the directional aspects of *defmm*, the cost of the upward and downward pass is linearithmic for surface meshes (see [94]). As opposed to the FMM algorithms for non-oscillatory kernels, these steps may not be so much faster to compute than the horizontal pass. There is thus a need for carefully optimizing their application. We first rely on the algorithm used in [8] for the treatment of the **M2M** and **L2L** operators exploiting the tensorized structure in the case of non-oscillatory kernels (non-directional case).

As described in 7.5.4.1, we extend the idea presented in [8] to equispaced grids and to oscillatory kernels, using vector stacking (see Sect. 7.5.4.2) and deinterleaved real and imaginary parts of input vectors (see Sect. 7.5.4.3). We also investigate new other approaches based on the properties of the Lagrange interpolation on equispaced grids (see Sect. 7.5.4.5) and on grouping **M2M** /**L2L** operators (see Sect. 7.5.4.6).

7.5.4.1 Original algorithm

We first describe the tools needed to express a general version of the algorithm suggested in [8]. The basic idea is that if the interpolation grid in a cell is the product of the same one-dimensional grid, the matrix representation of the **M2M** and **L2L** operators are tensorized matrices. In other words, let $\mathbb{M} \in \mathbb{R}^{L^d \times L^d}$ be the matrix representation of a **M2M** there exists $M^{(p)} \in \mathbb{R}^{L \times L}$, $p \in [\![1, d]\!]$, such that $\mathbb{M} = \bigotimes_{p=1}^{d} M^{(p)}$. For any $\mathbf{v} \in \mathbb{C}^{L^d}$, we have

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

where the sum on the left is a line of the result of a matrix-vector product of size $L \times L$ and with $\mathfrak{I}$ as in Def. 7.4.1. With i varying, this matrix-vector product is performed to L^{d-1} different restrictions of $\mathbf{v}$ but involving each time the *same* matrix. Hence, there exists a permutation $P \in \mathbb{R}^{L^d \times L^d}$ such that

$$
(\mathbb{M}\mathbf{v})_i = \sum_{\mathbf{J} \in Im(\mathfrak{I}) \mid J_d = 0} \prod_{p=1}^{d-1} M^{(p)}_{\mathfrak{I}(i)_p, J_p} \left(diag(M^{(d)}) P \mathbf{v} \right)_{\mathfrak{I}(i)_d}
$$

and 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. Because the same permutation can be used each time, $\mathbb{M}$ can actually be written as

$$
\mathbb{M} = \left(diag(M) P \right)^d .
$$

[3] According to a fixed direction tree.

This method reduces the application cost of a **M2M** from $\mathcal{O}\left(L^{2d}\right)$ to $\mathcal{O}\left(dL^{d+1}\right)$ but can be further improved from a computational viewpoint. Because the same matrix is applied to L^{d-1} vectors, these vectors can be stacked in order to perform matrix-matrix operations (see App. D.2), hence benefiting from the BLAS 3 compute efficiency [8].

Remark 7.5.3. *The **L2L** case is obtained similarly because the corresponding matrix representation $\mathbb{L}$ is such that $\mathbb{L} = \mathbb{M}^T$. We will not detail the **L2L** in the rest of this section.*

Remark 7.5.4. *Each time a block diagonal matrix with all blocks equals is applied to a vector numerically, this vector stacking is possible and allows to perform the matrix-vector product as a matrix-matrix one.*

We will refer to this version as the *tensorized* method. In [8], this tensorized method is presented for tensorized Chebyshev interpolation rules and non-oscillatory kernels. The extension of this method to the equispaced grids of *defmm* is straightforward: these are still tensorized grids. The directional tensorized method is simply obtained by remarking that the directional **M2M** matrix $\mathbb{M}(u)$ with direction u can be written

$$\mathbb{M}(u) = D_{left}(u) \left(\bigotimes_{p=1}^{d} M^{(p)} \right) D_{right}(u) \tag{7.8}$$

with two diagonal matrices $D_{left}(u)$ and $D_{right}(u)$ (see Sect. 4.2.2.4).

7.5.4.2 Directional operators and directional stacking

In the high-frequency regime, an optimization can be obtained using the similarity between the directional **M2M** matrices. Starting from Eq. 7.8, for a given cell, if we consider two directions u and u' with corresponding directional multipole expansions $\mathbf{v}(u)$ and $\mathbf{v}(u')$, we obtain

$$\begin{aligned}
\begin{bmatrix} \mathbb{M}(u)\mathbf{v}(u) \\ \mathbb{M}(u')\mathbf{v}(u') \end{bmatrix} &= \begin{bmatrix} D_{left}(u) \left(\bigotimes_{p=1}^{d} M^{(p)} \right) D_{right}(u)\mathbf{v}(u) \\ D_{left}(u') \left(\bigotimes_{p=1}^{d} M^{(p)} \right) D_{right}(u')\mathbf{v}(u') \end{bmatrix} \\
&= \begin{bmatrix} D_{left}(u) & \\ & D_{left}(u') \end{bmatrix} \begin{bmatrix} \bigotimes_{p=1}^{d} M^{(p)} & \\ & \bigotimes_{p=1}^{d} M^{(p)} \end{bmatrix} \begin{bmatrix} D_{right}(u)\mathbf{v}(u) & \\ & D_{right}(u')\mathbf{v}(u') \end{bmatrix} \\
&= \begin{bmatrix} D_{left}(u) & \\ & D_{left}(u') \end{bmatrix} diag \left(\bigotimes_{p=1}^{d} M^{(p)} \right) \begin{bmatrix} D_{right}(u)\mathbf{v}(u) & \\ & D_{right}(u')\mathbf{v}(u') \end{bmatrix}.
\end{aligned}$$

According to Rem. 7.5.4, the results of the multiplications of the directional expansions by the right diagonal matrices can be stacked in order to get matrix-matrix products. Combined with the optimization of Sect. 7.5.4.1, this leads to larger matrix-matrix products than with the tensorized method when applying $\bigotimes_{p=1}^{d} M^{(p)}$, which may further benefit from the BLAS 3 efficiency. Thanks to the blank passes (see Sect. 7.5.3), the number of directional expansions stored in a cell is minimal with regard to the direction tree and is known when the high-frequency **M2M** operator is applied.

Remark 7.5.5. *The ideas discussed in this section are also used in order to obtain matrix-matrix products in the high-frequency **P2M**/**L2P** instead of matrix-vector ones, hence exploiting the BLAS 3 efficiency.*

Remark 7.5.6. *The evaluations of the complex exponentials needed in the computation of the matrices D_{left} and D_{right} are costly. Denoting by $N_{dir}(E)$ the number of directions at the level E of the 2^d-trees, there are $N_{dir}(E)$ such matrices. We thus precompute these diagonal matrices for all*

*possible directions and for all levels. Notice that the diagonal matrices used in the **L2L** are obtained using complex conjugate of the precomputed diagonal matrices of the **M2M**. This conjugate trick can also be exploited in order to reduce the number of precomputed matrices D_{right} and D_{right} (see Eq. 7.8), provided that there exists a polar symmetry in the set of directions at any level (i.e. $-u$ is a direction at the same level than u for any direction u at any level). We do not use this last symmetry on the directions in our implementation.*

This method is referred to as the *tensorized+stacking* method. The potential gain with this increase of the vector stacking size depends on the particle distribution and the wavenumber since it is directly related to the number of expansions stored in each cell. We now present a method that further increases the size of the vector stacking while modifying the type of the matrix-matrix product, with no dependency on the particle distribution or on the frequency regime (i.e. also optimizing the low-frequency **M2M** / **L2L** operators).

7.5.4.3 Real/complex matrix-matrix products

We propose another level of vector stacking for the **M2M** (or **L2L**) application, exploiting the space of the matrices $M^{(p)}$. These matrices are real (since they are composed of evaluations of Lagrange polynomials), but are applied to complex vectors in a directional method. For any $\mathbf{v} \in \mathbb{C}^{L^d}$, we then have

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

In consequence, by deinterleaving (see Fig. 7.16) the real and imaginary parts in the expansion vectors, we can obtain a stacked matrix of real elements with twice the number of entries of the matrices obtained with the stacking method but with no additional storage. This transforms the complex matrix-matrix products of the tensorized algorithm and of the stacking method into larger (in terms of number of real entries compared to number of complex entries) real ones. In the same way, by discarding the imaginary part of the (real) **M2M** / **L2L** matrices, we reduce the number of effective operations performed by these matrix-matrix products. Once again, relying on the BLAS 3 efficiency, we expect this optimization to further accelerate the **M2M** and **L2L** application. This method is referred to as the *tensorized+stacking+real* method. The real method can be actually seen as a *matrix stacking*. Notice that in the low-frequency regime, since there is only one multipole/local expansion per cell, no stacking of the directional expansion is done and the tensorized+stacking+real method is simply a tensorized+real one.

7.5.4.4 Performance results

In Figs. 7.17 and 7.18 are detailed the different **M2M** timings for the variants we described on the two test cases in Sect. 7.1.3.

On the sphere, according to Fig. 7.17, the most efficient variant is clearly the last one we proposed, i.e. the tensorized+stacking+real method. On this test case, the tensorized+stacking and tensorized+stacking+real methods performs better than the tensorized method of [8]. The tensorized+stacking method, exploiting the multiple directional expansions of a given cell, can only be better than the tensorized method in the high-frequency regime. However, the tensorized+stacking+real method also positively impacts the performances of the **M2M** / **L2L** operators in the low-frequency regime. On the sphere, the tensorized+stacking+real method performs up to ≈ 2.16 times faster than the tensorized one.

Conclusions are different on the cube test case, following the results of Fig. 7.18. In Fig. 7.18 (top), using the same wavenumber than on Fig. 7.17, all the tree levels are in the high-frequency regime, so only high-frequency **M2M** operators are performed. In this case, the tensorized+stacking method performs better than the tensorized one. Indeed, the cube test case being an uniform

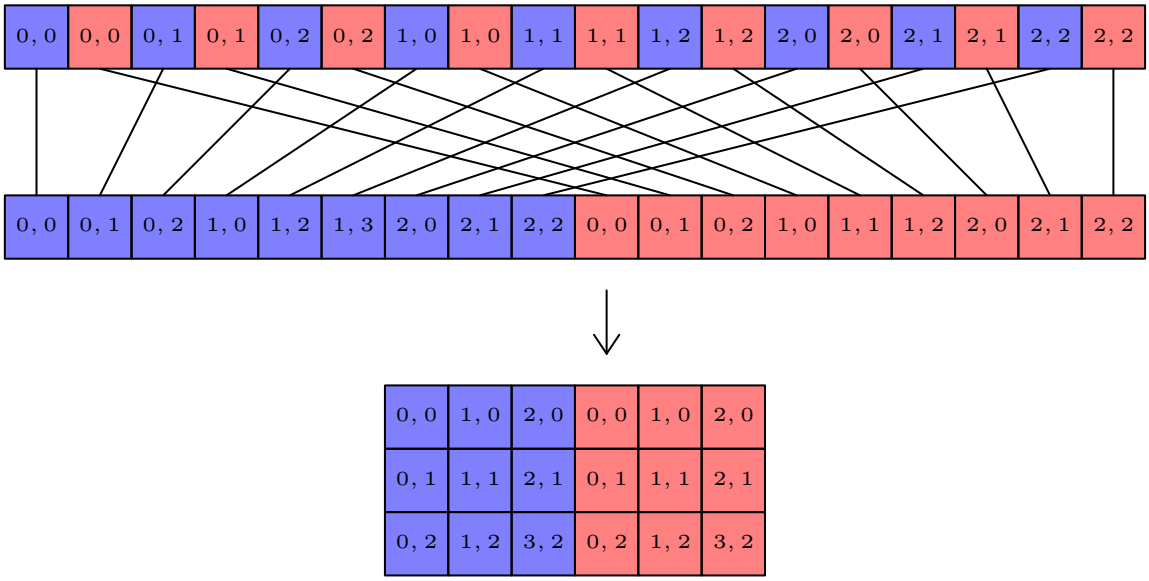


Figure 7.16: Blue: real part of a complex number. Red: imaginary part of a complex number. Top: input **M2M** / **L2L** vector before deinterleaving of real and imaginary parts. Middle: Deinterleaving process. Down: same data than in the middle, but seen as a column-major matrix. The input vector corresponds to a multipole/local expansion in 2D with one-dimensional order equal to 3. The indices in the arrays entries corresponds to the multi-indices of the corresponding interpolation nodes in the grid.

distribution, many cells have the maximal number of effective directions, which maximizes the size of the vector stackings. For an interpolation order equal to 3, the tensorized+stacking+real method performs even faster ($\approx$ 4 times faster than the tensorized method). However, in this test case, the tensorized+stacking+real method is less efficient than the tensorized+stacking one for larger interpolation orders. Actually, since one of the two matrices in the matrix-matrix products is very small (of size $L \times L$, L being the one-dimensional interpolation order), the maximal efficiency of the vector stacking is obtained for relatively small sizes for the second matrix. Hence, the cost reduction obtained by the additional vector stacking of the tensorized+stacking+real method and the lower number of operations needed on real matrix-matrix products are not sufficient to balance the interleaving cost in the uniform cube test case in the high-frequency regime. Nevertheless, in the low-frequency regime, the deinterleaving still allows faster **M2M** evaluations than the tensorized method of [8] (recalling that there is no stacking on directional expansions in the low-frequency regime). In practice, we only observed this performance drop of the tensorized+stacking+real method compared to the tensorized+stacking one on the uniform cube, so we keep the tensorized+stacking+real method by default in our implementation.

The same gains are obtained on the **L2L** operators.

7.5.4.5 Sparse M2M and L2L

When equispaced grids are used for the interpolation, the **M2M** matrix has many zeros. This particular form is obtained if the same interpolation order is used in all the source tree cells, which is the case in our implementation. Hence, some of the interpolation nodes of the sons of a cell c are also interpolation nodes of c. Because the Lagrange polynomials are equal to 0 or 1 on such points, this reduces the theoretical number of operations to be performed in order to recover the result of a **M2M** application. However, the induced sparsification of the full **M2M** matrix in itself is not sufficient to obtain a competitive methods with the schemes presented in Sect. 7.5.4.1, 7.5.4.2 and

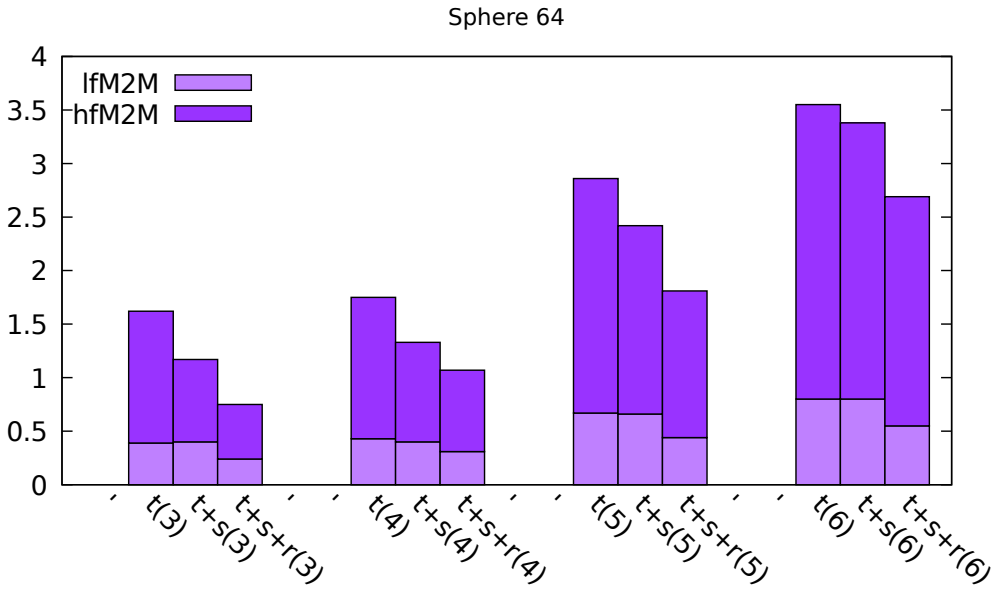


Figure 7.17: Cumulative times (seconds) spent in all **M2M** operators for the different **M2M** variants on the sphere with $\kappa a = 64$, where κ denotes the wavenumber and a the sphere diameter. t: tensorized; t+s: tensorized+stacking; t+s+r: tensorized+stacking+real. From left to right, the groups correspond to the one-dimensional interpolation orders equal to 3,4,5,6. The low-frequency **M2M** operation (pale) are separated from the high-frequency ones (dark).

7.5.4.3.

One can still combine the tensorized method with the knowledge about the zeros and ones in the small matrices involved (i.e. the matrices of the tensor product). For instance, if we consider the matrix $M^{(1)}$ corresponding to Fig. 7.19, we have

$$M^{(1)} = \begin{bmatrix} 1 & \star & 0 & \star & 0 \\ 0 & \star & 1 & \star & 0 \\ 0 & \star & 0 & \star & 1 \\ 0 & \star & 0 & \star & 0 \\ 0 & \star & 0 & \star & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \star & \star \\ 0 & 1 & 0 & \star & \star \\ 0 & 0 & 1 & \star & \star \\ 0 & 0 & 0 & \star & \star \\ 0 & 0 & 0 & \star & \star \end{bmatrix} P$$

where $\star$ denotes any non-zero real element and P is a well chosen permutation matrix. There exist thus $B_0 \in \mathbb{R}^{L \times \lfloor \frac{L}{2} \rfloor}$ and $B_1 \in \mathbb{R}^{\lfloor \frac{L}{2} \rfloor \times \lfloor \frac{L}{2} \rfloor}$ such that

$$M^{(1)} = \begin{bmatrix} Id & B_0 \\ 0 & B_1 \end{bmatrix} P.$$

This gives

$$\begin{bmatrix} Id & B_0 \\ 0 & B_1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_0 + B_0 \mathbf{v}_1 \\ B_1 \mathbf{v}_1 \end{bmatrix}.$$

This means that the in-place product by $M^{(1)}$ can be reduced to a permutation followed by two BLAS calls on B_0 and B_1, where the result of the first call (on B_0) is added to $\mathbf{v}_0$. The overall com-

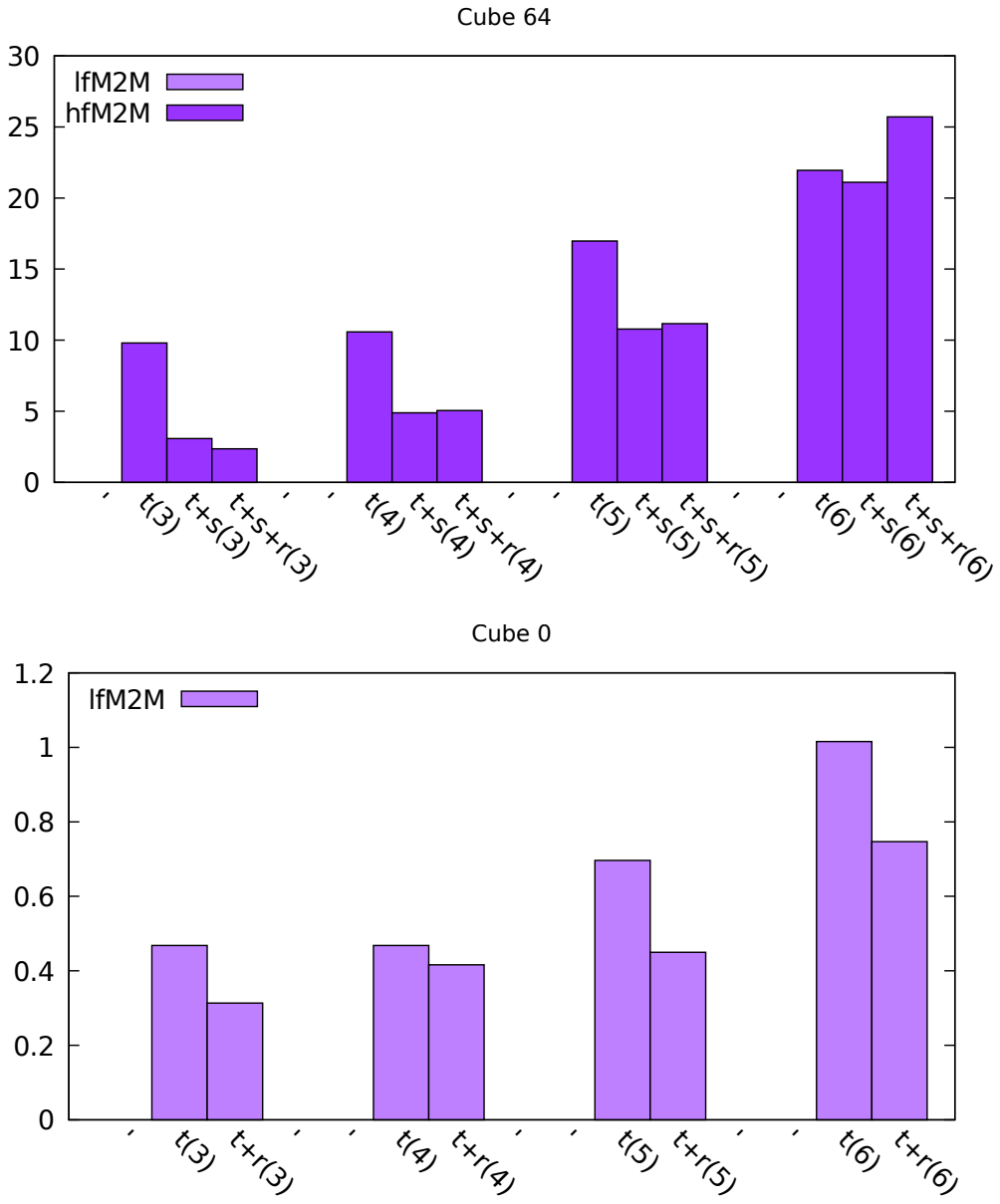


Figure 7.18: Cumulative times (seconds) spent in all **M2M** operators for the different **M2M** variants on the cube with $\kappa a = 64$ (top) and $\kappa a = 0$ (bottom), where κ denotes the wavenumber and a the sphere diameter. t: tensorized; t+s: tensorized+stacking; t+s+r: tensorized+stacking+real. From left to right, the groups correspond to the one-dimensional interpolation orders equal to 3,4,5,6. The low-frequency **M2M** operation (pale) are separated from the high-frequency ones (dark).

Figure 7.19: Interpolation order equal to 5. Parent grid on $[-1,1]$ (red) and son grid on $[-1,0]$ (blue).

plexity then drops from $\mathcal{O}\left(dL^{d+1}\right)$ to $\mathcal{O}\left(d\left(\lfloor\frac{L}{2}\rfloor\right)^{d+1}\right)$. This method is called the *sparse* method. However, we need two extra arrays to call the BLAS on different matrices and the new permutation exchange the data in these arrays. In addition, the matrix-matrix product sizes are highly varying: the left matrices are very small compared to the right one. Two BLAS calls on these sizes (for instance 3×2 and 2×2 for $L=5$) could be less efficient than a single call on larger matrices. We also wrote this algorithm without BLAS call, but with no computational gain.

In our tests, the sparse method leaded to less efficient **M2M** evaluations than the tensorized+stacking+real method (even for large interpolation orders). Thus, we did not used this sparse method in practice.

7.5.4.6 Son symmetries

One may consider a last optimization which is mainly useful for uniform distributions. This last method consists in looking at the possible different $M^{(p)}$ for cells on the same levels. Indeed, there are only two such matrices, that is there exist $M_0 \in \mathbb{R}^{L\times L}$ and $M_1 \in \mathbb{R}^{L\times L}$ such that $M^{(p)} \in \{M_0, M_1\}$, $\forall p \in [\![1,d]\!]$. Because of the symmetry of the one-dimensional interpolation grid with regard to zero, there exists a permutation matrix P such that $M_1 = P^T M_0 P$. This information can be used to obtained larger matrix products than with the tensorized+stacking+real method, only introducing additional permutations and assuming that the cell has more than one son. A complexity reduction compared to the tensorized algorithm and the tensorized+stacking+real method can be obtained by looking at the common matrices in the decomposition $\mathbb{M} = \otimes_{p=1}^{d} M^{(p)}$ of Sect. 7.5.4.1.

A cell has at most 2^d sons that can be identified using d bits. We suppose that all these sons exist. Each son is uniquely indexed in $[\![0, 2^d-1]\!]$, also using d bits. These bits can be interpreted as the indices of the matrices M_k that are applied to the corresponding multipole expansion in the same order than their apparition order in the bit representation of the index of the son. Hence, after the application of the first of these matrices, the same sequence of matrices is applied to groups of two partial results. The results of these products are added at the end of the tensorized method application. However, the sum can be done after the first product by a M_k, reducing the number of remaining expansions for the next product with a M_k by 2. This process can be repeated d times, leading to a $\mathcal{O}\left(L^{d+1}\sum_{p=1}^{d} 2^{d-p-1}\right) = \mathcal{O}\left(2(2^d-1)L^{d+1}\right)$ complexity if a cell has 2^d sons instead of $\mathcal{O}\left(d2^d L^{d+1}\right)$.

We did not implement this variant since we target highly non-uniform distributions on which the gain of this optimization should be limited. In addition, because we do not store the empty cells in *defmm* (*Ncrit* criterion), we have to test the relative positions of the sons of a cell to recover their index on a perfect tree. Finally, the complexity reduction is limited in our applications since we are interested in three-dimensional cases, the complexity reduction appearing mainly for $d > 3$.

7.5.5 Near field computation

In this section, we describe a set of optimizations used for the fast evaluation of the near field (i.e. the fast evaluation of the **P2P** operators). In Sect. 7.5.5.1 is presented the way the **P2P** operators are grouped in *defmm*, thanks to the DTT. In Sect. 7.5.5.2, an efficient SIMD vectorized code for the Helmholtz kernel is described, allowing to fully benefit from modern SIMD units. Finally, in Sect. 7.5.5.3, we present a precomputation strategy, relying on the static particle distributions of our application context.

7.5.5.1 Grouping P2P

As we explained in Sect. 7.3.3, we proposed a formulation of the DTT in Alg. 21 that switches to *particle-particle* interactions if at least one of the two tested cells is a leaf, as opposed to the implementation of [3], allowing to handle the constraint on the *cell-cell* interactions on the same level. Indeed, by considering a single **P2P** call instead of multiple ones, we benefit more from the $\mathcal{O}(N)$ arithmetic intensity (see App. D.1.1.1) of the near field computation (we will detail this in Sect. 7.5.5.2), hence obtaining better applications timings than with multiple calls. To be efficient, this needs the particles of the descendant of a given cell to be stored in consecutives positions of a global array, which is ensured by our tree construction, thanks to the particle sorting (see Sect. 7.3.4).

7.5.5.2 SIMD direct evaluation

The general form of the **P2P** operator, i.e. of the near field computation, is provided in Alg. 28. There mainly exist two approaches to vectorize the near field direct computation: by vectorizing either the outer loop over the target particles or the inner loop over the source particles of Alg. 28. Basically, a vectorization of the outer loop sums over the source particles for multiple target particles and a vectorization of the inner loop consists in summing the contributions of different source particles into variables corresponding to the same target particle. By vectorizing the outer loop, one avoids the additional reduction operations needed when vectorizing the inner loop.

Algorithm 28 NEARFIELD

```
1: // Input : cell t, cell s
2: // Output: ∅
3: procedure NEARFIELD(t, s)
4:     for x ∈ t ∩ X do
5:         for y ∈ s ∩ Y do
6:             p(x) += G(x, y)q(y)
7:         end for
8:     end for
9: end procedure
```

The SIMD treatment (see App. D.1) of the near field part has been studied in the context of Laplace 3D FMM in many references (see [25, 63, 150, 220, 222]). Notice that, because the source tree may differ from the target one in *defmm*, we are not interested in *mutual interactions*, that is in specifically optimizing the case in which the same sets of particles are both targets and sources.

In [143], a code allowing SIMD auto-vectorization for the Helmholtz **P2P** operator is described. In this code, the outer loop of Alg. 28 is vectorized and a deinterleaving strategy of the real and imaginary parts of the complex numbers is exploited. This deinterleave strategy consists in deinterleaving the data in the registers and performing only real operations. For a given target particle, the interleaving of real and imaginary parts is done once all the source particle contributions have been added. Hence, inspired by the source code provided in [143], we wrote our SIMD vectorized code, whose best version appears to be slightly different. First, we help the compiler to vectorize the outer loop (in practice using the OpenMP # **pragma omp simd** compiler directive). We could have written our SIMD code by hand but the ease of implementation and the portability for various SIMD architectures of the OpenMP vectorization leaded us to implement our near field evaluation using these compiler directives (this is also one of the motivations of the auto-vectorization in exploited in [143]). Second, because we do not have integral over crossing elements but only particles in our code, the singularity treatment consists in checking if two particles have equal positions and to skip the interaction in this case (see Sect. 2.1). We do not want this test to affect the SIMD vectorization, so we compute the norm of the difference of particle positions in

```
1     void simdDeinterleaveP2P<3>(cell<3>& t, cell<3>& s){
2       int N = t.nprt; // Number of target particles
3       int M = s.nprt; // Number of source particles
4       flt* pt;
5       flt* ps;
6       flt  pi4m1 = 1./pi4;
7       flt x, y, z, R, K, r, co, si, pr, pi, qr, qi;
8       cplx *pp = global::p+t.ind; // Where to add the near field contribution
9       cplx *qq = global::q+s.ind; // Charges locations
10    #pragma omp simd
11      for(int i = 0; i < N; i++){
12        pt = K_(t.prt[i].pos);
13        pr = 0.;
14        pi = 0.;
15        for(int j = 0; j < M; j++){
16          ps     = K_(s.prt[j].pos);
17          qr     = qq[j].real();   // Deinterleave data
18          qi     = qq[j].imag();
19          x      = pt[0] - ps[0];
20          y      = pt[1] - ps[1];
21          z      = pt[2] - ps[2];
22          R      = x*x+y*y+z*z;
23          K      = 1./sqrt(R);
24          if(R < 1.e-16){K = 0.;}
25          r      = K * R;
26          r     *= global::kappa;
27          K     *= pi4m1;
28          co     = cos(r);
29          si     = sin(r);
30          co    *= K;
31          si    *= K;
32          pr    += co*qr - si*qi;
33          pi    += co*qi + si*qr;
34        }
35        pp[i] += cplx(pr,pi); // Interleave data
36      }
37    }
```

Figure 7.20: C++ code for the near field using deinterleave of complex numbers and OpenMP SIMD vectorization. *flt* is an alias for *double* and *cplx* is an alias for $std :: complex < double >$.

all cases (singular or not). When such a distance is equal to zero, a variable containing the inverse distance is set to zero (line 24 of Fig. 7.20). Doing so, we keep a minimal instruction in the test that may be processed with masks in the vectorized code. Hence, because all the computations involving singular pairs are multiplied by zero, all the singular interactions generate the same operations than the non-singular ones, but adding a zero contribution. We also tried the trick with the "max" instruction used in [152] (in a GPU context) but to no avail. Finally, we used slightly different calls to mathematic functions compared to the code provided in [143].

It is important to notice that an Array-of-Structure (AoS) format for complex arrays is needed for BLAS routines on complex data and for the FFTs with FFTW, so *defmm* uses AoS for both the arrays of complex data and the arrays of particles. This means that an array of complex numbers (more generally of a given structure) is declared with interleaved real/imaginary parts of the elements stored in this array This has to be opposed to the Structure-of-Array (SoA) format where one array is used for the real parts and another for the imaginary one (more generally, an array is used for each field of the structures). The SIMD implementations usually benefit from the SoA format due to the aligned vector loads. Using AVX-512, efficient gather instructions can limit the AoS format loss of efficiency compared to the SoA one (see [143]).

Results for our implementation and on comparison with a naive implementation (no deinter-

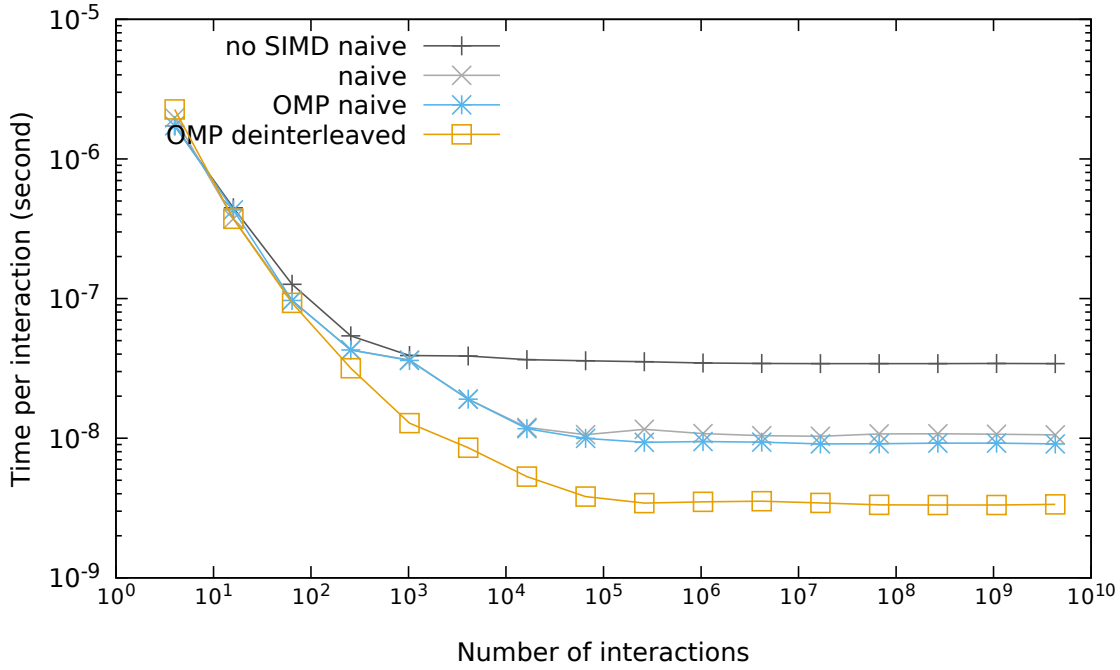


Figure 7.21: Timings (in seconds and divided by the number of computed interactions) for near field computation with regard to the number of computed interactions using the icpc compiler. The same amount of target and source particles are considered each time. "naive" refers to a naive implementation of the direct interactions (with complex numbers and complex arithmetic operations) and "deinterleaved" refers to the code in Fig. 7.20. We prohibit the compiler auto-vectorization for the variant "no SIMD" and we use OpenMP directives on the outer loop for the variant "OMP". With no indication, we rely on the compiler auto-vectorization.

leaving of data) are provided in Fig. 7.21. We used the icpc compiler to get these results. We also compiled our code using the g++ compiler, but it was not able to generate SIMD vectorized code, neither relying on auto-vectorization nor on OpenMP directives. The icpc compiler vectorizes the inner loop if no OpenMP directive is given and is able to vectorize the outer one if an OpenMP directive is provided. We thus need this OpenMP directive in practice to obtain the vectorization we are targeting. According to Fig. 7.21, without considering complex number deinterleaving, the vectorization of the outer loop of Alg. 28 (using OpenMP) leads to slightly better performances than the compiler auto-vectorization of the inner one.

We then emphasize that, except on very small sizes for which the number of target particles is too small to fully exploit the SIMD vectorization, the OpenMP vectorized code in Fig. 7.20 (i.e. the deinterleaved version) clearly provides the best performance results. We believe that the compiler struggles to vectorize the complex arithmetic operations, but manages to vectorize all real ones. The vectorization of this code using OpenMP is 7.6 times more efficient than the sequential version for a **P2P** with 512 source particles and 512 target particles. Since we used *double precision* arithmetic on AVX-512 CPU, we can expect the acceleration factor to be of order 8, which shows the very good speedup obtained in practice. The maximal efficiency of the vectorized codes (no deinterleaved+compiler auto-vectorization, no deinterleaved+OpenMP directives, deinterleaved+OpenMP directives) is reached for a number of interactions approximately equal to 10^5 which would corresponds to an ideal case with $Ncrit \approx 316$ since $Ncrit$ corresponds to the maximal number of target or source particles in a single **P2P** operator.

Notice that large **P2P** operators (in terms of number of particles) rarely appear in a FMM.

Indeed, in *defmm*, the *Ncrit* criterion is usually chosen to be less than 128 in order to minimize the overall FMM cost. However, thanks to the grouping of **P2P** we described in Sect. 7.5.5.1 and exploited in *defmm*, the full efficiency appearing in Fig. 7.21 for large amount of interactions can be obtained in practice on non-uniform particle distributions with our implementation.

7.5.5.3 Precomputation

Our code is supposed to be included into an iterative solver, which implies that the same global FMM will be applied many times until the convergence of this solver is reached. In that application context, only the charges change, but the particle locations remain the same. This is the reason why we can compute only once the near field part of the matrix representation of the FMM (that is we adopt a $\mathcal{H}^2$-matrix like treatment of the near field, see Sect. 2.2.5, in which the **P2P** operators are represented and stored as non compressed matrices). We implemented this precomputation feature.

More precisely, by introducing the matrix $R_{t,s} \in \mathbb{C}^{(\#t\cap X)\times(\#s\cap Y)}$ such that $(R_{t,s})_{\mathbf{x},\mathbf{y}} := G(\mathbf{x},\mathbf{y})$, $\mathbf{x} \in t\cap X$, $\mathbf{y} \in s\cap Y$, Alg. 28 consists in applying $(R_{t,s})$ to a vector $q_{|s} \in \mathbb{C}[s\cap Y]$ and in adding the result to a vector $p_{|t} \in \mathbb{C}[t\cap X]$. Since the particles are supposed to be static, the entries of this matrix $(R_{t,s})$ do not change between two FMM applications and $R_{t,s}$ can thus be computed once and reused. Hence, since the application of a **P2P** is resumed this way to a matrix-vector product, BLAS 2 routines can be exploited. This requires the vectors $p_{|t}$ and $q_{|s}$ to exist or to be assembled when $R_{t,s}$ is applied.

To efficiently benefit from the BLAS 2 calls, we allocate two global vectors of complex numbers, $\mathfrak{p} \in \mathbb{C}^{\#X}$ and $\mathfrak{q} \in \mathbb{C}^{\#Y}$, such that the entry i of the vector $\mathfrak{q}$ (resp. $\mathfrak{p}$) corresponds to the i^{th} source (resp. target) particle according to the Morton ordering induced by the tree construction (see Sect. 7.3.4). $\mathfrak{p}$ stores the potentials associated to the target particles (initialized with zeros before applying the FMM) and $\mathfrak{q}$ stores the charges associated to the source particles. Doing so, a given leaf cell can interact with an entire group of possibly non-leaf cells in a single matrix-vector product, which may be more efficient than multiple small ones (see Fig. 7.22). Notice that the BLAS 2 routines being memory bound (see App. D.1.1.1), the performance impact of this optimization depends on the particle distribution.

The precomputed near-field application can be easily separated from the far field application and it is convenient to do so when the precomputed **P2P** matrices are exploited. Indeed, if the precomputed **P2P** matrices are applied during the DTT, one has to be able to store them in such a way that any given **P2P** matrix corresponding to a non well-separated pair of cells can be efficiently found knowing the indices of these cells in the global cell arrays. On the other hand, by applying the precomputed **P2P** matrices *outside* the DTT, if each of them is tagged by the indices of the corresponding source and target cells, one may find all the needed information in $\mathcal{O}(1)$ operation count for each precomputed **P2P** matrix application (i.e. find the correct offsets in $\mathfrak{p}$ and $\mathfrak{q}$). During the precomputation step, we thus tag each target cell t by the indices of the source cells s such that the interaction between t and s is computed directly and we add to this tag a pointer on the corresponding precomputed **P2P** matrix. This construction is done in practice using Alg. 29 which corresponds to a blank DTT for the near field precomputation. This blank DTT can be fused with the blank DTT of Sect. 7.5.3.1, but we did not do so in our implementation.

Algorithm 29 NFBDTT (Near Field Blank Dual Tree Traversal)

```
1: // Input : cell t, cell s
2: // Output: ∅
3: procedure NFBDTT(t, s)
4:     if isLeaf(t) or isLeaf(s) then
5:         Tag t using the index of s
6:         Allocate and fill the local near field matrix using kernel evaluations
7:     else
8:         for t' ∈ Sons(t), s' ∈ Sons(s) do
9:             NFBDTT(t',s')
10:        end for
11:    end if
12: end procedure
```

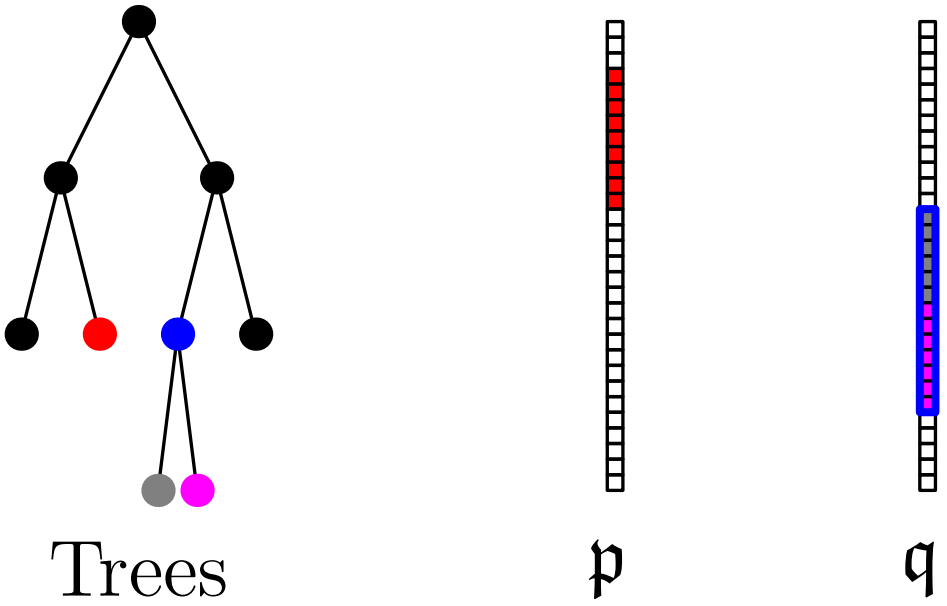


Figure 7.22: Schematic representation of the correspondences between cells and particle indices in the global arrays $\mathfrak{p}$ and $\mathfrak{q}$. Here, the source tree is equal to the target one but $\mathfrak{p}$ differs from $\mathfrak{q}$ since the first of these arrays corresponds to the potentials of the target particles and the second one to the charges of the source particles. Each entry of these arrays of a given color corresponds to a particle of the cell of the same color thanks to the Morton ordering. All the particles in the descendant of a cell have consecutive positions in these arrays (e.g. the blue cell containing the gray and the magenta one). The matrix representing the near field between the red cell and the gray one is then a restriction of the matrix of the near field between the red and blue cells. The corresponding array of charges of the gray cell is a subarray of the one of the blue cell.

The precomputation of the near-field part is particularly well-suited in the case of kernels that are costly to evaluate because it allows to switch **all** the kernel evaluations to the precomputation step. Only the charges at the source particles change between two FMM applications with static particles.

In Tab. 7.4, we compare the application timings of the precomputed **P2P** matrices (referred to as *precomputed NF*) and the SIMD vectorized direct evaluation of the **P2P** operators (referred to as *on-the-fly NF*) we described in Sect. 7.5.5.2. The application of the precomputed **P2P**

Method	Distribution	Time
on-the-fly NF	Cube	70.06
precomputed NF	Cube	40.60
on-the-fly NF	Sphere	53.47
precomputed NF	Sphere	27.63

Table 7.4: Application timings (seconds) of the near field depending on the **P2P** evaluation method and the particle distribution. *on-the-fly NF* corresponds to the direct evaluation of the near field using the method described in Sect. 7.5.5.2 and *precomputed NF* refers to the BLAS 2 evaluation using the near field precomputation.

matrices performs up to 50% faster than the direct *on-the-fly NF* (on the sphere test case). On the uniform cube case, the gain is less important but still significative ($\approx$ 43% faster). On this uniform volumic case, the *on-the-fly NF* performs better than on the sphere since the **P2P** involve approximately the same number of target and source particle, which are themselves approximately equal to $Ncrit$, which is not the case of the surfacic distributions on which a certain amount of cells contain significantly less particles than $Ncrit$.

Remark 7.5.7. *The precomputation of the near field has an impact on the memory footprint of the method. This may bound in practice the Ncrit values used with this optimization.*

7.5.6 Handling the FFTs

Numerous FFTs are performed in *defmm*. We use the *state-of-the-art* implementation FFTW [107] to compute them. The problem with these FFTs is that they operate on relatively small vectors/arrays. We may specify that the FFT computation is divided into two parts in FFTW. First, a "FFTW plan" is assembled, deciding on the best strategy to run the FFT on a specific machine. This FFTW plan is then executed on the input and output data, performing the FFT. The issue is that the FFTW plan creation is more costly than its execution for the FFT sizes we are considering. Fortunately, these FFTW plans can be reused, provided that the input/output arrays they are applying to have the same memory alignment. Since all the FFTs have the same size because we keep the same interpolation order all along the FMM, only one single FFTW plan has to be precomputed. The FFTs used in *defmm* perform on small arrays, on which the FFTW plan construction is costlier than the execution of these plans. In this section, we present a solution to handle these numerous small FFTs by using a FFTW routine (Sect. 7.5.6.1) and by intensively reusing the same plan (Sect. 7.5.6.2). In Sect. 7.5.6.3, we provide performance results for the two approaches.

7.5.6.1 Batched FFTs

The FFTW library proposes a function directly resulting in batch FFTs [4]. This function allows to initialize a single FFTW plan for a set of equally sized FFTs. The only needed assumption for this function to be applied is a regularity in the storage of the transformed arrays. These arrays have to be stored with equal distances in the memory. This is obtained in our implementation by using the global expansion arrays presented in Sect. 7.5.2.2 (the arrays on which the batch FFTs perform are thus the global arrays corresponding to the expansions in the Fourier domain). To use these batch FFTs, the expansion copies are performed all together, *before* or *after* FFT computation (depending on the expansion type). We refer to this method as the *batch* method. According to the FFTW3 documentation [4], the batch method can improve the performances compared to multiple FFTW calls.

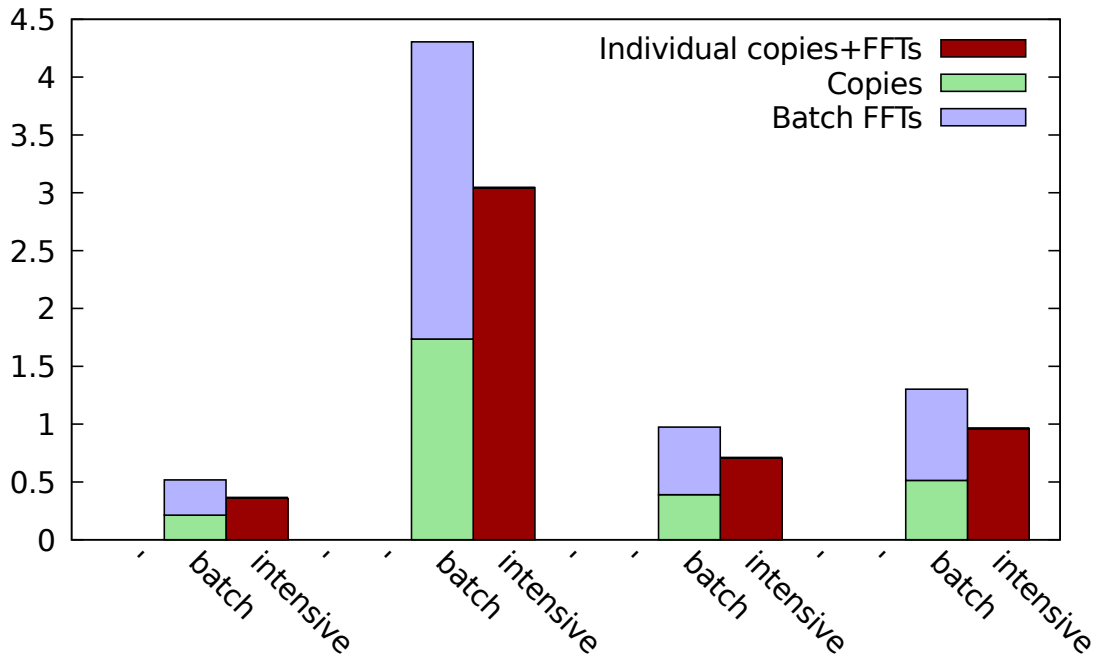


Figure 7.23: Timings (in seconds) of the entire *m2fpass* corresponding to a one-dimensional interpolation order $L = 4$. From left to right, K being the particle distribution length multiplied by the wavenumber: cube with $K = 0$, cube with $K = 64$, sphere with $K = 0$, sphere with $K = 64$.

7.5.6.2 Intensively reusing the FFTW plans

The directional expansions of a given cell are stored consecutively with a padding ensuring the same memory alignment for each expansion (see Sect. 7.5.2.1). A simple loop over the effective directions of a cell thus allows to evaluate the **M2F** or **F2L** operators on each of its directional expansions. In addition, thanks to the *m2f/f2l* passes, a single loop on the global cell array is needed to call these operators on each cell. Each **M2F** operator copies the multipole expansions in a larger array (storing the expansion in the Fourier domain) with gaps corresponding to the zero-padding introduced for the circulant embedding (see Sect. 4.1.4) and then performs a FFT on this array using the precomputed FFTW plan. The **F2L** is treated in a similar way, with first the FFT application and then the copy. We refer to this method as the *intensive* method.

Remark 7.5.8. *A scaling is also needed in the **F2L** because of the output of the FFTW routines. Since the data are also copied, we perform this scaling during these copies.*

7.5.6.3 Comparison

The timings of the two variants (batch and intensive) are given in Fig. 7.23. The intensive method is always less costly than the batch one on the two particle distributions and in the two frequency regimes. The data are likely loaded only once with the intensive method and both the copies and FFTs are performed in the cache memory (since the FFTs perform on small arrays). The batch method needs these two steps to be applied separately, implying that the data are loaded twice. Since the copies and FFTs are memory-bound (see App. D.1.1.1), this difference may explain the costs on Fig. 7.23. By splitting the timings of the copies and of the FFT applications in the batch method, we observe that the FFT application is approximately as costly as the overall cost of the intensive method, which seems to confirm our analysis. We observed the same timing differences

for the other tested interpolation orders. We thus discarded the batch method and the intensive one is the default choice in *defmm*.

7.5.7 M2P and P2L operators

The constraint of far field interactions performed only with cells at the same level (see Sect. 7.3.1.3) can lead in practice to large **P2P** applications. Since the complexity of a **P2P** is quadratic, we want to avoid the too large **P2P** operations. We then introduce the **M2P** and **P2L** operators that are commonly used in FMM (see for instance [73, 80, 176, 221]). Notice that such operators have been investigated in the polynomial interpolation-based FMM in [182] with a specific tree construction and a list-based approach (see Sect. 2.2.4.2). Actually, our approach is slightly different than these original ones: instead of using these extra FMM operators to avoid the expansion creation for cells containing a very small number of particles. We want to avoid here the too costly near field *cell-cell* interactions by transforming (some of) them into far field interactions, which is somehow the opposite strategy.

These new operators, described in Sect. 7.5.7.1, are able to perform far field interactions on cells lying at different tree levels. Because our global approach does not construct the interaction lists, we reformulated our DTT (see Sect. 7.3.3) into a refined traversal, which is described in Sect. 7.5.7.2. The new operators can be also expressed in the high-frequency regime, leading however to strong complications. We discuss in Sect. 7.5.7.3 the applicability of the two new operators in the high-frequency regime. Numerical results are provided in Sect. 7.5.7.4.

7.5.7.1 Operator formulations

The main point of the fully adaptive approach as presented in Sect. 2.2.4.2 is the introduction of two operators performing on cells on different levels. As opposed to the other FMM operators, these new operators perform interactions of different natures: *cell-particle* interactions for the **M2P** (**M**ultipole-to-**P**articles) and *particle-cell* interactions for the **P2L** (**P**articles-to-**L**ocal). The first transforms a multipole expansion of a cell c_0 on direct far field contribution for the particles of a given leaf c_1 such that for any particle $\mathbf{z} \in c_1$, c_0 and $\mathbf{z}$ are well-separated according to a criterion that needs to be defined (see Sect. 7.5.7.2). In the same manner, the **P2L** operator directly adds to the local expansion in a cell c_0 the contribution of a set of particles of a leaf cell c_1 well-separated with c_0 according to a criterion that needs to be defined. Assuming that a particle has a radius equal to zero and a center equal to its position, the criterion of the adaptive MAC (see Sect. 2.2.4.2) can be used. This leads to interaction patterns as illustrated in Fig. 7.24.

The new operators are formulated as direct computations where the target or source group of particles corresponds to an interpolation grid $\hat{\mathbb{G}}$. The **P2L** operator with source cell s reads:

$$\mathcal{L}(\hat{\mathbf{x}}) += \sum_{\mathbf{y} \in s \cap Y} G(\hat{\mathbf{x}} + ctr(t), \mathbf{y}) q(\mathbf{y}), \qquad \forall \hat{\mathbf{x}} \in \hat{\mathbb{G}},$$

where $\mathcal{L}$ denotes the local expansion in the target cell.

Remark 7.5.9. *This formulation exactly corresponds to our implementation: because the interpolation nodes are given relatively to the cell center and because the particle positions are given in exact coordinates, we need to add the target cell center.*

The **M2P** operator with target cell t reads in the following way, the charges being replaced by the multipole expansion elements and the interpolation nodes being the source particles:

$$p(\mathbf{x}) += \sum_{\hat{\mathbf{y}} \in \mathbb{G}} G(\mathbf{x}, \hat{\mathbf{y}}_l + ctr(s)) \mathcal{M}[s](\hat{\mathbf{y}}_l), \qquad \forall \mathbf{x} \in t \cap X,$$

where $\mathcal{M}[s] \in \mathbb{C}[\hat{\mathbb{G}}]$ denotes the multipole expansion in the source cell. The complexity of each of these new operators is equal to $\mathcal{O}\left(L^d N crit\right)$ where L refers to the one-dimensional interpolation

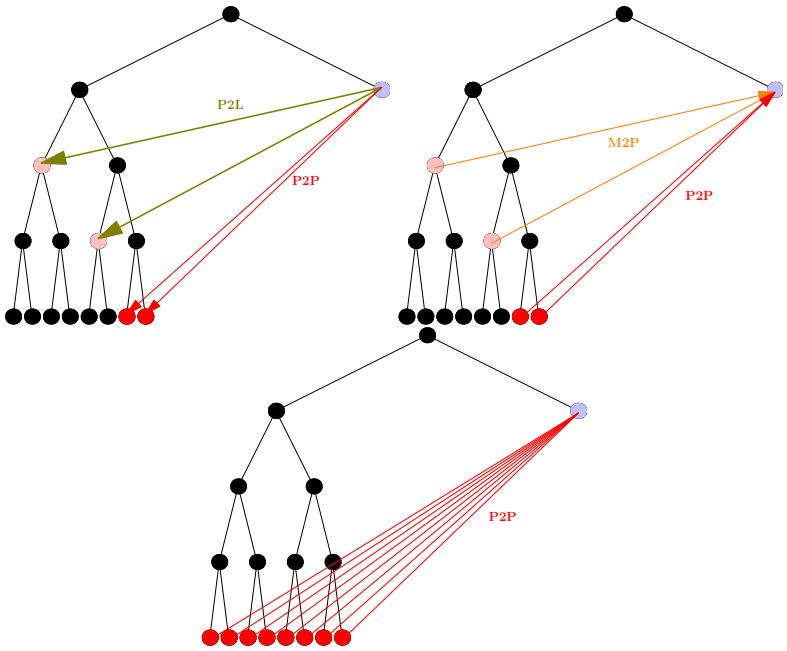


Figure 7.24: Schematic view of the **M2P** (orange arrow) and **P2L** (green arrow) operators. We consider the interactions of the pale blue cell. Instead of computing the near field between this cell and all the other leaves (bottom) with **P2P** operators (red arrows), we switch to *particle-cell* interactions (top left) if the pale blue cell is a source cell and to *cell-particle* interactions (top right) if it is a target cell.

order, provided that the kernel can be evaluated in $\mathcal{O}(1)$ operations. To be relevantly applied in the FMM, we still need to give a precise criterion allowing to use such operator and we also have to rewrite our DTT-based horizontal pass.

7.5.7.2 Wideband refined Dual Tree Traversal in *defmm*

The criterion we use in *defmm* to decide if the particles of a cell c_0 are well-separated from another cell c_1 for a **M2P** or **P2L** application, where c_0 and c_1 are low-frequency cells, is given by

$$\frac{rad(c_0) + rad(c_1)}{|ctr(c_0) - ctr(c_1)|} \leq \eta \tag{7.9}$$

where $\eta \in [0, 0.7]$. A **M2P** or **P2L** operator is less costly than a **P2P** one only if the number of interpolation node in the cell whose interaction is involved is strictly less than the particle number contained in this cell. Before testing the criterion in the inequality 7.9, we check that this condition on the number of particle is verified. This leads to a MAC $\mathcal{B}(c_0, c_1)$ which is true if these two last conditions are verified.

Remark 7.5.10. *From a programming viewpoint, the implementation of the **M2P** and **P2L** operators are directly based on the **P2P** one where one of the particle sets is an interpolation grid. We then benefit from the optimizations we discussed in Sect. 7.5.5.2 for the near field. The **M2P** and **P2L** operators are then applied on-the-fly without any precomputation.*

We then modified the first horizontal pass we gave in Alg. 21 (in Sect. 7.3.3) into a more flexible algorithm switching between **M2L**, **P2L**, **M2P** and **P2P** operators, depending on the local distribution. In the high-frequency regime, no **M2P** or **P2L** operator is performed, as justified in Sect. 7.5.7.3. This new horizontal pass, named *refined horizontal pass*, is formulated explicitly

in Alg. 30. The "outer if" takes into account the high-frequency regime. The adaptivity with the new operators is specified by testing which cell is a leaf when such a cell appears. According to this information, if the MAC $\mathcal{B}$ fails, the non-leaf cell is splitted and the process continues recursively. Because the roots of the source and target trees are supposed to be the same, the condition $Level(t) = Level(s)$ for the **M2L**s in the high-frequency regime is ensured by calling Alg. 30 on these roots.

Algorithm 30 RFNDHRZPASS (Refined Horizontal Pass). This algorithm is called on the roots of the target and source trees.

```
1: procedure RFNDHRZPASS(t, s)
2:     if isHF(t) or isHF(s) then
3:         if hfMAC(t, s) then
4:             hfM2L(t, s)
5:             return
6:         end if
7:         if isLeaf(t) or isLeaf(s) then
8:             P2P(t, s)
9:             return
10:        end if
11:    else
12:        if Level(t) = Level(s) and MAC(t, s) then        ▷ Apply M2L if possible, not P2P
13:            M2L(t, s)
14:            return
15:        end if
16:        if isLeaf(t) and isLeaf(s) then
17:            P2P(t, s)
18:            return
19:        end if
20:        if isLeaf(t) then
21:            if B(t, s) then
22:                M2P(t,s)
23:                return
24:            else
25:                for s' ∈ Sons(s) do
26:                    RFNDHRZPASS(t, s')
27:                end for
28:                return
29:            end if
30:        end if
31:        if isLeaf(s) then
32:            if B(t, s) then
33:                P2L(t,s)
34:                return
35:            else
36:                for t' ∈ Sons(t) do
37:                    RFNDHRZPASS(t', s)
38:                end for
39:                return
40:            end if
41:        end if
42:    end if
43:    for t' ∈ Sons(t) do
44:        for s' ∈ Sons(s) do
45:            RFNDHRZPASS(t', s')
46:        end for
47:    end for
48: end procedure
```

7.5.7.3 High-frequency constraints

A natural question concerns the definition of the **M2P** and **P2L** operators in the high-frequency regime, that is the definition of *directional* versions of these operators. They can be easily written from a mathematical viewpoint with a small modification of the high-frequency (directional) MAC, but their execution leads to technical difficulties and greater overall complexity. Indeed, two cells that would be admissible for a **M2P** or **P2L** can involve different directional expansions. This reduces the efficiency of the SIMD code we developed for the near field interactions and used for the **M2P** and **P2L** operations. Since there is no precomputed **M2L** matrix for these two cells, these directions have not been precomputed during the blank horizontal pass. Hence, one has to retrieve them *on-the-fly* and the cost of this operation increases from the leaves to the root of the 2^d-trees. In addition, the corresponding directional expansions may not be stored, so one has to determine them during the blank passes in order to allocate them before the FMM application. There is a problem when doing so: directional expansions may be created only to support a **M2P** or **P2L** operator, adding a cost on the upward and downward passes during the FMM application. This is justified for a **M2L** operation because of its low evaluation cost but the potential gain is unclear when introducing high-frequency directional **M2P** or **P2L** operations. We thus considered only a low-frequency refined DTT exploiting **M2P**/**P2L** operators in *defmm* (i.e. no high-frequency **M2P**/**P2L** operators).

There is another reason for this choice: when dealing with integral equations in the high-frequency regime and with meshing, a certain amount of elements in the boundary discretization is needed for each wavelength in order to ensure a given accuracy. Hence, we expect the majority of leaves to be in the low-frequency regime (or near the low- and high-frequency limit depending on the $Ncrit$ value). This means that above this limit, mainly far field **M2L** interactions are performed. So we believe that the implementation of the **M2P** and **P2L** operators in the high-frequency regime would lead to small computational gains for the practical applications.

7.5.7.4 Tests and discussion

The approach we provide in Sect. 7.5.7.2 is mainly justified on highly non-uniform distributions (i.e. with higher non-uniformity than the sphere test case we described in Sect. 7.1.3). Such case may appear in the integral equation context when dealing with mesh refinement (see for instance [61]). We mimic the effect of a mesh refinement in a highly non-uniform distribution on the boundaries of the unit cube (the refinement is done around the corners), as illustrated on Fig. 7.25. Our test case is composed of 10124406 particles. We present in Tab. 7.26 the performance of *defmm* with and without the refined horizontal pass on the test cases of Sect. 7.1.3 and on the refined cube. Since only a single application is considered and because we only provide the application timings, the **P2P** operators are evaluated *on-the-fly* using the code presented in Sect. 7.5.5.2.

On the uniform cube test case, the performances of our original horizontal pass and of our refined horizontal pass are the same. Indeed, the leaves are almost all at the same tree level because of the uniform distribution (and because the same value of $Ncrit$ is used for all the leaves). Since the chosen $Ncrit$ value is greater than the number of interpolation nodes in the grids associated to the leaves, no **M2P** or **P2L** is performed on the uniform cube test case. Hence, we just measure that the additional tests in the refined horizontal pass (see Alg. 30) have a negligible cost compared to the overall FMM application one. On the sphere test case, we observe that the refined horizontal pass performs slightly better than the horizontal pass. Indeed, on this distribution, a certain (relatively small) amount of cells leads to the application of **M2P** and **P2L** operators. On this distribution, the maximal size of the **P2P** operators is still limited, which leads to only a small performance gain of the refined horizontal pass over the original horizontal pass.

Conclusions are different on the refined cube: when applying the horizontal pass on this test case, large **P2P** operators are called due to the constraint of far field interaction at the same tree level (and to the grouping strategy of Sect. 7.5.5.1). On this distribution, the refined horizontal pass performs ≈ 2.7 times faster than the original horizontal pass, showing that our strategy is

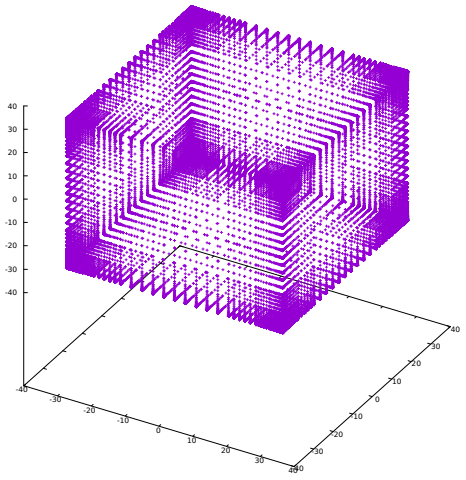


Figure 7.25: Illustration of a refined cube. The particles are concentrated around the corners and the edges of the boundaries of the cube.

clearly justified on highly non-uniform distributions.

Similar results were obtained for other tested interpolation orders, so we do not report them here. When high frequencies are considered, the conclusions are the same, except that the effect of the refined horizontal pass is less sensitive as the wavenumber increases since no **M2P**/**P2L** operation is performed in the high-frequency regime in *defmm*. Notice that the precomputation cost is not impacted by the refined horizontal pass.

Remark 7.5.11. *For static particles, the **M2P** and **P2L** matrices can be precomputed as we did for the **P2P** matrices in Sect. 7.5.5.3. This increases the precomputation time, but substantial gain may be expected on highly non-uniform distributions. We did not implemented nor tested this optimization.*

7.6 Numerical results

In this section, we provide a set of performance and convergence numerical results obtained with *defmm*. All codes run sequentially (the exact architecture has been described in Sect. 7.1.3). We first check the error of the overall method in Sect. 7.6.1 as well as a complexity verification. In Sect. 7.6.2, we provide comparisons between *defmm* and *dfmm*.

7.6.1 Relative error and numerical complexity

Relative error First of all, we provide the convergence results of our global method. The non-uniformity of the distribution, its size, the wavenumber and the interpolation order can have an impact on the overall error. We then start by providing a convergence result measuring the impact of the wavenumber on the error for a uniform cube test case in Fig. 7.27. The relative norms we use are computed, for a vector of potential $\mathbf{v} = (v_1, ..., v_N) \in \mathbb{C}^N$ and an approximation of $\mathbf{v}$ denoted by $\tilde{\mathbf{v}} \in \mathbb{C}^N$, using $\frac{||\mathbf{v}-\tilde{\mathbf{v}}||}{||\mathbf{v}||}$ for a norm $||\cdot||$ on $\mathbb{C}^N$. The charges at the source particles are generated randomly using an uniform rules on $[0, 1]$.

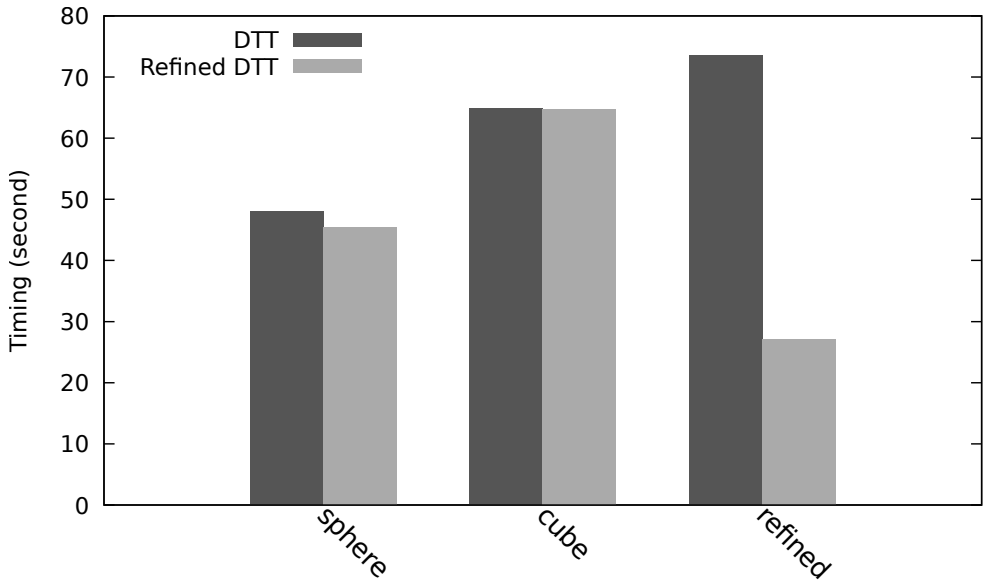


Figure 7.26: Application timings (seconds) of the FMM ($\kappa = 0$) when using the horizontal pass of Sect. 7.3.3 (DTT) and the refined horizontal pass of Sect. 7.5.7.2 (refined DTT) depending on the particle distribution (from left to right: sphere, cube, refined cube). We used a one-diemnsioanl interpolation order $L = 4$.

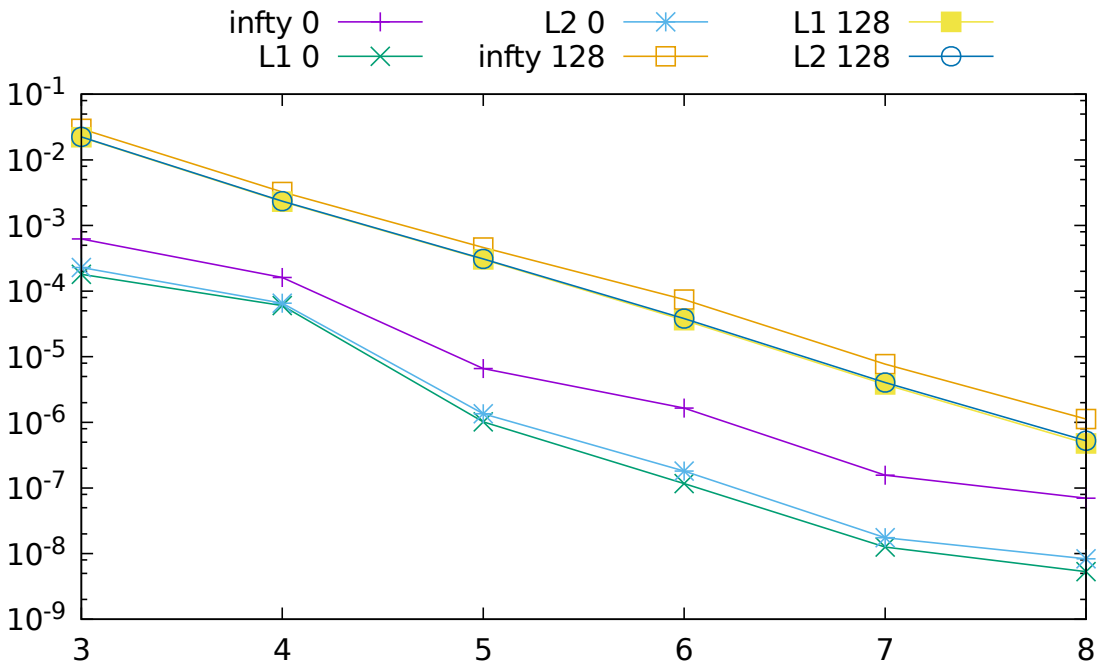


Figure 7.27: Relative error of the overall FMM with regard to one-dimensional interpolation order. $infty$ refers to the L^∞ norm, $L1$ to the L^1 one and L^2 to the L^2 one. After the norm, we indicate the length of the geometry multiplied by the wavenumber. Test case: uniform cube with $N = 125000$. We used $Ncrit = 32$.

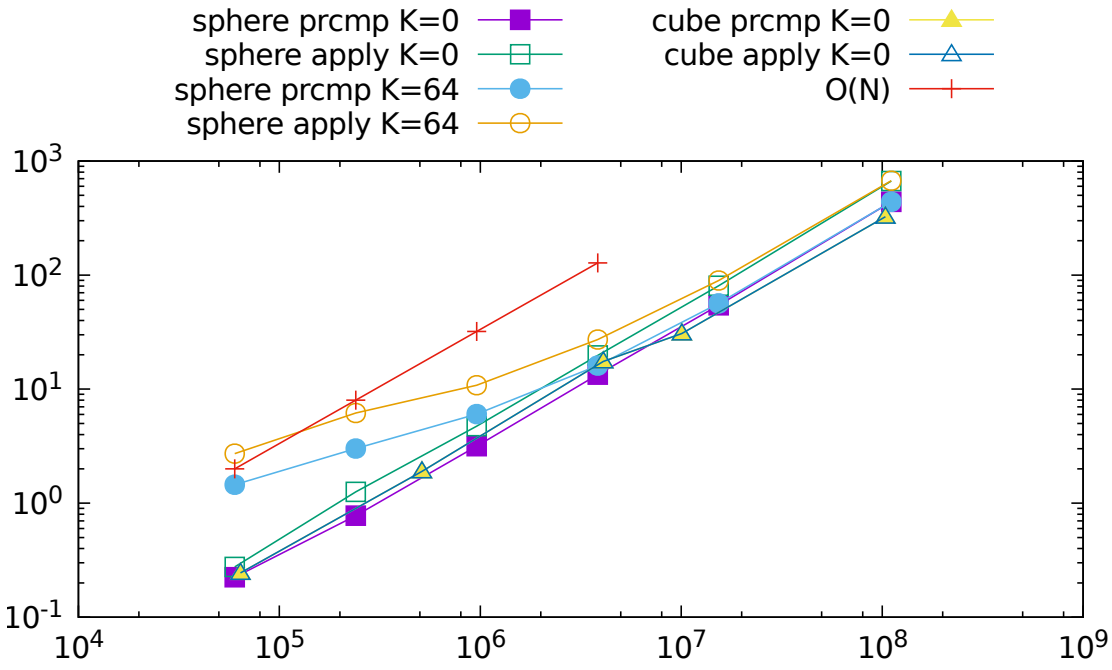


Figure 7.28: Costs (seconds) of the precomputation (prcmp) and application (apply) steps of *defmm* with regard to the number of source particles (equal to the number of target particles) for the distributions of Sect. 7.1.3. The geometry length multiplied by the wavenumber is indicated with K. A theoretical indicative $\mathcal{O}(N)$ curve is provided in red.

The convergence rate follows the estimate given in Sect. 7.2. We obtained similar results on the sphere test case, so we do not report these results. Clearly, the convergence follows the same rate in the two frequency regimes. However, for a fixed targeted accuracy, the requested interpolation order in the high-frequency regime should be chosen slightly greater than in the low-frequency regime. We believe that this is a consequence of the small initial set of directions we considered in our implementation for efficiency reasons (actually the same set than in *dfmm*), since the use of refined sets of directions reduces the error (while increasing the overall cost of the FMM).

Numerical complexity. A classical test for the FMM library consists in measuring the application timings with regard to the number of source (or target) particles (the two point clouds are supposed the same size). Such results are provided in Fig. 7.28.

For a fixed wavenumber (i.e. only varying the number of particles), *defmm* provides a numerical $\mathcal{O}(N)$ complexity on both surface and volumic distributions. Notice that this result does not contradict the complexity estimates of the directional FMM, that are supposed to be $\mathcal{O}(NlogN)$ on surface distributions. Indeed, the wavenumber should increase as the number of particle increases to obtain this complexity. In Fig. 7.29, the number of particle per wavelength is chosen to be approximately constant and we increase the wavenumber (i.e. the number of particles increases with the wavenumber) on the unit sphere. Both the precomputation and the application steps of *defmm* follow the $\mathcal{O}(NlogN)$ estimate.

7.6.2 Performance comparison with *dfmm*

In this section, we compare *defmm* with *dfmm* [1] on various geometries and for different wavenumbers. The *dfmm* code implements the closest method to *defmm* and is, to our knowledge, one of the

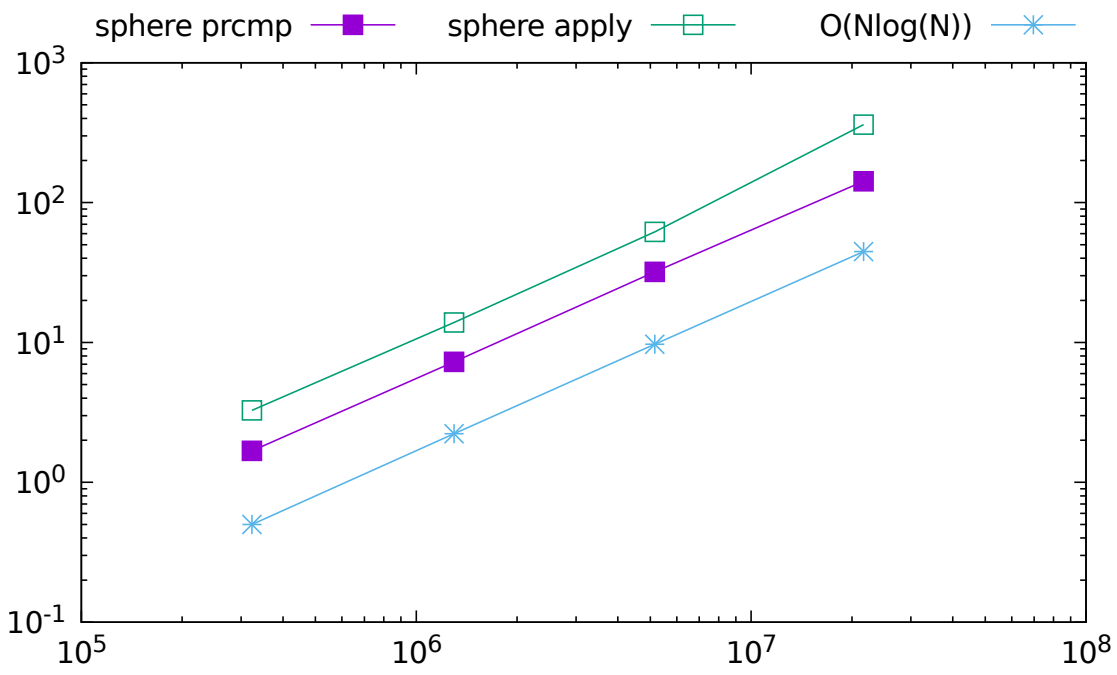


Figure 7.29: Costs (seconds) of the precomputation (prcmp) and application (apply) steps of *defmm* with regard to the number of source particles on the sphere. The wavenumber is chosen so that there is a constant number of particle per wavelegnth. A theoretical indicative $\mathcal{O}(Nlog(N))$ curve is provided in blue.

only available open source directional FMM libraries. We provide the comparison results with the two best variants of *dfmm* [173]: *dfmm-IAblk* and *dfmm-SArcmp*, that were described in Sect. 7.1.2. The *dfmm-SArcmp* method has a costly precomputation step but may lead to faster evaluations than the second thanks to recompressions. The *dfmm-IAblk* method is based on a vector stacking scheme combined with permutations, exploiting the efficiency of the BLAS 3 routines. We recall that *dfmm* uses the Chebyshev interpolation and relies in practice on explicit low-rank approximations of the **M2L** operators. Hence, the threshold in the low-rank approximations has to be chosen carefully to preserve the global accuracy of the FMM. We followed the methodology used in the tests in [174] and consisting in fixing the threshold to 10^{-L} for a one-dimensional interpolation order L. According to our tests, the accuracy of the Chebyshev interpolation is comparable to the accuracy of the *defmm* equispaced one with such a choice.

Optimizations of *defmm*. Our goal in this section is to validate the *defmm* approach (directional FMM with equispaced interpolation grids) as well as our high-level algorithmic choices (presented in Sect. 7.3). Hence, a certain amount of optimizations of *defmm* are not used for the comparisons. Since the near field part is computed *on-the-fly* in *dfmm*, our comparison tests are provided without precomputation of the near field for *defmm* (that is without the optimization described in Sect. 7.5.5.3 but with the SIMD optimizations described in Sect. 7.5.5.2). As we discussed in Sect. 7.5.5.3, the precomputation of the near field part is mainly interesting for multiple applications of the same FMM to different vectors of charges and strongly modifies the ratio between the precomputation cost and the application one. Since we are only concerned by a single application, this optimization has no mean here and would have prevent the effective comparison of application and precomputation timings independently.

For similar motivations, because *dfmm* does not benefit from the **M2P** and **P2L** operators, the

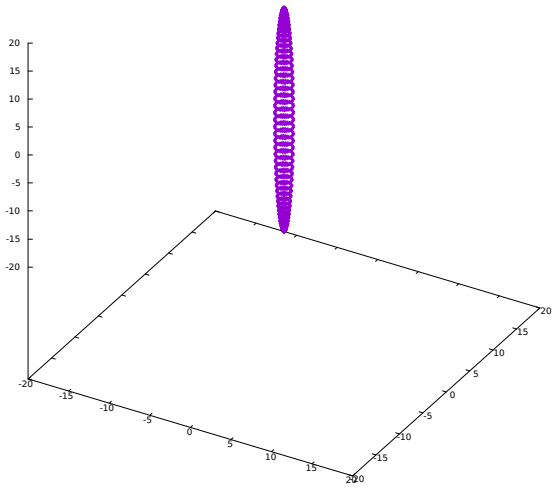


Figure 7.30: Representation of the ellipse particle distribution.

refined horizontal pass presented in Sect. 7.5.7.2 (mostly useful for highly non-uniform distributions) is not used here.

Test cases. Elongated objects are interesting for directional approaches especially with the *dfmm* optimizations [174]. We then added to the test cases presented in Sect. 7.1.3 and to the refined cube of Sect. 7.5.7.4 another distribution for our comparison with *dfmm*. This new test case, illustrated on Fig. 7.30, is an ellipse with a concentration of nodes at the poles. This test case is composed of 10M particles.

Results. We are interested in this section by the performances of the different libraries for a fixed accuracy, which is obtained by chosing the same interpolation orders. In Figs. 7.31 and 7.32 are presented the comparison between the two libraries with the two variants of *dfmm* for various particle distributions, wavenumbers and interpolation orders. Both the precomputation and application timings are provided. We tuned the parameters (the number of levels, i.e. the $MaxDepth$ criterion, for *dfmm* and the $Ncrit$ criterion for *defmm*) for both libraries in order to minimize the application time (not the precomputation one). Actually, we were not able to run *dfmm* (neither *dfmm-IAblk* nor *dfmm-SArcmp*) on the refined cube for one-dimensional interpolation orders greater than 4 because of the too high required memory. Hence, we only provide results on this particle distribution for $L = 4$ with *dfmm* (Fig. 7.31). We limited ourselves to particle distribution such that $\kappa a \leq 64$ (κ being the wavenumber and a the length of the particle distribution) since, above this value, all the leaves for the uniform cube test case are at high-frequency levels, which is not realistic for our practical applications.

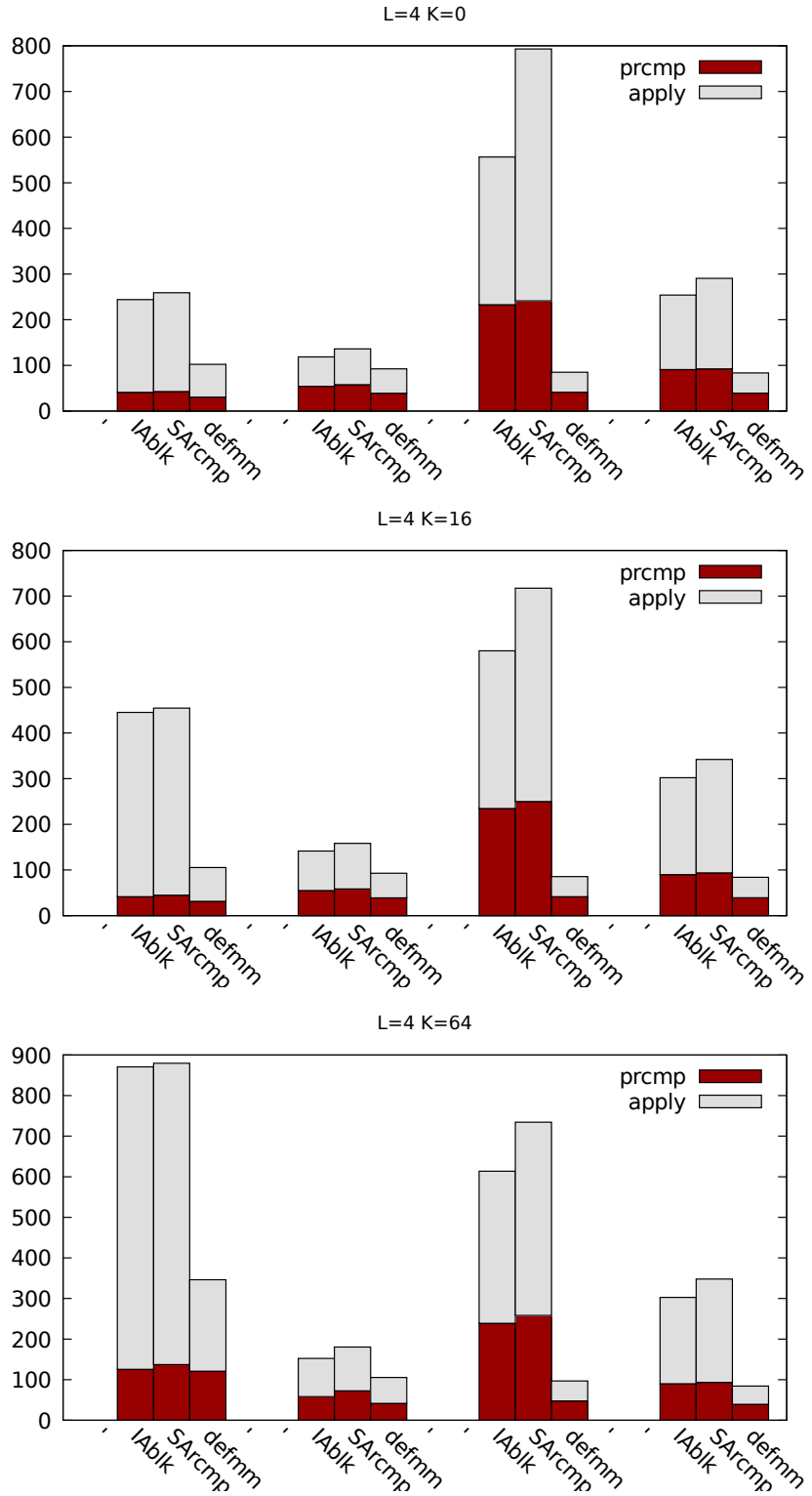


Figure 7.31: Timings (seconds) for the precompute step (prcmp) and the application (apply) of the different methods for various distributions (from left to right: uniform cube, sphere, refined cube, ellipse) and one-dimensional interpolation order equal to 4. From top to bottom, the geometry lengths multiplied by the tested wavenumbers are: 0, 16, 64. IAblk corresponds to *dfmm-IAblk* and SArcmp to *dfmm-SArcmp*.

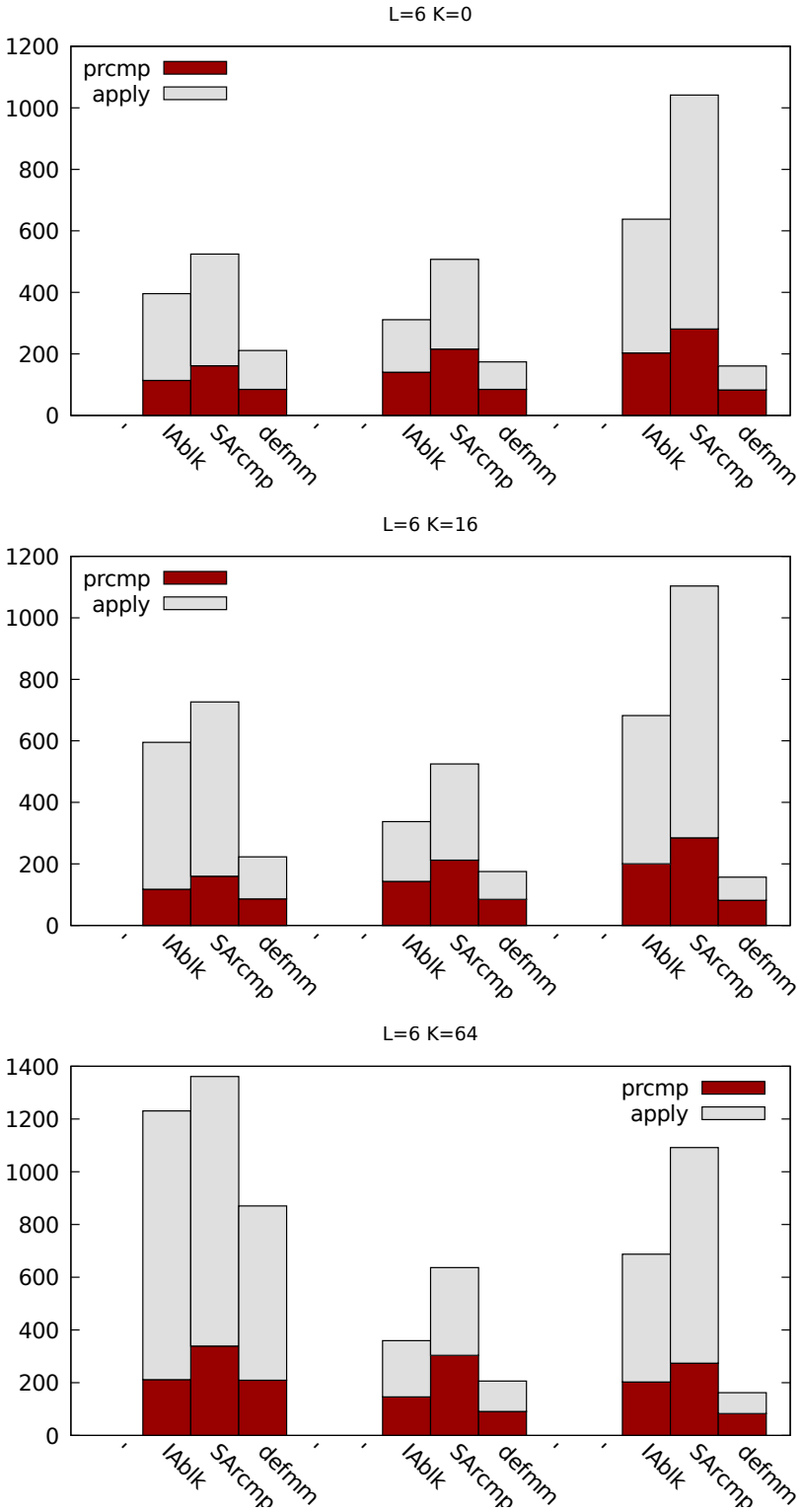
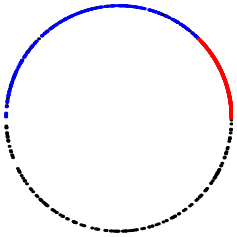
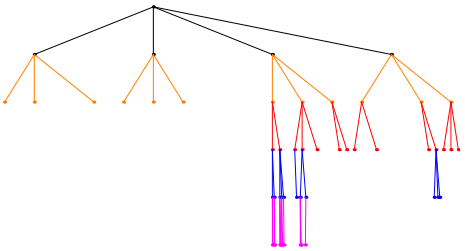


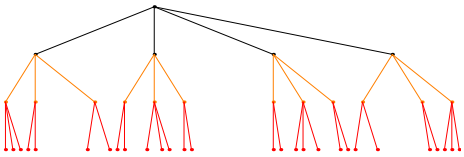
Figure 7.32: Timings (seconds) for the precompute step (prcmp) and the application (apply) of the different methods for various distributions (from left to right: uniform cube, sphere, ellipse) and one-dimensional interpolation order equal to 6. From top to bottom, the geometry lengths multiplied by the tested wavenumbers are: 0, 16, 64. IAblk corresponds to *dfmm-IAblk* and SArcmp to *dfmm-SArcmp*.



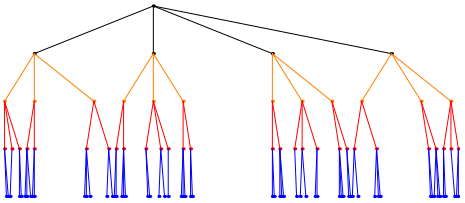
(a) 2D particle distribution on the unit circle composed of 3 different random uniform samplings: one on the entire circle (black), a second on the upper half circle (blue) and a third on a part of the north-east quarter (red).



(b) Corresponding quadtree using $Ncrit = 60$.



(c) Corresponding quadtree using $MaxDepth = 4$.



(d) Corresponding quadtree using $MaxDepth = 5$.

Figure 7.33: Quadtree corresponding to a 2D non-uniform particle distribution (without storing/representing the empty cells) for different tree construction strategies. The construction based on $Ncrit$ leads to a tree with 6 levels, and the two first sons of the root node only have descendants up to the 3^{rd} level. Using the $MaxDepth$ criterion, even for maximal depth strictly less than 6, these two first sons of the root have numerous descendants on the 4^{th} and 5^{th} levels. On each of the corresponding leaves, the average local number of particles is less than $Ncrit$ divided by the number of descendants of the ancestor at the 3^{rd} level. On the other hand, some leaves in the descendants of the third son of the root contain more than $Ncrit$ particles, by comparison with the tree building using $Ncrit$.

Sensitivity to the particle distribution. Since all the tested distributions are composed of $10M$ particles ($\pm 1\%$), the results of Figs. 7.31 and 7.32 allow to measure the sensitivity of each tested method to the particle distribution. Hence, we observe that, for $K = 0$ and $K = 16$, the *defmm* timings are few sensitive to the particle distribution, which is not the case for (the two variants of) *dfmm*. For instance, the minimal application costs for *dfmm* are obtained on the sphere test case, but the *dfmm* performances are more than twice more costly on the uniform cube and on the ellipse, and ≈ 4.6 times more costly on the refined cube for $K = 0$ and $L = 4$. This can be explained by the different tree construction strategies: *defmm* actually adapts its construction to the particle distribution thanks to $Ncrit$ criterion, but the $MaxDepth$ criterion can lead for the (highly) non-uniform distributions to the creation of numerous cells at the deepest levels with a small amount of particles. Said differently, the number of particles contained in two different leaf cells can be drastically different using $MaxDepth$, while $Ncrit$ bounds the number of particles in each cell while avoiding empty cells. This phenomenon is depicted on Fig. 7.33. As a consequence, $Ncrit$ is more suited to obtain a good balance between the costs of the far field and the near field evaluations for any distribution: at the deepest levels, using the best $MaxDepth$ value, cells containing only a very small amount of particle may be treated using far field **M2L** interactions while near field ones would be more efficient. The refined cube actually corresponds to an extreme case of the situation depticted on Fig. 7.33, which explains the strong difference of timings between *defmm* and *dfmm* on this distribution. The performance of *dfmm* on the cube and on the sphere is explained by the chosen values for $MaxDepth$, minimizing the application cost and generating far more near field interactions on the cube than on the sphere.

These results validate the *Ncrit* approach and the algorithmic we used to exploit the associated tree structure according to the particle distribution, i.e. the DTT-based horizontal pass of Sect. 7.5.7.2 and the blank passes of Sect. 7.5.3.

On the uniform cube test case with the highest tested wavenumber ($K = 64$), we observe a different phenomenon: *defmm* becomes more costly on this distribution than on the other ones. This is a consequence of the number of directional expansions in each cells: most cells store the maximal number of expansions, which is costly to construct. This may also be seen on the results of *dfmm* on this same distribution, which increases in the same manner. *defmm* still performs better than *dfmm* in this situation (up to ≈ 2.5 times better, for $L = 4$).

We do not observe such cost increasing on the sphere, which is also supposed to be a difficult case for the directional methods according to [94].

The comparative behavior of *dfmm* and *defmm* on the ellipse test case also shows that *defmm* is able to efficiently handle the effective directions. In other words, the blank passes of Sect. 7.5.3 and the storage of effective directions of Sect. 7.5.2.1 are able to efficiently process the directions effectively induced by the particle distribution.

Sensitivity to the wavenumber. When fixing a particle distribution and varying the wavenumber, one may observe similar differences between *defmm* and the variants of *dfmm* on almost all distributions. Indeed, increasing the wavenumber on a fixed particle distribution only modifies the constant in the overall cost estimate. Since we based our methods on the same directional approach, such behavior was expected. For $L = 6$ on the uniform cube, we observe that the cost difference between *defmm* and the *dfmm* is reduced by increasing the wavenumber. We believe that the BLAS 3 performance impact that benefit from the uniform distribution for the *dfmm-IAblk* variant explains this reduction since *defmm* does not benefit from such optimization. However, the same tendency is also observed when comparing to *dfmm-SArcmp*, so there is another element to take into account. Since mostly the application cost is impacted in this situation with *defmm*, we suspect that this difference reduction is related to the list-based approach of *dfmm* which benefit from the uniform distribution. Indeed, the lists being well filled in such a test case, a given target expansion can be kept in the cache memory while looping over the sources of the corresponding interaction list. Our DTT approach do not guarantee such optimization. This effect can only be observed on the uniform cube, which tends to confirm that the list-based approach of *dfmm* benefit from this situation.

Behavior with regard to the interpolation order. Excluding the uniform cube test case, the two variants of *dfmm* and *defmm* approximately behave in the same manner with regard to the interpolation order (notice that we perform our tests for other interpolation orders with a smaller particle number, up to $L = 8$, leading to the same conclusions). The cost of these methods increases with the interpolation order. However, once again, the cube test case in the high-frequency regime leads to a different interpretation. The relative gain obtained with *defmm* compared to *dfmm* is greater for $L = 4$ than for $L = 6$ on this uniform distribution. Since, once again, this relative difference reduction appears on the application timings, we suspect that this is a consequence of our efficient SIMD code for the near field, whose effect is less sensible as the cost of the far field increases (i.e. by increasing the interpolation order). This efficient **P2P** operator implementation benefits from the well filled cells (in terms of particle) on the uniform distribution, for the near field evaluation. At the same time, *dfmm* benefits from the uniform distributions for the well filled interaction lists (in terms of number of cells), for the far field evaluation. When modifying the interpolation order, mostly the far field is impacted (the optimal $MaxDepth$ does not change). Hence, these considerations may explain the phenomenon we observed.

Remarks. We can provide high-level conclusions to these tests.

- *defmm* performs better than any variant of *dfmm* on all the tested cases, interpolation orders and wavenumbers.

- The uniform cube test case with high wavenumbers is by far the most time-consuming one for our implementation. Notice that for the surfacic distributions coming from boundary integral method, this uniform cube test case is not realistic. Hence, we will never see this particle distribution in our applications.
- Except for the uniform cube with $K = 64$, the precomputation and application costs are balanced with *defmm*.
- For highly non-uniform distributions, the higher memory footprint of the FFT-based techniques compared to the low-rank approximation approaches is less critical than the higher memory requirements of the $MaxDepth$-based approach compared to the $Ncrit$-based one (since the tests on the refined cube cannot be performed for relatively large interpolation order with the low-rank approaches of *dfmm* due to the memory footprint while *defmm* allows to perform these computations). For the uniform cube test case in the high frequency regime for high orders ($K = 64$ and $L \geq 7$), the conclusions are different: the memory footprint of *defmm* is too high for the architecture described in Sect. 7.1.3 while *dfmm* uses less memory in this case and can perform with $K = 64$ and $L = 7$ on this architecture.
- The application order of the **M2L** operator using the FFT-based approach does not impact the performance as it does using the low-rank approaches since the Hadamard products obtained by the FFT-based approach do not benefit from BLAS 2/3 routines. The importance of the application order for *dfmm* is illustrated by the uniform cube test case.

Impact of SIMD for P2P. The implementation of the **P2P** operator is not vectorized in *dfmm*. The implementation of *defmm* exploits the SIMD optimization for the near field part we presented in Sect. 7.5.5.2. Hence, we also wanted to measure the impact of this optimization on the comparison between *defmm* and *dfmm*. Indeed, on the uniform cube, the optimal overall timing for *dfmm* is obtained with a not well balanced ratio of **M2L**/**P2P** costs: approximately 85% of the application timing is here spent by *dfmm* in the near field computation. We then did our tests again with *defmm* but discarding the OpenMP SIMD vectorization (and still without using precomputed **P2P**).

By using smaller $Ncrit$ values than for the results of Figs. 7.31 and 7.32, gains with *defmm* compared to *dfmm* were still observed on the refined cube and the ellipse test cases (e.g. 4 times faster using *defmm* on the refined cube and 2 times faster on the ellipse for $L = 4, K = 0$). On the uniform cube and sphere test case, we ended up with higher application timings with *defmm* than *dfmm-IAblk* in the low-frequency regime and relatively small interpolation orders ($L = 4, K = 0$). For high-frequencies ($L = 4, K = 64$ or $L = 6, K = 64$) on the sphere, we ended up with application timings higher than *dfmm-IAblk* ones but lower than *dfmm-SArcmp* ones. This conclusion is in accordance with our interpretation: *dfmm-IAblk* fully benefits from the uniform distribution (especially in the low-frequency regime) because of the efficient BLAS 3 routines while this is the worst case for *defmm*. The same holds for the uniform cube test case with $L = 4, K = 64$.

In conclusion, by discarding the SIMD computing for **P2P** operations in *defmm*, we obtain similar performances than *dfmm* (better than *dfmm-SArcmp* but worse than *dfmm-IAblk*) on surfacic or volumic uniform distributions. Hence, on its own, the FFT-based techniques exploited in *defmm* provide competitive results compared to the low-rank approaches of *dfmm* on these test cases in terms of application timings. On (highly) non-uniform distributions, *defmm* performs better. Hence, our method is better at handling the particle density variations in the two frequency regimes than *dfmm* but needs to be optimized (with SIMD computing for **P2P** operations) in order to become better than *dfmm* in the most regular distributions. We recall that the results in this section are given *without* the refined horizontal pass of Sect. 7.5.7.4 since *dfmm* does not implement the associated operators.

Conclusion. *defmm* is few sensitive to the particle distribution, for any interpolation order or frequency regime, except for the uniform cube distribution which corresponds to the most difficult

case for the directional methods and particularly for our implementation. However, we still obtained better timings on this distribution with *defmm* than with *dfmm*. In the general case, the main reasons of our gains on the application timings are both the $Ncrit$+DTT+blank passes strategies and the efficient SIMD treatment of the near field. Mostly the precomputation step perfomance is impacted by the FFT-based techniques, allowing to fastly precompute any **M2L** (diagonal) matrix. They also allow to obtain comparative timings with *dfmm* on the **M2L** applications.

Chapter 8

Conclusion and perspectives

The contributions of this thesis are divided into three different parts. The first part was dedicated to a high-level approach for the hierarchical methods, with applications to treecodes, hierarchical matrices and Fast Multipole Methods. In a second part, we described an algebraic approach to deal with the polynomial interpolation problem on the sphere in order to optimize the kernel-explicit FMM for the Helmholtz kernel in the high-frequency regime, namely *hf-fmm*, using quasi-optimal cubature rules on the sphere. Finally, the third part was concerned by the theory, the algorithmic design, the optimization and the validation of our directional FMM library, namely *defmm*.

High-level approach for the hierarchical methods. We developed a general framework, based on the concept of freely generated vector spaces, allowing to represent the hierarchical methods in an abstract way. We came up with effective matrix factorization representations of these methods. Because we are mainly interested in the FMM, we proposed a high-level presentation of the FMM based on simple invariance assumptions on the operators generating the coefficients of the matrix forms of the FMM operators. The main application of this presentation is the general treatment of symmetries appearing in the FMMs dealing with 2^d-trees, which is widely applied in the other parts of this thesis. While the exploitation of the symmetries in a FMM context is not a new thing, we believe that our contribution made it not case specific anymore, allowing to rely directly on assumptions on the FMM formulation rather than on the explicit form of the FMM operators.

Among the perspectives associated to this work, the implementation of a general numerical unified framework for the design of hierarchical methods based on the tools we presented would be a challenging task.

Lebedev rules in *hf-fmm*. To minimize the size of the diagonal **M2L** operators of *hf-fmm* and in order to extend this FMM formulation to the full exploitation of the symmetries described in our high-level presentation of the FMM structure, we proposed to use the Lebedev cubature rules on the sphere for the cubature of the integral appearing in the propagating planewave expansion instead of the product rules usually used in such a context. Our method reduces the complexity of the **M2L** operators to a quasi-optimal one but strongly complicates the evaluation of the interpolations over the sphere needed during the **M2M** / **L2L** applications. To tackle this issue, we developed a purely algebraic approach for the block-diagonalization of the matrix form of the interpolation operator on the sphere. We presented the applications of our method to the product cubature rules and to the Lebedev rules. Using an efficient BLAS-based strategy for the application of the block-diagonalized matrices, we strongly reduced the application timings of the interpolation over the sphere using Lebedev rules. We provided several ideas to further optimize the implementation we are using.

However, our method does not lower the theoretical complexity of the interpolation step, compared to the fast interpolation algorithms using product rules. We thus proposed a switching

strategy, allowing to fully exploit the Lebedev rules at the deepest octree level (to reduce the memory footprint, to accelerate the **M2L** applications and to exploit the entire set of symmetries in the octrees) and the fast **M2M** / **L2L** applications that can be achieved when using product rules at the upper levels (i.e. close to the root). This switching strategy also uses the block-diagonalization we described. To decide on the effective efficiency of the use of the Lebedev rules at the deepest levels and on the maximal application range of such rules in *hf-fmm* while benefiting from numerical gain compared to product rules, one still needs to implement a full realization of *hf-fmm* using the Lebedev rules with the fast interpolation scheme we developed.

The block-diagonalization approach we proposed is based on quite general ideas and can be applied to other operators than the interpolation one, or to any other invariant cubature rules than the Lebedev ones. However, this suffers from a strong constraint: the invariant cubature rules have to be tabulated, which is a difficult task when one wants to obtain quasi-optimal rules. Since the group-based approach we proposed supports the use of FFTs when abelian cyclic groups are considered, it would be very interesting to investigate the design of non-product cubature based on an abelian cyclic group invariance, with possibly better efficiency than the Gauss-Legendre product rules.

defmm. To face up the problem of evaluating oscillatory kernels in the high-frequency regime and on highly non-uniform particle distributions through a FMM algorithm, we proposed our own FMM library, named *defmm*, based on a directional kernel-independent polynomial interpolation approach using FFT techniques. We gave a consistency proof of the interpolation process on interpolation grids using equispaced nodes on well-separated sets. Then, we extended the symmetries presented in the first part of the contributions of this thesis, and already exploited in the *bbfmm* method, to the Fourier domain, leading to Hadamard products with indirections during the **M2L** evaluation and to a reduction of the precomputation cost. We proposed a new general group-based approach for the vectorization of these Hadamard products with indirections, that was however costlier than a simple OpenMP based approach on the vector sizes we considered, even if we exhibited effective cost reductions for larger vector sizes.

We were then concerned by the optimization of the different operators of our library. We introduced a new complex data deinterleaving and vector stacking based method for the **M2M** / **L2L** evaluation that showed significant performance gains in most particle distributions. We developed an efficient SIMD vectorized code for the near field evaluation also based on deinterleaving and a precomputed strategy for static particles. We proposed a new wideband Dual Tree Traversal algorithm exploiting the common but non-standard **M2P** and **P2L** operators and a storage strategy for the multipole expansions that better adapts to the particle distribution. Combined with a precomputed step based on blank tree traversals, we obtained a method with performance few sensitive to the particle distribution. We also investigated the most efficient way of applying the numerous small FFTs required in our interpolation-based FMM.

We finally proposed a numerical comparison between *defmm* and the two best variants of *dfmm* (a state-of-the-art interpolation-based FMM library), exhibiting important performance gains using *defmm*, especially for highly non-uniform distributions. We would now want to compare our implementation with an implementation of the directional FMM presented in [38, 94–96].

There are a lot of possible future work on *defmm*. Our code is supposed to be applied to BEM problems in electromagnetism and we are curretly working on it. This should lead to a comparison of the resolution timings with another hierarchical method, namely the hierarchical matrices for the same BEM problem. Since the same FMM (or hierarchical matrix) is applied many times to different vectors in this context, the overall timings only consider one single precomputation step (for both method) but multiple applications (which differs from the tests we provided in this manuscript, considering only single applications).

We obtained early promising results for the OpenMP-based parallelization of *defmm* in a shared-memory context using tasks. These results only concern the application step of *defmm*. We thus plan to investigate the shared memory parallelization of the precomputation step using tasks on

our blank passes and on the tree building, that should lead to important performance gain. The difficulty when doing so concerns the conflicts when multiple threads write concurrently in the **M2L** tables. We also plan to extend *defmm* to a distributed memory framework, taking into account both the particle distribution and directionality when scattering the data in order to obtain a good load balancing on non-uniform distributions while minimizing the communications.

The N-body problem formulated on *derivatives* of the kernel instead of direct kernel evaluations frequently appears in the integral equation context. Such problem has been tackled when polynomial interpolation is used (see [41, 172]). This feature can be easily added to *defmm*, but a careful numerical stability analysis has to be done first.

Concerning the SIMD vectorization, we plan to extend our near field code based on complex data deinterleaving to mutual interactions (when the input particles are both source and target particles). We also consider to add the evaluation of singular integrals appearing in the Galerkin discretization of integral equation directly in the **P2P** operators, which drastically complicates the vectorization. For the **M2L** application, the adaptation of the interleaving strategy proposed in *pvfmm* [166, 167] on locally uniform 2^d-tree parts may result in performance gain. In addition, an hybrid strategy, combining FFT-techniques for **M2L** interactions at the same tree level (i.e. the method we are currently using) with low-rank approximations when the tree levels of the well-separated cells differ may allow even faster far field evaluations.

The treatment of the symmetries in *defmm* is written for any dimension. We would like to investigate to what extent *defmm* can be efficiently used in more than three dimensions. Indeed, the computation and storage of the **M2L** matrices can become critical in more than three dimensions (for instance, 2320 **M2L** matrices have to be computed per low-frequency tree level in dimension four if the symmetries are not used) and the use of symmetries may at least limit the impact of the dimension on these two points.

Bibliography

[1] Matthias Messner, C++ Chebyshev interpolation-based directional FMM library: *dfmm*, https://github.com/burgerbua/dfmm.

[2] Pierre Marchand, C++ $\mathcal{H}$-matrix library https://github.com/htool-ddm/htool.

[3] Rio Yokota, C++ FMM library: *exafmm*, https://github.com/exafmm/exafmm.

[4] FFTW3 documentation on advanced complex DFTs: http://www.fftw.org/fftw3_doc/Advanced-Complex-DFTs.html.

[5] OpenMP specifications. Available fromhttp://www.openmp.org.

[6] IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2008*, pages 1–70, 2008.

[7] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. 1964. National Bureau of Standards Applied Mathematics Series, 55 For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C.

[8] Emmanuel Agullo, Bérenger Bramas, Olivier Coulaud, Eric Darve, Matthias Messner, and Toru 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.

[9] Lars V. Ahlfors. Complex analysis: An introduction to the theory of analytic functions of one complex variable; 3rd ed. pages xi+331, 1978.

[10] Cory Ahrens and Gregory Beylkin. Rotationally invariant quadratures for the sphere. *Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, 465(2110):3103–3125, 2009. With supplementary data available on-line.

[11] A. Aimi, L. Bassotti, and M. Diligenti. Groups of congruences and restriction matrices. *BIT*, 43(4):671–693, 2003.

[12] A. Aimi and M. Diligenti. Restriction matrices for numerically exploiting symmetry. *Adv. Comput. Math.*, 28(3):201–235, 2008.

[13] A. Aimi, M. Diligenti, and F. Lunardini. Panel clustering method and restriction matrices for symmetric Galerkin BEM. *Numer. Algorithms*, 40(4):355–382, 2005.

[14] M.P. Allen and D.J. Tildesley. *Computer Simulation of Liquids*. Oxford Science Publ. Clarendon Press, 1989.

[15] François Alouges, Matthieu Aussal, and Emile Parolin. Fast Boundary Element Method for acoustics with the Sparse Cardinal Sine Decomposition. *Eur. J. Comput. Mech.*, 26(4):377–393, 2017.

[16] François Alouges and Matthieu Aussal. The sparse cardinal sine decomposition and its application for fast numerical convolution. *Numer. Algorithms*, 70(2):427–448, 2015.

[17] Srinivas Aluru and Fatih E. Sevilgen. Dynamic compressed hypertoctrees with application to the n-body problem. In *Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science*, page 21–33, Berlin, Heidelberg, 1999. Springer-Verlag.

[18] Sivaram Ambikasaran and Eric Darve. An O (N log N) Fast Direct Solver for Partial Hierarchically Semi-Separable Matrices - With Application to Radial Basis Function Interpolation. *J. Sci. Comput.*, 57:477–501, 2013.

[19] Sivaram Ambikasaran and Eric Darve. The Inverse Fast Multipole Method. *ArXiv*, abs/1407.1572, 2014.

[20] Sivaram Ambikasaran, Judith Li, Peter Kitanidis, and Eric Darve. Large-scale stochastic linear inversion using hierarchical matrices. *Computational Geosciences*, 17:913–927, 12 2013.

[21] Congpei An and Siyong Chen. Numerical integration over the unit sphere by using spherical t-design, 2016. https://arxiv.org/abs/1611.02785.

[22] Christopher R. Anderson. An implementation of the fast multipole method without multipoles. *SIAM J. Sci. Statist. Comput.*, 13(4):923–947, 1992.

[23] Andrew Appel. An efficient program for many-body simulation. *SIAM J. Sci. Statist. Comput.*, 6(1):85–103, 1985.

[24] J. Aronsson, I. Jeffrey, and V. Okhmatovski. Generalization of the Barnes-Hut algorithm for the Helmholtz equation in three dimensions. *IEEE Antennas and Wireless Propagation Letters*, 8:425–428, 2009.

[25] Nitin Arora, Aashay Shringarpure, and Richard Vuduc. Direct N-body Kernels for Multicore Platforms. pages 379–387, 09 2009.

[26] Kendall Atkinson. Numerical integration on the sphere. *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 23:332 – 347, 01 1982.

[27] Martin Averseng. Fast discrete convolution in $\mathbb{R}^2$ with radial kernels using non-uniform fast Fourier transform with nonequispaced frequencies. *Numer. Algorithms*, 83(1):33–56, 2020.

[28] Jonas Ballani and Daniel Kressner. Matrices with hierarchical low-rank structures. In *Exploiting hidden structure in matrix computations: algorithms and applications*, volume 2173 of *Lecture Notes in Math.*, pages 161–209. Springer, Cham, 2016.

[29] Gang Bao, Liwei Xu, and Tao Yin. Boundary integral equation methods for the elastic and thermoelastic waves in three dimensions. *Computer Methods in Applied Mechanics and Engineering*, 354:464–486, Sep 2019.

[30] Josh Barnes and Piet Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324(6096):446–449, December 1986.

[31] Joshua E. Barnes. A modified tree code: don't laugh; it runs. *J. Comput. Phys.*, 87(1):161–170, 1990.

[32] Alexander H. Barnett, Jeremy Magland, and Ludvig af Klinteberg. A Parallel Nonuniform Fast Fourier Transform Library Based on an "Exponential of Semicircle" Kernel. *SIAM Journal on Scientific Computing*, 41(5):C479–C504, 2019.

[33] M. Bebendorf. Hierarchical LU Decomposition-based Preconditioners for BEM. *Computing (Wien), v.74, 225-247 (2005)*, 74, 05 2005.

[34] Mario Bebendorf. Approximation of Boundary Element Matrices. *Numerische Mathematik*, 86:565–589, 10 2000.

[35] Mario Bebendorf. *Hierarchical matrices*, volume 63 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 2008. A means to efficiently solve elliptic boundary value problems.

[36] Mario Bebendorf and Sergej Rjasanow. Adaptive Low-Rank Approximation of Collocation Matrices. *Computing*, 70:1–24, 02 2003.

[37] Casper H. L. Beentjes. Quadrature on a spherical surface. Tech. Rep., 2015.

[38] Austin R. Benson, Jack Poulson, Kenneth Tran, Björn Engquist, and Lexing Ying. A parallel directional fast multipole method. *SIAM J. Sci. Comput.*, 36(4):C335–C352, 2014.

[39] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM*, 18(9):509–517, September 1975.

[40] D. Blackston and T. Suel. Highly portable and efficient implementations of parallel adaptive N-body methods. *Proceedings of SC97*, 1997.

[41] Pierre Blanchard. *Fast hierarchical algorithms for the low-rank approximation ofmatrices with applications to materials physics, geostatistics and data analysis*. PhD thesis, Université de Bordeaux, 2017.

[42] Pierre Blanchard, Olivier Coulaud, and Eric Darve. Fast hierarchical algorithms for generating Gaussian random fields. Research Report 8811, Inria Bordeaux Sud-Ouest, December 2015. https://hal.inria.fr/hal-01228519v2/file/RR-8811.pdf.

[43] John A. Board, Ziyad S. Hakura, William D. Elliott, and William T. Rankin. Scalable Variants of Multipole-Accelerated Algorithms for Molecular Dynamics Applications. In *In Proceedings of the 7th Conference on Parallel Processing for Scientific Computing*, pages 295–300. SIAM, 1995.

[44] Steffen Börm. *Efficient numerical methods for non-local operators*, volume 14 of *EMS Tracts in Mathematics*. European Mathematical Society (EMS), Zürich, 2010. $\mathscr{H}^2$-matrix compression, algorithms and analysis.

[45] Alain Bossavit. Boundary Value Problems with Symmetry and Their Approximation by Finite Elements. *SIAM Journal on Applied Mathematics*, 53(5):1352–1380, 1993.

[46] A. Brandt and A. A. Lubrecht. Multilevel matrix multiplication and fast solution of integral equations. *J. Comput. Phys.*, 90(2):348–370, 1990.

[47] Achi Brandt. Multilevel computations of integral transforms and particle interactions with oscillatory kernels. *Comput. Phys. Comm.*, 65(1-2):24–38, 1991.

[48] D. Brunner, M. Junge, P. Rapp, M. Bebendorf, and L. Gaul. Comparison of the fast multipole method with hierarchical matrices for the Helmholtz-BEM. *CMES Comput. Model. Eng. Sci.*, 58(2):131–159, 2010.

[49] Jonathan Bull and Stefan Engblom. Distributed and Adaptive Fast Multipole Method In Three Dimensions, 2020. https://arxiv.org/pdf/2002.04894.pdf.

[50] Steffen Börm. Directional H2-matrix compression for high-frequency problems. *Numer. Linear Algebra Appl.*, 24(6), 2017.

[51] Steffen Börm and Lars Grasedyck. Low-Rank Approximation of Integral Operators by Interpolation. *Computing*, 72:325–332, 06 2004.

[52] Steffen Börm and Jens M. Melenk. Approximation of the high-frequency helmholtz kernel by nested directional interpolation: error analysis. *Numerische Mathematik*, 137(1):1–34, Mar 2017.

[53] R. Capuzzo-Dolcetta and P. Miocchi. A comparison between the fast multipole algorithm and the tree-code to evaluate gravitational forces in 3-d. *Journal of Computational Physics*, 143(1):29–48, Jun 1998.

[54] Q. Carayol and F. Collino. Error estimates in the fast multipole method for scattering problems. Part 2: Truncation of the Gegenbauer series. *M2AN Math. Model. Numer. Anal.*, 39(1):183–221, 2004.

[55] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM J. Sci. Statist. Comput.*, 9(4):669–686, 1988.

[56] Cris Cecka and Eric Darve. Fourier-based Fast Multipole Method for the Helmholtz equation. *SIAM J. Sci. Comput.*, 35(1):A79–A103, 2013.

[57] Cristopher Cecka, Pierre-David Létourneau, and Eric Darve. Fast multipole method using the Cauchy integral formula. *Lect. Notes Comput. Sci. Eng.*, 82:127–144, 2012. Numerical analysis of multiscale computations, Springer, Heidelberg.

[58] Stéphanie Chaillat. *Fast Multipole Methodfor 3-D elastodynamic boundary integral equations.Application to seismic wave propagation*. PhD thesis, Paris Est, 2008.

[59] Stéphanie Chaillat, Marc Bonnet, and Jean-François Semblat. A Fast Multipole Method formulation for 3D elastodynamics in the frequency domain. *Comptes Rendus Mecanique*, 335, 11 2007.

[60] Stéphanie Chaillat and Francis Collino. A wideband fast multipole method for the Helmholtz kernel: theoretical developments. *Comput. Math. Appl.*, 70(4):660–678, 2015.

[61] Stéphanie Chaillat, Samuel Groth, and Adrien Loseille. Metric-based anisotropic mesh adaptation for 3D acoustic boundary element methods. *Journal of Computational Physics*, 372:473–499, 11 2018.

[62] Tony F. Chan and Per Christian Hansen. Low-rank revealing QR factorizations. *Numerical Linear Algebra with Applications*, 1(1):33–44, 1994.

[63] A. Chandramowlishwaran, S. Williams, L. Oliker, I. Lashuk, G. Biros, and R. Vuduc. Optimizing and tuning the fast multipole method for state-of-the-art multicore architectures. In *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, pages 1–12, 2010.

[64] 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(4):045007, Apr 2018.

[65] H. Cheng, Leslie Greengard, and Vladimir Rokhlin. A fast adaptive multipole algorithm in three dimensions. *J. Comput. Phys.*, 155(2):468–498, 1999.

[66] Hongwei Cheng, William Y. Crutchfield, Zydrunas Gimbutas, Leslie F. Greengard, J. Frank Ethridge, Jingfang Huang, Vladimir Rokhlin, Norman Yarvin, and Junsheng Zhao. A wideband fast multipole method for the Helmholtz equation in three dimensions. *J. Comput. Phys.*, 216(1):300–325, 2006.

[67] W. Chew, E. Michielssen, J. M. Song, and J. M. Jin. *Fast and Efficient Algorithms in Computational Electromagnetics*. 2001. Artech House, Norwood, MA.

[68] Min Hyung Cho and Wei Cai. A wideband fast multipole method for the two-dimensional complex Helmholtz equation. *Comput. Phys. Comm.*, 181(12):2086–2090, 2010.

[69] I. Chowdhury and V. Jandhyala. Integration and interpolation based on fast spherical transforms for the multilevel fast multipole method. *Microwave Optical Technol. Lett.*, 48:1961–1964, 2006.

[70] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method for the wave equation: a pedestrian prescription. *IEEE Antennas and Propag. Mag.*, 35:7–12, 1993.

[71] Ronald Cools. Constructing cubature formulae: the science behind the art. *Acta Numer.*, 6:1–54, 1997.

[72] Olivier Coulaud, Pierre Fortin, and Jean Roman. High performance BLAS formulation of the multipole-to-local operator in the Fast Multipole Method. *Journal of Computational Physics*, 227(3):1836–1862, 2008.

[73] Olivier Coulaud, Pierre Fortin, and Jean Roman. High-performance BLAS formulation of the adaptive Fast Multipole Method. *Mathematical and Computer Modelling*, 51(3-4):177–188, February 2010.

[74] Marion Darbas, Eric Darrigrand, and Yvon Lafranche. Combining Analytic Preconditioner and Fast Multipole Method for the 3-D Helmholtz Equation. *Journal of Computational Physics*, 236:289–316, 2013.

[75] Eric Darve. *Méthodes multipôles rapides : résolution des équations de Maxwell par formulations intégrales*. PhD thesis, Université Paris 6, 1999.

[76] Eric Darve. The fast multipole method: numerical implementation. *J. Comput. Phys.*, 160(1):195–240, 2000.

[77] Eric Darve. The fast multipole method I: Error analysis and asymptotic complexity. *Numer. Anal.*, 38:98–128, 2000.

[78] Eric Darve and Pascal Havé. Efficient fast multipole method for low-frequency scattering. *J. Comput. Phys.*, 197(1):341–363, 2004.

[79] Walter Dehnen. A hierarchical O(N) force calculation algorithm. *J. Comput. Phys.*, 179(1):27–42, 2002.

[80] Walter Dehnen. A fast multipole method for stellar dynamics. *Computational Astrophysics and Cosmology*, 1(1), Sep 2014.

[81] James Demmel, Laura Grigori, Mark Hoemmen, and Julien Langou. Communication-optimal parallel and sequential QR and LU factorizations. *SIAM J. Sci. Comput.*, 34(1):A206–A239, 2012.

[82] Hong-Qiang Ding, Naoki Karasawa, and William A. Goddard. Atomic level simulations on a million particles: The cell multipole method for Coulomb and London nonbond interactions. *The Journal of Chemical Physics*, 97(6):4309–4315, 1992.

[83] Leon Dommelen and Elke Rundensteiner. Fast, adaptive summation of point forces in the two-dimensional Poisson equation. *Journal of Computational Physics*, 83:126–147, 07 1989.

[84] Jack Dongarra, Jeremy Croz, Sven Hammarling, and Iain Duff. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software (TOMS)*, 16:1–17, 03 1990.

[85] Jack Dongarra and Francis Sullivan. Guest Editors Introduction to the top 10 algorithms. *Computing in Science & Engineering*, 2:22–23, 02 2000.

[86] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Richard J. Hanson. An Extended Set of FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Softw.*, 14(1):1–17, March 1988.

[87] Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst. *Numerical Linear Algebra for High-Performance Computers*. Society for Industrial and Applied Mathematics, 1998.

[88] John Dubinski. A parallel tree code. *New Astronomy*, 1(2):133–147, Oct 1996.

[89] A. Dutt, M. Gu, and V. Rokhlin. Fast algorithms for polynomial interpolation, integration, and differentiation. *SIAM J. Numer. Anal.*, 33(5):1689–1711, 1996.

[90] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14(6):1368–1393, 1993.

[91] Benedikte Elbel and Gabriele Steidl. Fast Fourier transforms for nonequispaced data. In *Approximation theory IX, Vol. 2 (Nashville, TN, 1998)*, Innov. Appl. Math., pages 39–46. Vanderbilt Univ. Press, Nashville, TN, 1998.

[92] William Dewey Elliott. *Multipole Algorithms for Molecular Dynamics Simulation on High Performance Computers*. PhD thesis, 1995. Technical Report 95-003, Duke University, Department of Electrical Engineering, Doctoral dissertation.

[93] Stefan Engblom. On well-separated sets and fast multipole methods. *Applied Numerical Mathematics*, 61(10):1096–1102, Oct 2011.

[94] Björn Engquist and Lexing Ying. Fast directional multilevel algorithms for oscillatory kernels. *SIAM Journal on Scientific Computing*, 29:1710–1737, 2007.

[95] Björn Engquist and Lexing Ying. A fast directional algorithm for high frequency acoustic scattering in two dimensions. *Commun. Math. Sci*, 7:327–345, 2009.

[96] Björn Engquist and Lexing Ying. Fast directional algorithms for the Helmholtz kernel. *J. Comput. Appl. Math.*, 234:1851–1859, 2010.

[97] M. A. Epton and B. Dembart. Multipole translation theory for the three-dimensional Laplace and Helmholtz equations. *SIAM Journal on Scientific Computing*, 16(4):865–897, 1995.

[98] O. Ergul and L. Gurel. Optimal interpolation of translation operator in multilevel fast multipole algorithm. *IEEE Trans. Antennas and Propagation*, 54:3822–3826, 2006.

[99] Alexandre Ern and Jean-Luc Guermond. *Theory and Practice of Finite Elements*, volume 159. Springer-Verlag New York, 2004.

[100] Klaas Esselink. A comparison of algorithms for long-range interactions. *Computer physics communications*, 87(3):375–395, 1995.

[101] P. P. Ewald. Die berechnung optischer und elektrostatischer gitterpotentiale. *Annalen der Physik*, 369(3):253–287, 1921.

[102] Hualong Feng, Amlan Barua, Shuwang Li, and Xiaofan Li. A Parallel Adaptive Treecode Algorithm for Evolution of Elastically Stressed Solids. *Communications in Computational Physics*, 15(2):365–387, 2014.

[103] William Fong and Eric Darve. The black-box fast multipole method. *J. Comput. Phys.*, 228(23):8712–8725, 2009.

[104] Pierre Fortin, Evangélie Athanassoula, and Jean-Charles Lambert. Comparisons of different codes for galactic N-body simulations. *Astronomy and Astrophysics - A&A*, 531:A120, July 2011.

[105] Message P Forum. MPI: A Message-Passing Interface Standard. Technical report, USA, 1994.

[106] Daan Frenkel and Berend Smit. *Understanding molecular simulation : from algorithms to applications. 2nd ed*, volume 50. 01 1996.

[107] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[108] Mariano Gasca and Tomas Sauer. Polynomial interpolation in several variables. *Advances in Computational Mathematics*, 12(4):377–410, 2000.

[109] Maximilian Gaß, Kathrin Glau, Mirco Mahlstedt, and Maximilian Mair. Chebyshev interpolation for parametric option pricing. *Finance and Stochastics*, 22(3):701–731, Apr 2018.

[110] Ladnor D. Geissinger and D. Kinch. Representations of the hyperoctahedral groups. *J. Algebra*, 53(1):1–20, 1978.

[111] Klaus Gibermann. Multilevel Approximation of Boundary Integral Operators. *Computing*, 67(3):183–207, November 2001.

[112] Kathrin Glau and Mirco Mahlstedt. Improved error bound for multivariate Chebyshev polynomial interpolation. *International Journal of Computer Mathematics*, 96(11):2302–2314, Apr 2019.

[113] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd ed.)*. 1996. Baltimore: Johns Hopkins.

[114] L. Greengard, Jingfang Huang, V. Rokhlin, and S. Wandzura. Accelerating fast multipole methods for the Helmholtz equation at low frequencies. *IEEE Computational Science and Engineering*, 5(3):32–38, 1998.

[115] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987.

[116] Leslie Greengard. *The rapid evaluation of potential fields in particle systems.* MIT Press, 1988.

[117] Leslie Greengard and Jingfang Huang. A New Version of the Fast Multipole Method for Screened Coulomb Interactions in Three Dimensions. *Journal of Computational Physics*, 180:642–658, 08 2002.

[118] Leslie Greengard and Vladimir Rokhlin. The rapid evaluation of potential fields in three dimensions. *Lecture Notes in Math.*, 1360:121–141, 1988.

[119] Leslie Greengard and John Strain. The Fast Gauss Transform. *SIAM J. Sci. Stat. Comput.*, 12(1):79–94, January 1991.

[120] Laura Grigori, James W. Demmel, and Hua Xiang. CALU: a communication optimal LU factorization algorithm. *SIAM J. Matrix Anal. Appl.*, 32(4):1317–1350, 2011.

[121] Martin Gutknecht. *A Brief Introduction to Krylov Space Methods for Solving Linear Systems*, pages 53–62. 01 2007.

[122] W. Hackbusch. The Panel Clustering Method for BEM. In Günther Kuhn and Herbert Mang, editors, *Discretization Methods in Structural Mechanics*, pages 299–306, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[123] W. Hackbusch. A sparse matrix arithmetic based on $\mathscr{H}$-matrices. I. Introduction to $\mathscr{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[124] W. Hackbusch, B. Khoromskij, and S. A. Sauter. On $\mathscr{H}^2$-matrices. In *Lectures on applied mathematics (Munich, 1999)*, pages 9–29. Springer, Berlin, 2000.

[125] Wolfgang Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*, volume 49. 12 2015.

[126] Wolfgang Hackbusch and Steffen Börm. H2-matrix approximation of integral operators by interpolation. *Applied Numerical Mathematics*, 43:129–143, 2002.

[127] Wolfgang Hackbusch, Lars Grasedyck, and Steffen Börm. An introduction to hierarchical matrices. In *Proceedings of EQUADIFF, 10 (Prague, 2001)*, volume 127, pages 229–241, 2002.

[128] Wolfgang Hackbusch, Christian Lage, and S Sauter. *On the Efficient Realization of Sparse Matrix Techniques for Integral Equations with Focus on Panel Clustering, Cubature and Software Design Aspects*, pages 51–75. 01 1997.

[129] Wolfgang Hackbusch and Z. Nowak. On the fast multiplication in the boundary element method by panel clustering. *Numerische Mathematik*, 54:463–491, 07 1989.

[130] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, Jan 2011.

[131] Tsuyoshi Hamada, Rio Yokota, Keigo Nitadori, Tetsu Narumi, Kenji Yasuoka, and Makoto Taiji. 42 TFlops hierarchical N-body simulations on GPUs with applications in both astrophysics and turbulence. *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 11 2009.

[132] R. H. Hardin and N. J. A. Sloane. McLaren's improved snub cube and other new spherical designs in three dimensions. *Discrete Comput. Geom.*, 15(4):429–441, 1996.

[133] 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*, pages 42–42, 2002.

[134] Bhanu Hariharan and Srinivas Aluru. Efficient parallel algorithms and software for compressed octrees with applications to hierarchical methods. *Parallel Computing*, 31:311–331, 03 2005.

[135] Michael Larkin Hastriter, Shinichiro Ohnuki, and Weng Cho Chew. Error control of the translation operator in 3D MLFMA. *Microwave Optical Technol. Lett.*, 37(3):184–188, 2003.

[136] J.L. Hennessy, D.A. Patterson, and K. Asanović. *Computer Architecture: A Quantitative Approach*. Computer Architecture: A Quantitative Approach. Elsevier Science, 2012.

[137] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. 1988. Institute of Physics Publishing.

[138] Marcus Holm, Stefan Engblom, Anders Goude, and Sverker Holmgren. Dynamic Autotuning of Adaptive Fast Multipole Methods on Hybrid Multicore CPU and GPU Systems. *SIAM Journal on Scientific Computing*, 36(4):C376–C399, Jan 2014.

[139] Y. Hu and S. L. Johnsson. Implementing O(N) N-body algorithms efficiently in data-parallel languages. *Scientific Programming*, 5(4):337–364, 1996.

[140] Jingfang Huang, Jun Jia, and Bo Zhang. FMM-Yukawa: An adaptive fast multipole method for screened Coulomb interactions. *Computer Physics Communications*, 180(11):2331–2338, 2009.

[141] Bayram Ali Ibrahimoglu. Lebesgue functions and Lebesgue constants in polynomial interpolation. *J. Inequal. Appl.*, (93), 2016.

[142] Intel. A Guide to Vectorization with Intel C++ Compilers. 2012.

[143] Mustafa Abdul Jabbar, Mohammed A. Al Farhan, Noha Al-Harthi, Rui Chen, Rio Yokota, Hakan Bagci, and David E. Keyes. Extreme Scale FMM-Accelerated Boundary Integral Equation Solver for Wave Scattering. *CoRR*, abs/1803.09948, 2018.

[144] Rüdiger Jakob-Chien and Bradley K. Alpert. A Fast Spherical Filter with Uniform Resolution. *J. Comput. Phys.*, 136(2):580–584, September 1997.

[145] Atsushi Kawai, Toshiyuki Fukushige, Junichiro Makino, and Makoto Taiji. GRAPE-5: A special-purpose computer for N-body simulations. *Publ. of the Astronomical Society of Japan*, 52:659–676, 2000.

[146] Robert Krasny and Lei Wang. A treecode based on barycentric Hermite interpolation for electrostatic particle interactions. *Computational and Mathematical Biophysics*, 7(1):73 – 84, 2019.

[147] Sanjeev Krishnan and Laxmikant V. Kalé. A parallel adaptive fast multipole algorithm for N-body problems. *International Conference on Parallel Processing*, page 46–50, 1995.

[148] V. I. Krylov. Convergence of algebraic interpolation with respect to roots of Čebyšev's polynomial for absolutely continuous functions and functions of bounded variation. *Dokl. Akad. Nauk SSSR (N.S.)*, 107:362–365, 1956. (Russian).

[149] Jun Lai, Sivaram Ambikasaran, and Leslie Greengard. A Fast Direct Solver for High Frequency Scattering from a Large Cavity in Two Dimensions. *ArXiv*, abs/1404.3451, 2014. https://arxiv.org/pdf/1404.3451.pdf.

[150] Benoit Lange and Pierre Fortin. Parallel dual tree traversal on multi-core and many-core architectures for astrophysical N-body simulations. In *20th International Conference Euro-Par 2014 Parallel Processing*, volume 8632 of *Lecture Notes in Computer Science*, pages 716–727, Porto, Portugal, August 2014. Springer.

[151] M. Harper Langston. *An Adaptive Fast Multipole Method-Based PDE Solver in Three Dimensions.* PhD thesis, 2012.

[152] Ilya Lashuk, Aparna Chandramowlishwaran, M. Harper Langston, Tuan-Anh Nguyen, Rahul Sampath, Aashay Shringarpure, Richard Vuduc, Lexing Ying, Denis Zorin, and George Biros. A Massively Parallel Adaptive Fast Multipole Method on Heterogeneous Architectures. *Communications of the ACM*, 55:101–109, 05 2012.

[153] V. I. Lebedev and D. N. Laĭkov. A quadrature formula for a sphere of the 131st algebraic order of accuracy. (Russian). *Dokl. Akad. Nauk*, 366(6):741–745, 1999.

[154] V. I. Lebedev and A. L. Skorokhodov. Quadrature formulas for a sphere of orders 41,47 and 53. (Russian). *Dokl. Akad. Nauk*, 324(3):519–524, 1992. translation in Russian Acad. Sci. Dokl. Math. 45 (1992), no. 3, 587–592 (1993).

[155] Vyacheslav Ivanovich Lebedev. A type of quadrature formulas of increased algebraic accuracy for the sphere. (Russian). *Dokl. Akad. Nauk SSSR*, 231(1):32–34, 1976.

[156] Vyacheslav Ivanovich Lebedev. Quadratures on the sphere. (Russian). *Ž. Vyčisl. Mat i Mat. Fiz.*, 16(2):293–306, 1976.

[157] Vyacheslav Ivanovich Lebedev. Quadrature formulas for the sphere of 25th to 29th order accuracy. (Russian). *Sibirsk. Mat. Ž.*, 18(1):132–142, 1977.

[158] Vyacheslav Ivanovich Lebedev. A quadrature formula for a sphere that is the 59th algebraic order of accuracy. (Russian). *Dokl. Akad. Nauk*, 338(4):454–456, 1994. translation in Russian Acad. Sci. Dokl. Math. 50 (1995), no. 2, 283–286.

[159] David Lee. Fast Multiplication of a Recursive Block Toeplitz Matrix by a Vector and Its Application. *J. Complex.*, 2(4):295–305, December 1986.

[160] J. Y. Lee and L. Greengard. The type 3 nonuniform FFT and its applications. *Journal of Computational Physics*, 206(1):1–5, 2005.

[161] S. Li, J. Trevelyan, W. Zhang, and D. Wang. Accelerating isogeometric boundary element analysis for 3-dimensional elastostatics problems through black-box fast multipole method with proper generalized decomposition. *Internat. J. Numer. Methods Engrg.*, 114(9):975–998, 2018.

[162] Y.J. Liu and N. Nishimura. The fast multipole boundary element method for potential problems: A tutorial. *Engineering Analysis with Boundary Elements*, 30:371–381, 05 2006.

[163] Benoît Lizé. *Résolution directe rapide pour les éléments finis de frontière en électromagnétisme et acoustique : H-Matrices. Parallélisme et applications industrielles.* PhD thesis, 2014.

[164] Pierre-David Létourneau, Cris Cecka, and Eric Darve. Cauchy fast multipole method for general analytic kernels. *SIAM J. Sci. Comput.*, 36(2):A396–A426, 2014.

[165] Junichiro Makino. Yet another fast multipole method without multipoles-pseudoparticle multipole method. *Journal of Computational Physics*, 151(2):910–920, 1999.

[166] Dhairya Malhotra and George Biros. PVFMM: A Parallel Kernel Independent FMM for Particle and Volume Potentials. *Commun. Comput. Phys.*, 18(3):808–830, 2015.

[167] Dhairya Malhotra and George Biros. Algorithm 967: A Distributed-Memory Fast Multipole Method for Volume Potentials. *ACM Trans. Math. Software*, 43(2), 2016.

[168] William March, Bo Xiao, and George Biros. ASKIT: Approximate Skeletonization Kernel-Independent Treecode in High Dimensions. *SIAM Journal on Scientific Computing*, 37, 10 2014.

[169] Pierre Marchand. *Domain decomposition method and boundary integral equations.* PhD thesis, Sorbonne Université, 2020.

[170] P. Martinsson and Vladimir Rokhlin. An Accelerated Kernel-Independent Fast Multipole Method in One Dimension. *Society for Industrial and Applied Mathematics*, 29:1160–1178, 01 2007.

[171] A. D. McLaren. Optimal numerical integration on a sphere. *Discrete Comput. Geom.*, 17:361–383, 1963.

[172] Matthias Messner. *Fast Boundary Element Methods in Acoustics*. PhD thesis, Technischen Universität Graz, 2012.

[173] Matthias Messner, Bérenger Bramas, Olivier Coulaud, and Eric Darve. Optimized M2L kernels for the Chebyshev interpolation based fast multipole method. 2012. https://arxiv.org/abs/1210.7292.

[174] Matthias Messner, Martin Schanz, and Eric Darve. Fast directional multilevel summation for oscillatory kernels based on Chebyshev interpolation. *J. Comput. Phys.*, 231(4):1175–1196, 2012.

[175] Bernhard Mößner and Ulrich Reif. Error bounds for polynomial tensor product interpolation. *Computing*, 86:185–197, 10 2009.

[176] 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(3):713–735, May 1994.

[177] K. Nabors, J. Phillips, F. T. Korsmeyer, and J. White. Multipole and precorrected-FFT accelerated iterative methods for solving surface integral formulations of three-dimensional Laplace problems. In *Domain-based parallelism and problem decomposition methods in computational science and engineering*, pages 193–215. SIAM, Philadelphia, PA, 1995.

[178] F.W.J. Olver, D.W. Lozier, Boisvert, R.F., and C.W. Clark. *NIST Handbook of Mathematical Functions*. 2010. Cambridge University Press, Cambridge.

[179] Valentin Pavlov, Nikola Andonov, and Georgi Kremenliev. Porting and Verification of ExaFMM Library in MIC Architecture. February 2014.

[180] G. Peyré. *L'algèbre discrète de la transformée de Fourier: niveau M1*. Mathématiques à l'université : cours et exercices corrigés. Ellipses, 2004.

[181] Rodrigo B. Platte, Lloyd N. Trefethen, and Arno B. J. Kuijlaars. Impossibility of fast stable approximation of analytic functions from equispaced samples. *SIAM Review*, 53(2):308–318, 2011.

[182] Hadi Pouransari and Eric Darve. Optimizing the Adaptive Fast Multipole Method for Fractal Sets. *SIAM Journal on Scientific Computing*, 37:A1040–A1066, 03 2015. https://arxiv.org/abs/1508.02666.

[183] W.T. Rankin. Efficient parallel implementations of multipole based N-body algorithms. PhD. Dissertation, Duke University, Department of Electrical Engineering, 1999.

[184] V. Rokhlin. Diagonal Forms of Translation Operators for Helmholtz Equation in Three Dimensions. *Appl. Comput. Harmon. Anal. 1, 82–93 (1993).*, 1:32, 03 1992.

[185] Youcef Saad and Martin H. Schultz. Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Math. Comp.*, 44(170):417–424, 1985.

[186] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.

[187] Stefan A. Sauter and Christoph Schwab. *Boundary element methods*, volume 39 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2011. Translated and expanded from the 2004 German original.

[188] Dennis Schobert and Thomas Eibert. A multilevel interpolating fast integral solver with fast Fourier Transform acceleration. *Symposium Digest - 20th URSI International Symposium on Electromagnetic Theory, EMTS 2010*, pages 520 – 523, 09 2010.

[189] Jean-Pierre Serre. *Linear representations of finite groups.* Springer-Verlag, New York-Heidelberg, 1977. Translated from the second French edition by Leonard L. Scott, Graduate Texts in Mathematics, Vol. 42.

[190] F. E. Sevilgen, S. Aluru, and N. Futamura. A provably optimal, distribution-independent parallel fast multipole method. In *Proceedings 14th International Parallel and Distributed Processing Symposium. IPDPS 2000*, pages 77–84, 2000.

[191] G. F. Smith. Projection operators for symmetric regions. *Arch. Rational Mech. Anal.*, 54:161–174, 1974.

[192] Simon Smith. Lebesgue constants in polynomial interpolation. *Annales Mathematicae et Informaticae*, 33:109–123, 01 2006.

[193] Sergueï Lvovitch Sobolev. Cubature formulas on the sphere which are invariant under transformations of finite rotation groups. *Dokl. Akad. Nauk SSSR*, 146:310–313, 1962. (Russian).

[194] Marina Spivak, Shravan Veerapaneni, and Leslie Greengard. The Fast Generalized Gauss Transform. *SIAM J. Scientific Computing*, 32:3092–3107, 01 2010.

[195] Olaf Steinbach. *Numerical approximation methods for elliptic boundary value problems.* Springer, New York, 2008. Finite and boundary elements, Translated from the 2003 German original.

[196] Xiaobai Sun and Nikos P. Pitsianis. A Matrix Version of the Fast Multipole Method. *SIAM Review*, 43(2):289–300, 2001.

[197] Guillaume Sylvand. *La méthode multipôle rapide en électromagnétisme : performances, parallèlisation, applications.* PhD thesis, École des ponts et chaussées, 2002.

[198] Toru Takahashi, Chao Chen, and Eric Darve. Parallelization of the inverse fast multipole method with an application to boundary element method. *Comput. Phys. Commun.*, 247:106975, 14, 2020.

[199] Toru Takahshi. An interpolation-based fast-multipole accelerated boundary integral equation method for the three-dimensional wave equation. *Journal of Computational Physics*, 258, 01 2014.

[200] Peter Thoman, Kiril Dichev, Thomas Heller, Roman Iakymchuk, Xavier Aguilar, Khalid Hasanov, Philipp Gschwandtner, Pierre Lemarinier, Stefano Markidis, Herbert Jordan, Thomas Fahringer, Kostas Katrinis, Erwin Laure, and Dimitrios S. Nikolopoulos. A taxonomy of task-based parallel programming technologies for high-performance computing. *The Journal of Supercomputing*, 74(4):1422–1434, 2018.

[201] Anna-Karin Tornberg and Leslie Greengard. A fast multipole method for the three-dimensional Stokes equations. *Journal of Computational Physics*, 227(3):1613–1619, 2008.

[202] L. N. Trefethen and J. A. C. Weideman. Two Results on Polynomial Interpolation in Equally Spaced Points. *J. Approx. Theory*, 65(3):247–260, June 1991.

[203] L.N. Trefethen and D. Bau. *Numerical linear algebra.* 1997. SIAM.

[204] Paul Tsuji and Lexing Ying. A fast directional algorithm for high-frequency electromagnetic scattering. *J. Comput. Phys.*, 230:5471–5487, 2011.

[205] Nathan Vaughn, Leighton Wilson, and Robert Krasny. A GPU-Accelerated Barycentric Lagrange Treecode. 2020. https://arxiv.org/abs/2003.01836.

[206] H. Wallen, S. Jarvenpaa, P. Yla-Oijala, and J. Sarvas. Broadband Müller-MLFMA for Electromagnetic Scattering by Dielectric Objects. *IEEE Transactions on Antennas and Propagation*, 55(5):1423–1430, 2007.

[207] Haitao Wang, Ting Lei, Jin Li, Jingfang Huang, and Zhenhan Yao. A parallel fast multipole accelerated integral equation scheme for 3D Stokes equations. *International Journal for Numerical Methods in Engineering*, 70:812 – 839, 05 2007.

[208] Jun Wang and Leslie Greengard. An Adaptive Fast Gauss Transform in Two Dimensions. *SIAM Journal on Scientific Computing*, 40(3):A1274–A1300, Jan 2018.

[209] Lei Wang, Krasny, Robert, and Svetlana Tlupova. A kernel-independent tree-code based on barycentric Lagrange interpolation. 2019. https://arxiv.org/abs/1902.02250.

[210] Ruoxi Wang, Chao Chen, Jonghyun Lee, and Eric Darve. PBBFMM3D: a Parallel Black-Box Fast Multipole Method for Non-oscillatory Kernels. 2019. https://arxiv.org/abs/1903.02153.

[211] M. S. Warren and J. K. Salmon. A parallel hashed oct-tree N-body algorithm. In *Supercomputing '93:Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*, pages 12–21, 1993.

[212] Michael S. Warren and John K. Salmon. A Portable Parallel Particle Program. *Computer Physics Communications*, 87:266–290, 1995.

[213] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM*, 52(4):65–76, April 2009.

[214] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra with Applications*, 17(6):953–976, 2010.

[215] Jinyou Xiao, Wenjing Ye, Yaxiong Cai, and Jun Zhang. Precorrected FFT accelerated BEM for large-scale transient elastodynamic analysis using frequency-domain approach. *Internat. J. Numer. Methods Engrg.*, 90(1):116–134, 2012.

[216] Zhao Yan, Jun Zhang, and Wenjing Ye. Rapid solution of 3-D oscillatory elastodynamics using the pFFT accelerated BEM. *Engineering Analysis with Boundary Elements*, 34:956–962, 11 2010.

[217] N. Yarvin and V. Rokhlin. Generalized Gaussian quadratures and singular value decompositions of integral operators. *SIAM J. Sci. Comput.*, 20(2):699–718, 1998.

[218] Lexing Ying, George Biros, and Denis Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *J. Comput. Phys.*, 196(2):591–626, 2004.

[219] Lexing Ying, George Biros, Denis Zorin, and Harper Langston. A new parallel kernel-independent fast multipole method. *16th ACM/IEEE Conference on Supercomputing*, page 591–626, 2003.

[220] Rio Yokota. An FMM based on dual tree traversal for many-core architectures. *CoRR*, abs/1209.3516, 2012.

[221] Rio Yokota and Lorena A. Barba. Treecode and Fast Multipole Method for N-Body Simulation with CUDA. *GPU Computing Gems Emerald Edition*, page 113–132, 2011.

[222] Rio Yokota and Lorena A Barba. A tuned and scalable fast multipole method as a preeminent algorithm for exascale systems. *The International Journal of High Performance Computing Applications*, 26(4):337–346, Jan 2012.

[223] Bo Zhang, Jingfang Huang, Nikos P. Pitsianis, and Xiaobai Sun. Revision of FMM–Yukawa: An adaptive fast multipole method for screened Coulomb interactions. *Computer Physics Communications*, 181(12):2206–2207, 2010.

[224] Bo Zhang, Jingfang Huang, Nikos P. Pitsianis, and Xiaobai Sun. A Fourier-series-based kernel-independent fast multipole method. *J. Comput. Phys.*, 230(15):5807–5821, 2011.

[225] Zhiqiang Wang, J. Lupo, A. McKenney, and R. Pachter. Large Scale Molecular Dynamics Simulations with Fast Multipole Implementations. In *SC '99: Proceedings of the 1999 ACM/IEEE Conference on Supercomputing*, pages 56–56, 1999.

[226] Bruno P. Zimmermann. On finite groups acting on spheres and finite subgroups of orthogonal groups. *Siberian Electronic Mathematical Reports*, 9(1), 2011.

Appendices

Appendix A

Tensor products

A.1 Definition and notations

Let V and W be two vector spaces over the same commutative field. The tensor product of V and W, denoted by $V \otimes W$ is defined as in the following.

Theorem A.1.1. *$V \otimes W$ is the space (unique up to isomorphism) such that there exists a bilinear application $\otimes : V \times W \to V \otimes W$ such that for any vector space X and any application $f : V \times W \to X$, $\exists!\, g : V \otimes W \to X$ such that*

$$\forall v \in V, w \in W,\ f(v,w) = g(V \otimes W).$$

This space is unique up to isomorphism.

A.2 Kronecker's product

Let $M, N, P, Q \in \mathbb{N}^*$. Let $A \in \mathbb{C}^{M \times N}$ and $B \in \mathbb{C}^{P \times Q}$ be two matrices over the same field with possibly different dimensions. The Kronecker's product of A and B is a particular case of the general tensor product and refers to the linear map $A \otimes B \in \mathbb{C}^{(MP)\times(NQ)}$ defined by

$$(A \otimes B)_{P(i-1)+k,Q(j-1)+l} := (A)_{i,j}(B)_{k,l}.$$

This corresponds to the block matrix

$$A \otimes B = \begin{bmatrix} (A)_{1,1}B & \dots & (A)_{1,N}B \\ \vdots & \ddots & \vdots \\ (A)_{M,1}B & \dots & (A)_{M,N}B \end{bmatrix}$$

This matrix can be interpreted as a linear map from the tensor space $\mathbb{C}^N \otimes \mathbb{C}^Q$ to $\mathbb{C}^M \otimes \mathbb{C}^P$.

More generaly, let $d \in \mathbb{N}^*$ and $\mathbb{A} := \{A_i \in \mathbb{C}^{M_i \times N_i} \mid M_i, N_i \in \mathbb{N}^*, i \in [\![1,d]\!]\}$. The Kronecker's product of the matrices of $\mathbb{A}$ refers to

$$\bigotimes_{i \in [\![1,d]\!]} A_i := A_1 \otimes \ldots \otimes A_d.$$

Notice that the order of the tensor products matters. Obviously we have $\bigotimes_{i \in [\![1,d]\!]} A_i \in \mathbb{C}^{\left(\Pi_{i \in [\![1,d]\!]} M_i\right) \times \left(\Pi_{i \in [\![1,d]\!]} N_i\right)}$.

If $M_i = M_1$, $N_i = N_1$, $A_i = A_1\ \forall i \in [\![2,d]\!]$, then the notation $A_1^{\otimes d}$ can be substituted to $\otimes_{i \in [\![1,d]\!]} A_i$.

Appendix B

Examples of FMM formulations

B.1 *kifmm*

B.1.1 Anderson's method

The use of integral representation formulas to obtain the multipole expansions in [22] paved the way to an important family of kernel-independent methods. The idea of the Anderson's method, presented in [22], is to represent the far field as the solution of an exterior Dirichlet problem on a ball encompassing the particles thanks to the Poisson formula. The algorithm in [22] relies on a discretization of the integral involved in this new problem over the sphere using appropriate cubature rules. The potential induced by a group of particles is represented on the cubature nodes only and can be evaluated on any point by simply evaluating this cubature. For a more complete description and possible optimizations of the Anderson's method, see [165]. A schematic 2D view of the Anderson's method is given in Fig. B.1.

In [40], a comparison between the Anderson's method and the classical FMM is proposed. Better timings are observed using the Anderson's method than using the classical FMM. In addition, the ease of implementation of the Anderson's method compared to the classical FMM makes it attractive.

Even if the Anderson's method is not kernel independent and requires the analytic form of the Green's function of the underlying kernel, which may not be available for arbitrary kernel in practice, it expresses a representation of the potential that differs from the classical FMM and has been used as the foundation of a few kernel-independent methods.

B.1.2 Kernel-Independent FMM

Anderson's ideas were used in [218,219] and extended to arbitrary kernels corresponding to second-order constant coefficient non-oscillatory elliptic partial differential equations. Their method relies on expressing local Dirichlet problems in each cell. To be more precise, considering any leaf cell c, a surface $E(c)$ encompassing c is chosen. The potentials induced by the source charges and the points of $E(c)$ satisfying a PDE with a unique solution, these potentials are equal in c if they coincide in any surface $C(c)$ enclosing $E(c)$. Some constraints coming from potential theory have to be imposed on $C(c)$ and $E(c)$ and we refer to [218] for their description. Such $E(c)$ are called *equivalent surfaces* and such $C(c)$ are called *check surfaces* (see Fig. B.2). The corresponding equation is:

$$\int_{E(c)} G(\mathbf{x},\mathbf{y})\phi_c(\mathbf{y})d\mathbf{y} = \sum_{\mathbf{y}\in Y\cap c} G(\mathbf{x},\mathbf{y})q(\mathbf{y}), \qquad \forall \mathbf{x}\in C(c)$$

where G refers to a Green kernel, ϕ_c is unknown and numerically computed by inverting this ill-posed integral equation with a regularization (typically a Tikhonov regularization). The sampling

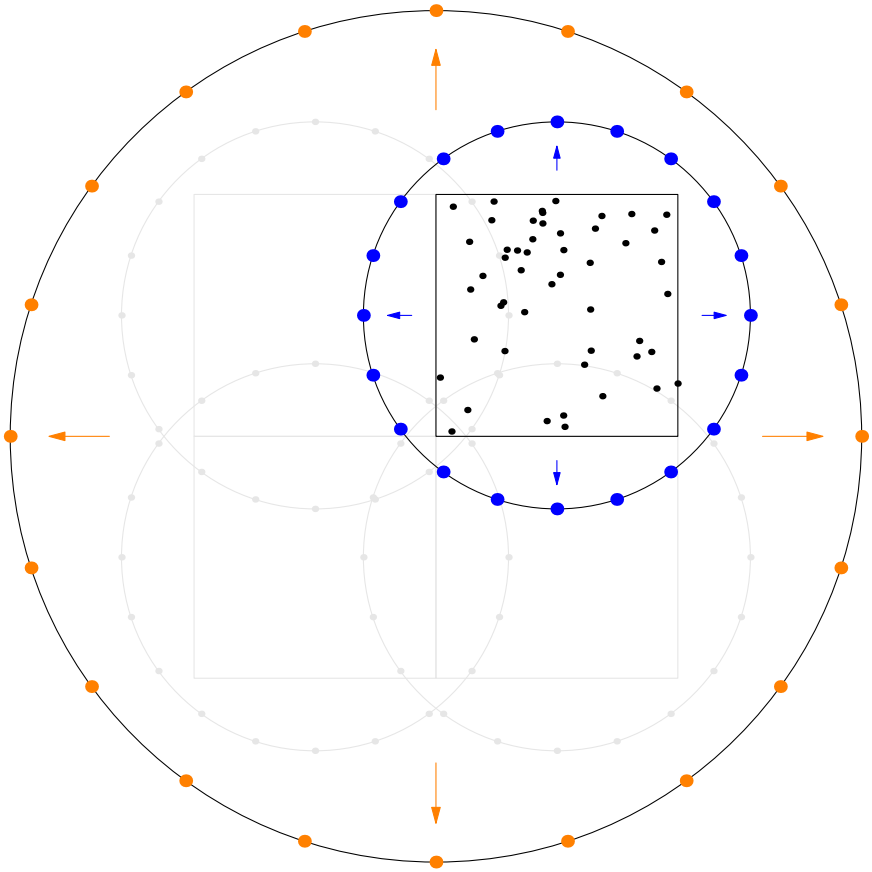


Figure B.1: Schematic view of the multipole aggregations using Anderson's method. The particle's charges (black dots) are switched to cubature nodes in each leaf cell (blue dots). The information on cubature nodes is then transmitted to the cubature nodes of the father's cell (orange dots).

values of ϕ_c on cubature points over $E(c)$ take the role of the terms of a multipole/local expansion and are called *densities*. The way the densities are translated is obtained, once again, by solving equations of the form

$$\int_{E(s)} G(\mathbf{x},\mathbf{y})\phi_s(\mathbf{y})d\mathbf{y} = \int_{E(t)} G(\mathbf{x},\mathbf{y})\phi_t(\mathbf{y})d\mathbf{y}, \qquad \forall \mathbf{x} \in C(t)$$

where s and t are two cells. The exact conditions on equivalent surfaces and check surfaces depend on the type of translation (**M2M**, **M2L**, **L2L**). These translations are depicted in Fig. B.3. By analogy with the classical FMM, the number of term in each expansion (multipole or local) is equal to the number of points in the discretization of the equivalent surface of the cell this expansion corresponds to.

Only kernel evaluations are required, but no explicit analytic expression of it is needed. As a consequence, the method is kernel independent, in the sense that the same code and methodology can be used for a large family of kernels. This method is thus often referred to as $kifmm$ for Kernel-Independent-FMM.

In practice, equivalent and check surfaces are chosen so that they are spheres or boundaries of a d-cube for fast evaluation motivations.

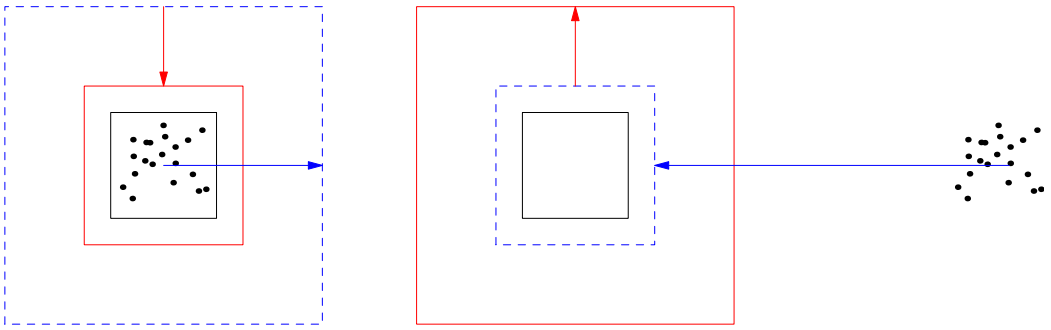


Figure B.2: Equivalent (red) and check (dashed blue) surfaces associated to a cell (black square), depending on the traversal of the tree (upward pass on the left and downward on the right). The link between particles and the equivalent surface is the check surface (links are represented using arrows).

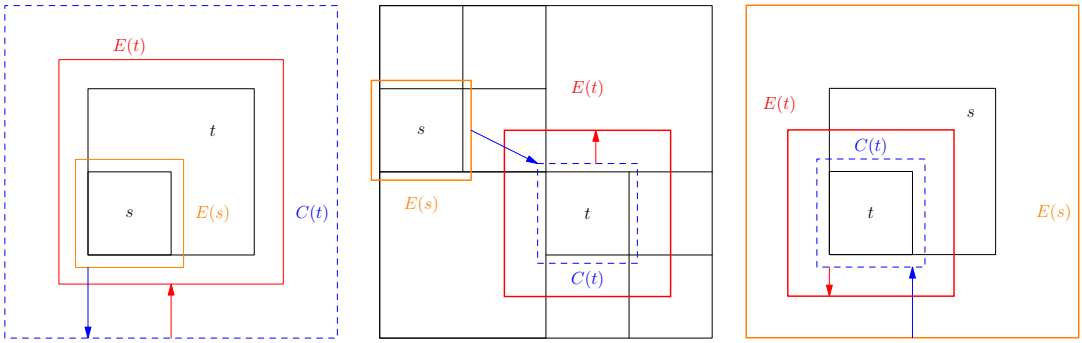


Figure B.3: Translations in *kifmm*. **M2M** (left), **M2L** between separated cells (middle) and **L2L** (right).

B.2 Diagonal kernel-independent FMM

The previously described kernel-independent methods need acceleration techniques for fast **M2L** applications to practically obtain efficient algorithms. By minimizing the number of operations performed in a M2L without modification, one can hope a significant computational gain. We present in this section two methods that provide efficient **M2L** operators in a *diagonal form*, meaning that the matrix representations of these operators are diagonal.

B.2.1 Fourier-based FMM

In [224], a kernel-independent FMM based on Fourier series is introduced. The idea is to approximate a translationally invariant kernel by such series on well separated cells:

$$\begin{aligned} G(\mathbf{x},\mathbf{y}) &= G(\mathbf{x}-\mathbf{y}) \\ &\approx \sum_{\mathbf{j}\in\mathbb{Z}^d \,|\, \mathbf{j}|\leq p} \gamma_{\mathbf{j}} e^{i\alpha\langle \mathbf{j},\mathbf{x}-\mathbf{y}\rangle} \end{aligned} \tag{B.1}$$

where $\alpha \in \mathbb{R}$ is the fundamental frequency number (see [224]) and $\gamma_{\mathbf{j}} \in \mathbb{C}$.

Because the difference between point locations can be splitted in the exponential in the approx-

imation provided in Eq. B.1, this directly lead to a fast algorithm:

$$\begin{aligned}\sum_{\mathbf{y}\in s\cap Y} G(\mathbf{x},\mathbf{y})q(\mathbf{y}) &\approx \sum_{\mathbf{y}\in s\cap Y}\left(\sum_{\mathbf{j}\in\mathbb{Z}^d\,|\,\mathbf{j}|\leq p}\gamma_{\mathbf{j}}e^{i\alpha\langle\mathbf{j},\mathbf{x}-\mathbf{y}\rangle}\right)q(\mathbf{y})\\ &= \sum_{\mathbf{j}\in\mathbb{Z}^d\,|\,\mathbf{j}|\leq p}\gamma_{\mathbf{j}}e^{i\alpha\langle\mathbf{j},\mathbf{x}\rangle}\sum_{\mathbf{y}\in s\cap Y}e^{-i\alpha\langle\mathbf{j},\mathbf{y}\rangle}q(\mathbf{y})\end{aligned}$$

where $\mathbf{x}\in t$, t and s are well-separated cells and p is chosen so that the user's prescribed accuracy is reached. Under this form, the transformation of a multipole expansion into a local one is clearly diagonal.

The way of finding this kind of kernel approximation is based on an adaptive sampling algorithm using a sequence of resolutions of least square problems.

In this FMM formulation, the terms of the multipole/local expansions correspond to the multi-indices $\mathbf{j}\in\mathbb{Z}^d$ with $|\mathbf{j}|\leq p$.

B.2.2 Cauchy FMM

In [57,164], a kernel-independent FMM based on the Cauchy's integral representation formula, on Laplace integrals and on accurate quadrature rules is presented. The aim of this method is to provide a kernel-independent formulation using diagonal **M2L** matrix forms. We first recall the following classical result in complex analysis.

Theorem B.2.1. *(Cauchy's integral formula, [9] Chapter 2, Th. 6) Suppose that f is analytic in an open disk Δ and let γ be a closed curved in Δ. For any $a\in\gamma$*

$$n(\gamma,a)f(a)=\frac{1}{2i\pi}\int_\gamma\frac{f(z)}{z-a}dz,$$

where $n(\gamma,a)$ is the index of a with respect to γ.

We present the main ideas of [164] in the one-dimensional case. Let us consider G an asymptotically smooth non-oscillatory translationally invariant kernel. The first step consists in writing G on well separated sets by means of Cauchy's integral formula:

$$G(x-y)=\frac{1}{2i\pi}\int_\gamma\frac{G(z)}{z-(x-y)}dz \tag{B.2}$$

with a path γ such that the index is equal to 1.

Then, the denominator in the integrand may be expressed as an integral using the Laplace transform:

$$\begin{aligned}\frac{1}{z-(x-y)} &= \frac{e^{i\theta}}{e^{i\theta}(z-(x-y))}\\ &= e^{i\theta}\int_0^\infty e^{-se^{-i\theta}(z-(x-y))}\end{aligned}$$

where $e^{i\theta}$ is used to rotate $z-(x-y)$ in the complex plane such that the real part of this quantity is positive. We then obtain, decomposing γ into a finite (small) set of paths γ_k:

$$\begin{aligned}G(x-y) &= \frac{1}{2i\pi}\sum_k\int_{\gamma_k}e^{i\theta_k}G(z)\int_0^\infty e^{-se^{-i\theta_k}(z-(x-y))}dz\\ &= \frac{1}{2i\pi}\sum_k\int_{\gamma_k}e^{i\theta_k}G(z)\int_0^\infty e^{-se^{-i\theta_k}x}e^{-se^{-i\theta_k}z}e^{-se^{-i\theta_k}-y}dz\\ &= \frac{1}{2i\pi}\sum_k e^{i\theta_k}\int_0^\infty e^{-se^{-i\theta_k}x}\left(\int_{\gamma_k}e^{-se^{-i\theta_k}z}G(z)dz\right)e^{se^{-i\theta_k}y}dz\end{aligned} \tag{B.3}$$

with $\gamma = \bigcup_k \gamma_k$, $\gamma_k \cap \gamma_j = \emptyset \forall k \neq j$ and θ_k such that the positivity of the real part of $e^{i\theta}(z-(x-y))$ is verified. The quadrature of Eq. B.3 defines a single level approximation scheme. The extension of this scheme to a multilevel method is done by centering the expansions on the center of the corresponding cells, and by switching these centers applying diagonal products.

Appendix C

The Brandt's method

In [47], A. Brandt considers that an oscillatory asymptotically smooth kernel can be splitted into an asymptotically smooth part and an oscillatory part:

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

This factorization allows to derive a fast summation scheme.

C.1 One dimensional case

Let $X, Y \subset \mathbb{R}$ be the target and source point clouds respectively. For any $x \in X$, the summation problem expressed in Eq. 2.1 can be splitted into

$$p(x) = \left(\sum_{y\in Y} G_+(x,y)q(y)\right) + \left(\sum_{y\in Y} G_-(x,y)q(y)\right)$$

where

$$\begin{cases} G_+(x,y) = G(x,y)\delta_{x\leq y} \\ G_-(x,y) = G(x,y)\delta_{x>y} \end{cases}$$

and $\delta_{\mathcal{B}} = 1$ if $\mathcal{B}$ is true and 0 otherwise. This splitting is illustrated on Fig. C.1.

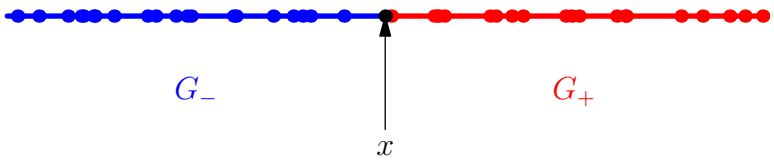


Figure C.1: Decomposition of the source point cloud and according kernel splitting with regard to a single target particle (in black).

Thus, considering any of those two new sums, for instance the first one, we have:

$$\begin{aligned} \sum_{y\in Y} G_+(x,y)q(y) &= \sum_{y\in Y} G_+(x,y)e^{-\kappa(x-y)}e^{\kappa(x-y)}q(y) \\ &= e^{-\kappa x}\sum_{y\in Y} \underbrace{\left(G_+(x,y)e^{\kappa(x-y)}\right)}_{\text{Does not oscillate}} (e^{\kappa y}q(y)) \end{aligned}$$

where the new red kernel is clearly asymptotically smooth. In addition, $e^{-\kappa x}$ and $e^{\kappa y}$ are respectively applied on the result of the N-body problem to each target particle and on the entries of this problem for each source particle in order to recover the final result. Because this process has to be done twice (once for G_+ and once for G_-), but involving different target particles for each source particle, the complexity of this method can be bounded by twice the cost of the N-body problem applied on asymptotically smooth kernels.

The signs in the complex exponential is changed in G_-, compared to G_+, to recover an asymptotically smooth kernel:

$$\sum_{y \in Y} G_-(x,y)q(y) = e^{\kappa x} \sum_{y \in Y} \left(G_-(x,y) e^{-\kappa(x-y)} \right) \left(e^{-\kappa y} q(y) \right)$$

so that the two directions $x - y > 0$ and $x - y < 0$ are treated differently, using different *directional* approximations (meaning that the approximations depend on the direction $+1$ or -1) by a given approximation method for asymptotically smooth kernels. These directional approximations correspond to the result of any hierarchical method to the modified kernel (in red on equations) by using pre- and post-multiplications by complex exponentials. This *directional* idea is the key of the method proposed by A. Brandt.

C.2 More than one dimensional case

To extend the algorithm to the multivariate case, one first need to adapt the definition of a direction in more than one dimension.

Definition C.2.1. *Let $\mathbb{S}^{d-1} \subset \mathbb{R}^d$ be the unit sphere in $\mathbb{R}^d$. Any point $\lambda \in \mathbb{S}^{d-1}$ is named a direction.*

Clearly, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $\frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|} \in \mathbb{S}^{d-1}$, so there exists a direction $e \in \mathbb{S}^{d-1}$ (actually equal to $\frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|}$) such that

$$\begin{aligned} e^{i\kappa|\mathbf{x}-\mathbf{y}|} e^{-i\kappa\langle \mathbf{x}-\mathbf{y}, e\rangle} &= e^{i\kappa\langle \mathbf{x}-\mathbf{y}, \frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|}\rangle} e^{-i\kappa\langle \mathbf{x}-\mathbf{y}, e\rangle} \\ &= e^{i\kappa\langle \mathbf{x}-\mathbf{y}, \frac{\mathbf{x}-\mathbf{y}}{|\mathbf{x}-\mathbf{y}|} - e\rangle} \\ &= 1 \end{aligned}$$

which does not oscillate.

The problem is that if e explicitly depends on both $\mathbf{x}$ and $\mathbf{y}$, we can not except a computational gain, since all directional approximations will correspond to a single pair of target and source particles. A. Brandt proposes to consider only a subset of the set of directions, $E = \{e_1, ..., e_P \in \mathbb{S}^{d-1}\}$, $P \in \mathbb{N}^*$ such that the distance between any two "neighbors" directions of E is bounded by a parameter δ and such that, using interpolation on E, a directional approximation corresponding to any direction can be efficiently approximated from the directional approximations corresponding to directions in E. For any function ϕ defined on E, this may be written as

$$\phi(e) \approx \sum_{l \in [\![1,P]\!]} w_l^E(e) \phi(e_l)$$

where w_l^E denotes the interpolation function associated to $e_l \in E$.

The previous one-dimensional case corresponds to the case $P = 2$, with $\mathbb{S}^0 = \{-1, 1\}$, so that no interpolation is needed by directly considering these two directions.

Appendix D

HPC for Fast Multipole Method

This appendix is dedicated to various methods that are exploited for implementations of the FMM on HPC architecture. In Sect. D.1, we present fundamental notions on general HPC architectures. In Sect. D.2, we present the *vector stacking* idea and in Sect. D.3, we describe how the cells may be ordered in practical FMM implementations.

D.1 HPC architectures

Here we recall the bases on the HPC architectures and their programming. This high level overview of HPC architectures is divided in two sections: we first present in Section D.1.1 considerations about the lowest level of the architecture, i.e. the Central Processing Unit (CPU or *processor*); then we introduce in Section D.1.2 the multiple parallelism levels. As main reference for the HPC architectures, we refer to [136].

D.1.1 Central Processing Unit

The frequency of a CPU is not a sufficient notion in a HPC context to measure the efficiency of a program on a particular processor. Since the operations performed on data already loaded in the registers can be performed much faster than memory accesses, a predominance of load/store instructions prevents the theoretical peak performance of the processor to be reached. In addition, because of the increasing size of the registers, the *vectorization* also has to be taken into account for efficient program implementations on general modern architectures. We here present a simple model used to measure the efficiency of an implementation in terms of memory accesses compared to floating point operations.

D.1.1.1 Memory gap and arithmetic intensity

In a computer, the different memories are organized hierarchically in the way depicted on Fig. D.1. The closer to the CPU the memory, the faster the memory accesses are. The best performance is reached by minimizing the main memory access and maximizing the use of data already stored in the two levels of cache memories.

A useful tool to quantify the efficiency of an algorithm in term of cache optimization is the *roofline model* [213]. This model introduces the *arithmetic intensity* I of a routine, defined by

$$I = \frac{W}{Q}$$

where W expresses the amount of *work* operated by this routine in terms of floating point operations and Q refers to the *traffic*, counting the number of bytes of memory transfers incurred during the

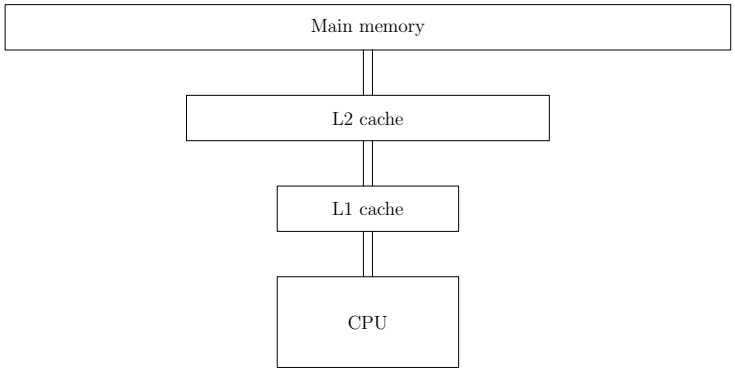


Figure D.1: Schematic view of the memory hierarchy.

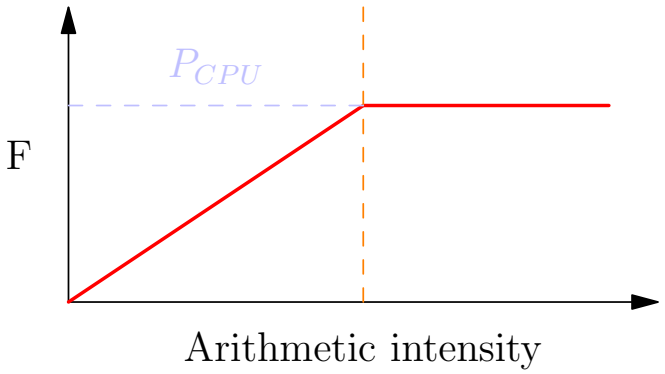


Figure D.2: Schematic representation of the roofline model. Before the dashed orange line, the arithmetic is not sufficient enough to reach the CPU performance peak and the routine is memory-bound. Beyond this line, the routine is compute-bound.

execution. This traffic strongly depends on the architecture (namely on the cache memory). The greater the arithmetic intensity, the faster the implementation, up to a certain limit dictated by the cache memory size and the CPU frequency. The objective is thus to design algorithms able to maximize this arithmetic intensity. Hence, the roofline model expresses the attainable number of floating point operations per second F as

$$F = min\{P_{CPU},\ P_{Mem} \times I\} \tag{D.1}$$

where P_{CPU} denotes the theoretical peak floating point operation performance of the CPU and P_{Mem} denotes the peak memory bandwidth. If the minimum in Eq. D.1 is realized by P_{CPU}, the routine is said to be *compute-bound*: the practical limitation for a better efficiency is only linked to the CPU performance. In the other case, if the minimum in Eq. D.1 is realized by $P_{Mem} \times I$, then the routine is said to be *memory-bound*: the arithmetic intensity is not sufficient to fully benefit from the CPU performance. See Fig. D.2 for a representation of the model.

D.1.1.2 Fused Multiple Add instruction

An important feature in modern processors is the Fused Multiple Add (FMA) instruction [6]. The idea is that the combination of the two instructions $a = b \times c$ and $d = d + a$ allows to perform both operations at the cost of one. Potential performance gains can then be expected by favoring, for instance in the C syntax, the instructions of the form "$a = b * c + d$;" "$a = b * c + d$".

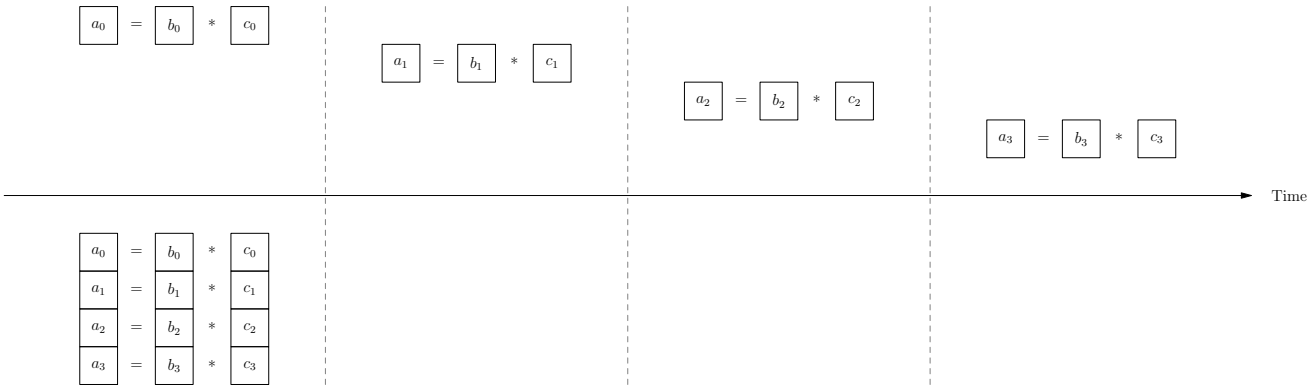


Figure D.3: Schematic view of the difference between SIMD vector "multiply" ("*") operation and scalar "multiply" operations on the vector data. Notice that the frequency can be slightly affected (decreasing of a small factor) by the use of vector operations, so this representation is not fully exact.

D.1.2 Multiple parallelism levels

Modern HPC architectures include multiple parallelism levels in their conception. Three different levels can be distinguished [136]:

- The *Instruction-Level Parallelism* (ILP);
- The *Data-Level Parallelism* (DLP);
- The *Thread-Level Parallelism* (TLP).

D.1.2.1 Instruction-Level Parallelism

The ILP refers to the parallel execution of a sequence of instructions belonging to a specific thread of execution of a process. Practically, such parallelism can be obtained, for instance, through *instruction pipelining* that consists in decomposing the incoming instruction flow (i.e. a sequence of instructions performed on a data set) into a series of sequential steps involving different processor units. Hence, independent steps involving different processor units can be performed in parallel. From the developer viewpoint, pipelining effects can be obtained with minimal effort by the use of *loop unrolling*, consisting in repeating a certain amount of loop iterations[1]. Modern compiler optimizations use the unroll technique to optimize the codes.

D.1.2.2 «SIMD Data»-Level Parallelism

The execution of the same instruction on different data can be realized simultaneously on computers with particular multiple processing elements (which is a form of DLP parallelism). This is referred to as SIMD parallelism for Single Instruction Multiple Data. A code that includes SIMD instructions is said to be vectorized. Basically, data are loaded into vector registers (SIMD registers) whose length depends on the processor type[2] and the operations are performed on these vectors directly, all the entry at the same time, instead of a sequence of scalar operations performed on all these entries (see Fig. D.3). The main requirement for vectorization is the data alignment, even if *gather instructions* are able to fill the SIMD registers. Bytes that are aligned in memory according to the SIMD instruction set can be loaded and treated directly.

There exist several approaches to produce SIMD codes:

[1] This can also be achieved using precompiler directives, such as # **pragma unroll** with *gcc*.
[2] Up to 512 bits on processors supporting AVX-512.

- Explicitly write SIMD instructions using processor specific functions called intrinsics (Advanced Vector Extensions AVX, Streaming SIMD Extensions SSE, ...);
- Add precompiler directives (e.g. # **pragma omp simd** with OpenMP using the C++ language)
- Call routines that are implemented using SIMD instructions (e.g. BLAS calls, see App. D.2);
- Rely on compiler auto-vectorization (see for instance [142]).

The potential SIMD performance gain depends on the architecture but may be important enough to justify the modification of a naive algorithmic: up to 8x speedup for double precision and 16x speedup for single precision using AVX-512.

D.1.2.3 Thread-Level Parallelism

The last level of parallelism (TLP) corresponds to the simultaneous running of independent instruction flows (*threads*). There exist multiple ways of exploiting the TLP. Different threads can be assigned to different cores and their instructions performed in parallel, or a single core can simultaneously execute multiple threads (*SMT*). The SMT idea consists in filling the CPU pipelines with instructions from different threads. TLP can be obtained, from a programming viewpoint, by using OpenMP [5] compiler directives. Another general paradigm in TLP is the *task parallelism*, first introduced in Cilk and further improved in advanced task runtimes (see [200]). OpenMP also supports task parallelism. One of the bottlenecks in shared-memory parallelism comes from the synchronizations required by the concurrent accesses many threads can have to the same data.

In a *distributed memory* framework, the distant compute nodes communicate with *messages*. In terms of programming, the Message Passing Interface (MPI) standard [105] is widely used to write a single program executed on different processes and whose behavior varies depending on the process identifier.

D.2 BLAS routines and vector stacking for Fast Multipole Methods

The Basic Linear Algebra Subprograms (BLAS) [84, 86, 87] are highly efficient implementations of linear algebra operations. Depending on the operation type, a level is associated to these operations, divided in three different levels:

- Level 1: vector-vector operations (such as scalar product, scaling, addition, ...)
- Level 2: matrix-vector operations (such as matrix-vector product, ...)
- Level 3: matrix-matrix operations (such as matrix-matrix products, ...)

In practice, Level 3 BLAS routines are more efficient than multiple applications of Level 2 BLAS routines. To be more precise, when a set of matrix-vector products can be formulated as a matrix-matrix product (with the same number of mathematical operations), the matrix-matrix BLAS product will be far more efficient than the multiple matrix-vector BLAS products.

The idea of concatenating vectors in matrices to use BLAS 3 routines has been applied in the FMM in different ways. We refer to this technique as *vector stacking*. At the most abstract level, the optimizations we describe in the following all rely on the fact that similar patterns are applied many times in the FMM based on 2^d-trees, involving each time a product with a same matrix on different entries [72, 139, 166, 173].

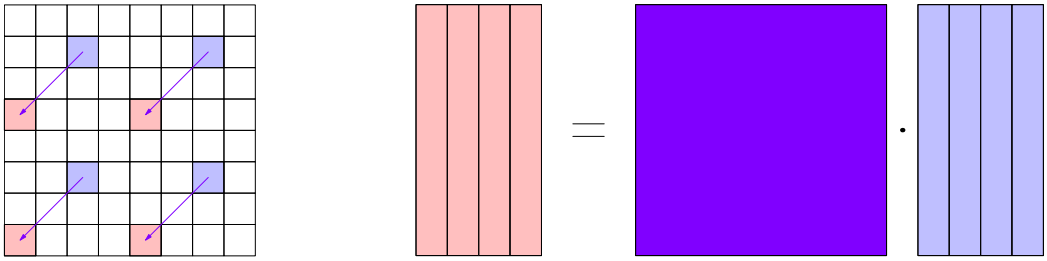


Figure D.4: Similar **M2L** interactions: all the blue cells interact with the red one according to the purple arrows (left). Matrix-matrix interactions: concatenation of multipole expansions (blue), **M2L** matrix (purple), resulting local expansions (red) after the matrix-matrix product (right).

D.2.1 Similar M2L interactions

When the target and source spaces are represented using 2^d-trees with the same root, any target cell has a position strictly equal to a possible source cell and reciprocally. Because of the regularity of these trees, if the expansions are given relatively to a *fixed position in the cells*[3], similarities in the **M2L** operators may appear. To be more precise, in many FMM formulations, the **M2L** operators between cells with the same relative positions are equal. An example of similar **M2L** interactions is given in Fig. D.4.

In [72] is introduced a matrix form for the **M2L** of the 3D Laplace FMM and in [73] a way of stacking vectors sharing a same **M2L** operator using an adaptive FMM. Focusing on the cells of Fig. D.4, the high level idea of the associated optimization is to regroup these interactions together. Because the **M2L** matrix does not change, the multipole expansions (seen as vectors) can be concatenated into a matrix such that the task of computing all the interactions depicted in Fig. D.4 corresponds to the computation of a matrix-matrix product. Hence, the use of BLAS 3 operations reduces the application time. We may notice that this method can be affected by the cost of the copies of the expansions in the case of non-uniform distributions and performs better in the case of uniform distributions.

D.2.2 Extension to other operators

The vector stacking idea can be used for the application of other FMM operators than the **M2L** one. In [166], BLAS 3 are also applied to the **P2P** operator obtained by interpolating the charges and the potentials considered as continuous functions. The evaluation of the W- and X-lists (see Sect. 2.2.4.2) are also accelerated using vector stacking in [166].

D.2.3 Multiple right-hand-sides

The BLAS 3 operations can also be exploited when the matrix-vector product accelerated using the FMM is applied to multiple right-hand-sides. This can be written as a matrix-matrix product instead of multiple matrix-vector products. Hence, the FMM linear operators can be applied to matrices instead of vector, which may transform matrix-vector products into matrix-matrix product. This is done for instance in the case of interpolation-based FMM in [210].

D.2.4 Arithmetic intensity

In the FMM algorithm, the same data[4] are used many times in different computations. For instance, a same multipole expansion can be used for **M2L** operations with 2 spatially close target

[3] i.e. centered in the center of these cells for the 3D Laplace FMM, using cubature on the sphere centered on this center for *HF-FMM* or using the same interpolation nodes relatively to this center for the interpolation based FMM.

[4] e.g. multipole/local expansions or, if precomputed, the **M2M**/**M2L**/**L2L** matrices

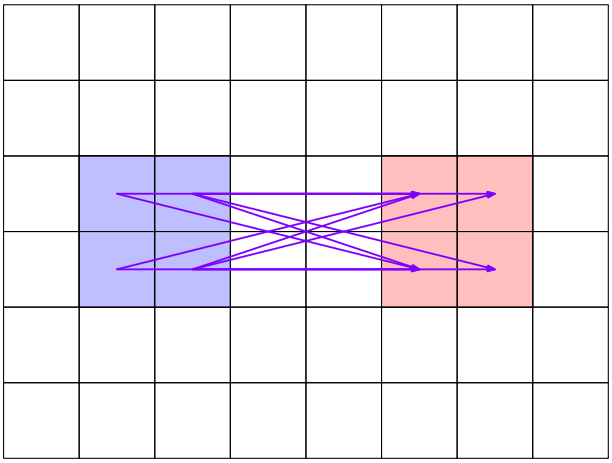


Figure D.5: All interactions (purple) between a source sibling group (blue) and a target one (red)

cells. Since these different **M2L** interactions have no reason to be computed consecutively, this multipole expansion may be loaded multiple times in the cache memory. Hence, by reordering the computations in order to perform these different **M2L** evaluations consecutively, one may increase the arithmetic intensity.

In [166], associated to the FMM library *pvfmm* [167], is introduced the concept of *sibling groups*, referring to the set of sons of a given a node. By treating together the **M2L** interactions of each member of a given sibling group with the members of another one (replacing the non-existing interactions with multiplications by zeros), one may obtain an increasing arithmetic intensity. Indeed, when the target and source expansions corresponding to two interacting siblings are loaded in the cache memory (assuming that this memory is sufficiently large), groups of interactions involving the same source and target cells (see Fig. D.5) can be realized alltogether without extra memory accesses. A similar idea is describe in [173]. These optimizations are rather suited for uniform distributions.

The method presented in [166] adds another level of optimization. Since the **M2L** matrices are diagonal in the associated FMM formulation, the entries of the grouped source/target cells are interleaved in order to recover sequences of small matrix-matrix products instead of Hadamard products during the **M2L** applications, benefiting from a higher arithmetic intensity.

D.3 Indexing cells

Suppose that all the cells of a perfect 2^d-tree are stored in a cell array. There exists efficient ways for indexing the cells in this array. This allows, for instance, to retrieve a given cell without tree traversal. These techniques usually rely on space filling curves.

D.3.1 Space Filling Curves

In the case of a perfect 2^d-tree T, this problem of efficiently storing the cells consists in indexing these cells at level l of T using indices in $[\![0, \left(2^l - 1\right)^d]\!]$. Because these cells realize the partition of a d-cube (an interval in 1D, a square in 2D and a cube in 3D), each of them can be trivially indexed using a multi-dimensional index in $[\![0, 2^l - 1]\!]^d$ computed from their center and the number of cell along each axis[5]. An array of size 2^{ld} storing the cells on the l^{th} level may then be declared. Hence,

[5] the set of centers of cells of a given level being a regular grid in dimension d

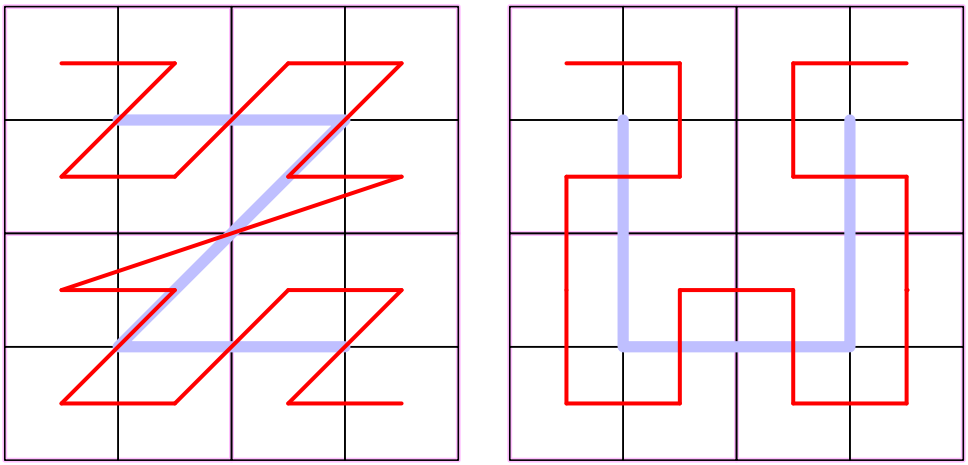


Figure D.6: 4-tree representation of the Morton on the left (resp. Hilbert on the right) ordering at level 2 (red curve and black cells) and 1 (pale blue curve and purple cells)

the problem is to find an easily invertible application h such that the range of h verifies

$$h([\![0, 2^l - 1]\!]^d) = [\![0, \left(2^l - 1\right)^d]\!]$$

that associates to any cell with spatial multi-index in $[\![0, 2^l - 1]\!]^d$ its position in the cell's array. In addition, we are concerned with

- a *data locality*: the spatial proximity of two cells has to be passed on in the cell ordering in the sense that their indices have to be close;
- a *hierarchical property*: we want to be able to find the index of the father of a cell and the indices of its sons.

The first example of such an ordering we present is the Morton ordering [183, 211] (sometimes called Z-curve). The idea is to interleave the terms of the dyadic decomposition of the elements of the multi-index $I \in [\![0, 2^l - 1]\!]^d$ to create an index $i \in [\![0, \left(2^l - 1\right)^d]\!]$ called the *Morton Index*. This process is illustrated on Eq. D.2, where h denotes the Morton mapping from $[\![0, 2^l - 1]\!]^d$ to $[\![0, \left(2^l - 1\right)^d]\!]$. On Fig. D.6, the way the cells of a given level are traversed according to the induced ordering in $[\![0, \left(2^l - 1\right)^d]\!]$ is depicted. Suppose that we want to find the Morton index of a cell located on the level 4, its multi-index being equal to $(4, 5, 2)$. On level 4 there are at most 2^4 cells along each axis, so 4 bits are sufficient to represent each term of this multi-index, which can be expressed as $(0100, 0101, 0010)$. The Morton index is computed as

$$h(0100, 0101, 0010) = 000110001010 \tag{D.2}$$

The computation of a Morton index from any multi-index and the reciprocal of this operation are easy to implement using bit operations, leading to highly efficient conversions which make this ordering particularly popular. The second widely used ordering in the hierarchical method is the Hilbert ordering, also represented on Fig. D.6. Compared to the Morton one, it is costlier to compute.

D.3.2 Hashed OcTree

When the distribution is not uniform, there may exist cells of the 2^d-trees that contain no particle with the $MaxDepth$ criterion (see Sect. 2.2.1.2). One wants to avoid their storage since they can

cause a significant extra memory cost. To use the Morton or Hilbert ordering in such a case, the structures have to be adapted because the analogy between Morton/Hilbert index and index in the cell's array does not handle any more. The *hashed octree* structure [211] is designed to store such 2^d-trees with missing cells, by the use of hash tables and linked lists.

Résumé

Nous nous intéressons dans ce manuscrit aux méthodes hiérarchiques pour l'accélération des résolutions de systèmes linéaires issus de la méthode des éléments finis de frontière pour des problèmes hautements oscillants (tels qu'apparaissant en éléctromagnétisme). Une attention particulière est portée aux méthodes multipolaires rapides (MMR). Nous détaillons une nouvelle approche abstraite des méthodes hiérarchiques, en particulier des différentes formulations MMR, en présentant dans quelle mesure les symétries des structures arborescentes de ces méthodes peuvent être exploitées au sein des différentes MMR. Afin d'étendre le cadre de la formulation MMR explicite pour le noyau de Helmholtz en haute fréquence à ces symétries, nous introduisons les règles de Lebedev dans ce contexte. Cette modification conduit à d'importantes difficultés, compliquant les méthodes utilisées pour obtenir une MMR multi-niveaux. Pour pallier ce problème, nous proposons une approche pour la diagonalisation par bloc de matrices particulières à ce contexte ainsi qu'une stratégie pour l'évaluation rapide des produits par ces matrices. Enfin, nous décrivons la réalisation complète d'une bibliothèque MMR directionnelle *kernel-independent* usant d'interpolation sur des grilles cartésiennes. Ce type d'interpolation autorise l'usage de transformées de Fourier rapides dans le traitement des interactions approchées par la MMR. Les aspects théoriques sont abordés (consistance de la méthode d'approximation, application des symétries de la MMR après application des transformées de Fourier, ...) ainsi que les aspects algorithmiques et ceux liés au calcul haute performance sur un cœur de calcul. Des résultats numériques et des comparaisons avec une bibliothèque MMR directionnelle de référence illustrent les performances de notre implémentation.

Mots clés — méthodes multipolaires rapides, symétries, transformées de Fourier rapides, calcul haute performance sur un cœur, noyaux hautement oscillants, distributions non-uniformes de particules

Abstract

We are interested in this manuscript in hierarchical methods for accelerating the resolution of linear systems derived from the boundary element method for problems appearing in electromagnetism. Particular emphasis is placed on the Fast Multipole Methods (FMMs). We detail a new abstract approach for the hierarchical methods, and particularly for different FMM formulations, presenting in what extent the symmetries of the tree structures of these methods can be exploited within different FMM formulations. We extend the explicit FMM formulation for the Helmholtz kernel in the high frequency regime to these symmetries, introducing the Lebedev's rules in this context. This modification leads to important difficulties when trying to obtain a multi-level FMM. To overcome this problem, we propose an approach for the explicit block-diagonalization of specific matrices as well as a strategy for the rapid evaluation of the products by these matrices. Finally, we describe the complete realization of a *kernel-independent* directional FMM library using interpolation on cartesian grids. Such interpolation allows the use of Fast Fourier Transforms in the processing of the interactions approximated by the FMM. The theoretical aspects are discussed (such as the consistency of the approximation method or the application of the symmetries in this FMM formulation) as well as aspects related to algorithmics and to high-performance computing on one CPU core. Results and comparisons with a *state-of-the-art* directional library illustrate the performance of our implementation.

Keywords — fast multipole methods, symmetries, fast fourier transforms, high performance computing on one CPU core, highly oscillatory kernels, non-uniform particle distributions