The “Interpolated Factored Green Function” Method

Thesis by
Christoph Bauinger

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2023
Defended August 15, 2022

ACKNOWLEDGEMENTS

When I began my studies in Austria nearly a decade ago, I would have never guessed that my academic path would lead me to the United States, to do a PhD at Caltech. Now, at the end of this journey, I am glad that it did. In this past decade, I was able to travel the world, to learn lessons for life, to dedicate myself to science, to invent and develop new things, and to meet amazing people, some of which left a significant impression on my life and without whom this thesis would never have been possible. In what follows, I want to thank those that accompanied and supported me on this journey.

The first person I want to thank is my advisor here at Caltech, Professor Oscar P. Bruno. The present thesis is based upon a strong and enjoyable collaboration between us. Without him, my PhD would not have been possible, and I will be thankful for the rest of my life for the opportunities he offered and the knowledge he imparted to me. I wish him all the best and hope that our friendship will continue after my departure from Caltech. I would also like to thank the members of my committee, Professors Dan Meiron, Houman Owhadi, and Peter Schroeder for their time and effort in providing guidance and insightful comments regarding my thesis work.

I want to thank Edwin Jimenez, who guided the work on incorporating the IFGF method for the solution of scattering problems. This thesis, Section 5.5 in particular, would not exist in the present form without him, and I hope that our paths will cross again in the future.

A PhD is not always pleasant, fun, or easy. When things did not work as intended, my friends were always there to support me and help me push through, sometimes simply by virtue of encountering and sharing similar difficulties, other times through valuable advice and help. Thank you to all of you.

I also want to thank my family for their unwavering support. It was not easy to leave home, especially since I did not know if I would return to Austria and to my family, and I would not have been able to do it without their encouragement.

Finally, I want to thank Anna-Sophia Bauer, who came into my life when I least expected it and who is enriching it in a way I had not thought possible. I am looking forward to many more years together.

ABSTRACT

This thesis presents a novel *Interpolated Factored Green Function* (IFGF) method for the accelerated evaluation of the integral operators in scattering theory and other areas. Like existing acceleration methods in these fields, the IFGF algorithm evaluates the action of Green function-based integral operators at a cost of $O(N \log N)$ operations for an N-point surface mesh. The IFGF strategy capitalizes on slow variations inherent in a certain Green function *analytic factor*, which is analytic up to and including infinity, and which therefore allows for accelerated evaluation of fields produced by groups of sources on the basis of a recursive application of classical interpolation methods. Unlike other approaches, the IFGF method does not utilize the Fast Fourier Transform (FFT), and it is thus better suited than other methods for efficient parallelization in distributed-memory computer systems. In fact, a (hybrid MPI-OpenMP) parallel implementation of the IFGF algorithm is proposed in this thesis which results in highly efficient data communication, and which exhibits in practice excellent parallel scaling up to large numbers of cores—without any hard limitations on the number of cores concurrently employed with high efficiency. Moreover, on any given number of cores, the proposed parallel approach preserves the linearithmic ($O(N \log N)$) computing cost inherent in the sequential version of the IFGF algorithm. This thesis additionally introduces a complete acoustic scattering solver that incorporates the IFGF method in conjunction with a suitable singular integration scheme. A variety of numerical results presented in this thesis illustrate the character of the proposed parallel IFGF-accelerated acoustic solver. These results include applications to several highly relevant engineering problems, e.g., problems concerning acoustic scattering by structures such as a submarine and an aircraft-nacelle geometry, thus establishing the suitability of the IFGF method in the context of real-world engineering problems. The theoretical properties of the IFGF method, finally, are demonstrated by means of a variety of numerical experiments which display the method's serial and parallel linearithmic scaling as well as its excellent weak and strong parallel scaling—for problems of up to $4,096$ wavelengths in acoustic size, and scaling tests spanning from 1 compute core to all $1,680$ cores available in the High Performance Computing cluster used.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Christoph Bauinger and Oscar P. Bruno. "Interpolated Factored Green Function" method for accelerated solution of scattering problems. In: *Journal of Computational Physics* 430 (Jan. 2021). DOI: 10.1016/j.jcp.2020.110095.
C.B. participated in the conception of the project and the formulation and proofs of the theorems, he produced an implementation of the method in the C++ programming language, generated the data and figures, and participated in the writing of the manuscript. Chapter 3 and the serial tests presented in Chapter 5 are based on this contribution.

[2] Christoph Bauinger and Oscar P. Bruno. Massively parallelized Interpolated Factored Green Function method. In: *arXiv:2112.15198* (2021).
C.B. participated in the conception of the project, implemented the method, generated the data and figures, and participated in the writing of the manuscript. Chapter 4 and the parallel tests presented in Chapter 5 are based on this contribution.

[3] Edwin Jimenez, Christoph Bauinger, and Oscar P. Bruno. IFGF-accelerated integral equation solvers for acoustic scattering. In: *arXiv:2112.06316* (2021).
C.B. participated in the conception of the project, the implementation of the solver, the generation of data and figures and the writing of the manuscript. Section 5.5 is based on this contribution.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

This thesis presents a novel *Interpolated Factored Green Function* (IFGF) method for the accelerated evaluation of discrete integral operators that arise as discrete versions of Green function-based boundary integral formulations of boundary value problems (BVP) for certain types of partial differential equations (PDEs). More precisely, in what follows, we focus on the particularly challenging high-frequency boundary integral equations (BIE) associated with scattering of acoustic or electromagnetic waves by three-dimensional obstacles. These problems are highly relevant in areas such as communications, stealth, remote sensing, radar, sonar, imaging, photonics, electronics, noise management and many other important areas of civilian and military interest in electrical engineering, applied physics, and, indeed, science and engineering in general [1–10]. A brief overview of the mathematical and computational methods associated with the field of integral equations is provided in Section 1.1; a more thorough description can be found in [11–20].

For the types of PDE problems considered in this thesis, namely linear PDE problems for which an explicit Green function [21] exists, integral equation-based solvers provide a number of advantages over direct PDE discretization methods such as the Finite Difference Method (FDM) [22] and the Finite Element Method (FEM) [23]. On one hand, integral methods only require discretization of the scattering surface, i.e., the boundary of the obstacle, instead of the propagation volume—which is particularly beneficial for large volume-to-surface ratios. Secondly, they inherently satisfy radiation conditions at infinity, and are thus especially well-suited for exterior problems over unbounded propagation domains, whereas the aforementioned PDE discretization methods typically require use of specialized domain truncation methodologies [24]. And, finally, boundary integral equation methods do not suffer from dispersion and pollution effects inherent in the FDM and FEM [24–26] which, resulting from accumulation of truncation errors over propagation domains, require use of fine discretizations for accuracy and thus give rise to high computational costs in terms of both memory and computing time.

Boundary integral equation methods do give rise to a certain significant challenge, however, which is tackled by the IFGF method presented in this thesis—namely the

prohibitive computational cost that results from straightforward BIE computational implementation. In detail, since the discretization of BIEs typically results in a densely populated linear system of equations (in contrast to the sparse systems that are obtained in the FDM and FEM contexts), a direct solution algorithm (e.g., Gauss elimination, LU factorization, etc., see [27, Sec. 4], [28, Sec. 1], [29, Sec. 1], or [30, Sec. 2]) requires in general $O(N^3)$ operations, where N denotes the number of surface discretization points. Clearly, this cubic complexity leads to unacceptable computing costs for most of the large problems arising in applications. A first remedy toward alleviating this difficulty can be found in the use of iterative solvers, like GMRES (as described in Section 2.1), which solves the resulting discrete dense system iteratively and thus reduces the cost of inverting a dense matrix to that of repeated evaluation of the discrete integral operator, which requires the significantly smaller $O(N^2)$ cost per evaluation. Still, the $O(N^2)$ computational expense proves prohibitive in most high-frequency applications, which has lead to the search for suitable algorithmic acceleration methods such as the IFGF approach introduced in this thesis. Like previous acceleration methods, such as the Fast Multipole Method (FMM) [14, 31–35] and other approaches [16, 20, 36–42], the IFGF method reduces the $O(N^2)$ cost of the discrete operator evaluation, both in terms of computing time and memory requirements, to $O(N \log N)$. Applying an iterative solver to high-frequency scattering problems considered in this thesis is not straightforward, and, theoretically, the number of iterations required to approximate the solution to any given accuracy ε requires a number of iterations proportional to the number N of surface discretization points. To reduce the number of iterations in such an approach, a suitable pre-conditioning is required. Since the pre-conditioning of the resulting discrete integral equation is not closely related to the novelties presented in this thesis, it is not further pursued. A possible approach can be found in, e.g., [43].

As indicated above, the IFGF method is certainly not the first method to tackle the fast evaluation of discrete integral operators occurring in high-frequency scattering problems. On the contrary: Significant literature has been devoted to the development of fast, stable and simple methods to reduce the algorithmic complexity of this problem and to enable the solution of increasingly large problems [34, 44–51]. Additionally, the emergence of parallel computers and heterogeneous cluster systems in the past decade necessitated the development of numerical methods, which incorporate and utilize the available hardware in a manner that optimizes the "parallel scaling" properties. An overview of previous work in this field and the relevance of the IFGF method is therefore given in Section 1.2.

1.1 Integral equations

As indicated above, the IFGF method tackles the particularly challenging problem of accelerating the evaluation of discrete integral operators of high-frequency scattering problems. For definiteness, in what follows, we focus on the problems associated with the *Helmholtz equation*. Clearly, due to their close relation, similar considerations as presented in this thesis for the Helmholtz equation are applicable—with minimal adjustments—to Maxwell and Laplace equations (see [11–16]). For a concise and self-contained presentation, we introduce the Helmholtz equation in what follows

$$\Delta u(x) + \kappa^2 u(x) = 0 \quad \text{for } x \in \Omega \text{ or } x \in \mathbb{R}^3 \setminus \bar{\Omega}, \tag{1.1}$$

where $\Omega \subset \mathbb{R}^3$ denotes a bounded domain. The cases $x \in \Omega$ and $x \in \mathbb{R}^3 \setminus \bar{\Omega}$ are called the *interior* and *exterior* problem, respectively.

Definition 1 (Wavenumber)**.** *The constant κ in* (1.1) *is called the* wavenumber *and it relates to the* wavelength λ *as* $\kappa = 2\pi/\lambda$. *Further, it relates to the* frequency f *as* $2\pi f = c\kappa$, *where c denotes the* speed of sound/light *of the medium under consideration, and the* angular frequency ω *as* $\omega = 2\pi f$.

The following introduction of integral equations follows the presentation in [13], which focuses on the *Dirichlet* and *Neumann* problems shown in Definitions 2 and 3, respectively, and which bases its analysis of the corresponding integral equations on Riesz' theory for compact operators.

Definition 2 (Dirichlet problem)**.** *Let $\Omega \subset \mathbb{R}^3$ denote a bounded domain with a twice differentiable boundary $\Gamma = \partial\Omega$. Further, let $f \in C(\Gamma, \mathbb{C})$ be a complex-valued and continuous function defined on Γ and let κ denote the wavenumber. The* Dirichlet problem for the Helmholtz equation *is given as*

$$\Delta u(x) + \kappa^2 u(x) = 0 \quad \textit{for } x \in \Omega \textit{ or } x \in \mathbb{R}^3 \setminus \bar{\Omega} \tag{1.2}$$

$$u(x) = f(x) \quad \textit{for } x \in \Gamma. \tag{1.3}$$

Definition 3 (Neumann problem)**.** *Let $\Omega \subset \mathbb{R}^3$ denote a bounded domain with a twice differentiable boundary $\Gamma = \partial\Omega$ and outwards pointing normal vector $\nu = \nu(x)$, $x \in \Gamma$. Further, let $g \in C(\Gamma, \mathbb{C})$ be a complex-valued and continuous function defined on Γ and let κ denote the wavenumber. The* Neumann problem for the Helmholtz equation *is given as*

$$\Delta u(x) + \kappa^2 u(x) = 0 \quad \text{for } x \in \Omega \text{ or } x \in \mathbb{R}^3 \setminus \bar{\Omega} \tag{1.4}$$

$$\frac{\partial u}{\partial \nu}(x) = g(x) \quad \text{for } x \in \Gamma, \tag{1.5}$$

where the derivative in the boundary condition (1.5) *denotes the* normal derivative*.* In the case of scattering problems, the Dirichlet problem is used to model so-called *sound-soft* obstacles. In contrast, the Neumann problem models so-called *sound-hard* obstacles.

As shown in [13], the solutions of the interior and exterior Dirichlet and Neumann problems defined above can be represented in terms of the following *single-layer* potential $\mathcal{S}_\kappa$ and *double-layer* potential $\mathcal{D}_\kappa$,

$$\mathcal{S}_\kappa[\varphi](x) := \int_\Gamma G(x, y)\varphi(y)\, dS(y), \tag{1.6}$$

$$\mathcal{D}_\kappa[\varphi](x) := \int_\Gamma \frac{\partial G(x, y)}{\partial \nu(y)}\varphi(y)\, dS(y), \tag{1.7}$$

where

$$G(x, y) = \frac{e^{\iota\kappa|x-y|}}{4\pi|x - y|} \tag{1.8}$$

denotes the Green function associated with the Helmholtz equation 1.1 (ι denotes the imaginary unit and κ the wavenumber), and $\varphi \in C(\Gamma, \mathbb{C})$ a given *surface density*.

Remark 1. *The notation used throughout this thesis does not explicitly indicate the dependence of the Green function G on the wavenumber κ, for the sake of readability.*

For either the single-layer (1.6) or the double-layer (1.7) to be a solution to the interior/exterior Dirichlet or Neumann problem, the surface density φ is required to satisfy certain associated integral equations, which guarantee that (1.6) or (1.7) satisfy the boundary conditions (1.3) or (1.5), respectively; see, e.g., [13, Thm. 6.22-6.28]. Two particular cases are presented in what follows for the sake of concreteness.

Theorem 1. *The double-layer potential* (1.7) *($x \in \Omega$) with continuous surface density φ is a solution to the interior Dirichlet problem, as per Definition 2, provided that φ satisfies the following integral equation*

$$\varphi(x) - 2 \int_\Gamma \varphi(y) \frac{\partial G(x,y)}{\partial \nu(y)} dS(y) = -2f(x) \quad x \in \Gamma. \tag{1.9}$$

Theorem 2. *The single-layer potential* (1.6) *($x \in \mathbb{R}^3 \setminus \bar{\Omega}$) with continuous surface density φ is a solution to the exterior Neumann problem, as per Definition 3, provided that φ satisfies the following integral equation*

$$\varphi(x) - 2 \int_\Gamma \varphi(y) \frac{\partial G(x,y)}{\partial \nu(x)} dS(y) = -2g(x) \quad x \in \Gamma. \tag{1.10}$$

Remark 2. *Reference [13] bases its analysis on Riesz' theory and thus—as indicated by the above Theorems 1 and 2—focuses on* Fredholm integral equations of the second kind

$$(I + K)\varphi = f,$$

where f is some function, I the identity operator, K a compact integral operator (between suitable normed space) and φ the unknown surface density. Fredholm integral equations of the first kind

$$K\varphi = f$$

are not covered by Riesz' theory and are therefore not presented in [13]. Nevertheless, approaches based on Fredholm integral equations of the first kind are viable strategies in practice and may be used to solve the Dirichlet and Neumann problems. Even combined-layer formulations, i.e., linear combinations of the single-layer (1.6) *and the double-layer* (1.7)*, may be used, as shown in Section 5.5.*

To solve the arising boundary integral equations, as introduced in Theorems 1 and 2 (or similar BIEs associated with different problems), the occurring integral requires a suitable numerical evaluation strategy for every $x \in \Gamma$. This is a challenging problem due to the singularity in the Green function (1.8), although there are viable solutions in literature (e.g., [14, 16, 52]) and therefore not covered in this thesis. Further, provided a suitable discretization of the integral and the BIE is given, the resulting dense linear system of the form $A_N \varphi_N = f_N$ needs to be solved accurately and quickly. A straightforward inversion of the dense $N \times N$ matrix A_N would require $O(N^3)$ operations, where N denotes the number of surface discretization

points, which is clearly unfeasible for practical purposes. To reduce this cost, there are two acceleration approaches. First, the accelerated inversion of the matrix A_N, as performed by so-called *direct solvers* (see [27, Sec. 4], [28, Sec. 1], [29, Sec. 1], or [30, Sec. 2]). Secondly, the iterative solution of the linear system $A_N\varphi_N = f_N$ with the generalized minimal residual method (GMRES)—or similar iterative solvers (cf. [27, Sec. 8]). As discussed in more detail in Section 2.1, the GMRES algorithm solves the linear system by iteratively building a *Krylov subspace* through evaluation of a matrix vector product of the form $A_N r_k$ for some vector r_k in each iteration k of the algorithm. The cost associated with the evaluation of such a dense matrix-vector product $A_N r_k$, if performed naively, reduces the cubic cost of a direct inversion method to a quadratic algorithmic complexity, namely $O(N^2)$, of each iteration in the GMRES algorithm.

Remark 3. *In the present context, with reference to Remark 2, the matrix-vector product occurring in the GMRES algorithm requires the evaluation of the identity matrix, which can be performed in $O(N)$ operations and, thus, does not pose a challenge that requires further consideration, and the evaluation of discrete integral operators of the form*

$$\sum_{\substack{m=1 \\ m\neq\ell}}^{N} a_m K(x_\ell, x_m), \quad \ell = 1, \ldots, N,$$

where the discrete surface density φ_N and the discretization scheme for the boundary integral yield the coefficients a_m, $m = 1, \ldots, N$, and the kernel K denotes either the Green function G or its derivatives.

The IFGF method accelerates the evaluation of the matrix-vector product through an accumulation and approximate evaluation by interpolation of the field emitted by increasingly large groups of discretization points in a hierarchical fashion, resulting in an $O(N \log N)$ time and memory accelerated algorithm for the application of the discrete operator.

The Helmholtz equation (1.1) poses a particularly challenging problem in the high-frequency regime, i.e., for large values of the wavenumber κ (which corresponds to small values for the wavelength λ) relative to the size of the domain Ω, since i) The number of wavelengths within the domain Ω is large and therefore requires a suitable large number of surface discretization points for an accurate discrete representation and ii) The low-rank approximability of the involved discrete solution operators [16,

Sec. 3.1] [20, Sec. 4], which is the underlying property typically utilized in acceleration methods for these problems in the low-frequency regime (κ small) and Laplace case ($\kappa = 0$), does not hold for high-frequency problems.

1.2 Previous work and contribution

The development of the IFGF method was motivated by certain shortcomings inherent in integral-equation acceleration methods [37, 38, 53], all of which, including the method [37] previously developed by our research group, rely on the use of the Fast Fourier Transform (FFT). In particular, one of the goals of the development of the IFGF method was to resolve the limitations on parallel scaling capabilities of the previous methods (which result directly from corresponding difficulties associated with parallelization of the FFT [54]), while achieving optimal algorithmic complexity ($\mathcal{O}(N \log N)$). Additional aspirations driving the development of the IFGF method included a goal to bypass the complexities of existing mathematical acceleration algorithms and associated intricate computational implementations. The IFGF method presented in this thesis achieves these purposes: It does not utilize previously-employed acceleration elements such as the Fast Fourier Transform (FFT), special-function expansions, high-dimensional linear-algebra factorizations, translation operators, equivalent sources, or parabolic scaling [31–33, 36–38, 40–42, 53, 55, 56]. Instead, the IFGF method relies on straightforward interpolation of the operator kernels—or, more precisely, of certain factored forms of the kernels—which, when collectively applied to larger and larger groups of Green function sources, in a recursive fashion, gives rise to the desired $\mathcal{O}(N \log N)$ accelerated evaluation. In what follows, we compare the serial and parallel IFGF method to existing methods, and emphasize the differences.

As alluded to above, the IFGF strategy is based on the interpolation properties of a certain factored form of the scattering Green function into a singular and rapidly-oscillatory *centered factor* and a slowly-oscillatory *analytic factor*. Importantly, the analytic factor is analytic up to and including infinity (which enables interpolation over certain unbounded conical domains on the basis of a finite number of radial interpolations nodes), and, when utilized for interpolation of fields with sources contained within a cubic box B of side H, it enables uniform approximability over semi-infinite cones, with apertures proportional to $1/H$. In particular, unlike the FMM based approaches (e.g., [31, 32]), the algorithm does not require separate treatment of the low- and high-frequency regimes. On the basis of these prop-

erties, the IFGF method orchestrates the accelerated operator evaluation utilizing two separate tree-like hierarchies which are combined in a single boxes-and-cones hierarchical data structure. Thus, starting from an initial cubic box of side H_1 which contains all surface discretization points considered, the algorithm utilizes, like other approaches, the octree $\mathcal{B}$ of boxes that is obtained by partitioning the initial box into eight identical child boxes of side $H_2 = H_1/2$ and iteratively repeating the process with each resulting child box until the resulting boxes are sufficiently small.

Along with the octree of boxes, the IFGF algorithm incorporates a hierarchy $\mathcal{C}$ of spherical *cone segments*, which are used to enact the required interpolation procedures. Each box in the tree $\mathcal{B}$ is thus endowed with a set of box-centered spherical cone segments at a corresponding level of the cone hierarchy $\mathcal{C}$. In detail, a set of box-centered cone segments of extent $\Delta_{s,d}$ in the analytic radial variable s, and angular apertures $\Delta_{\theta,d}$ and $\Delta_{\varphi,d}$ in each of the two spherical angular coordinates θ and φ, are used for each d-level box $B \in \mathcal{B}$. (Roughly speaking, $\Delta_{s,d}$, $\Delta_{\theta,d}$ and $\Delta_{\varphi,d}$ vary in an inversely proportional manner with the box size H_d for large enough boxes, but they remain constant for small boxes; full details are presented in Chapter 3.) The set of cone segments centered at a box $B \in \mathcal{B}$ is used by the IFGF algorithm to set up an interpolation scheme over all of space around B, except for the region occupied by the union of B itself and all of its nearest neighboring boxes at the same level. Thus, the leaves (level D) in the box tree, that is, the cubes of the smallest size used, are endowed with cone segments of largest angular and radial spans $\Delta_{s,D}$, $\Delta_{\theta,D}$ and $\Delta_{\varphi,D}$ considered. Each ascent $d \to (d-1)$ by one level in the box tree $\mathcal{B}$ (leading to an increase by a factor of two in the cube side $H_{d-1} = 2H_d$) is accompanied by a corresponding descent by one level (also $d \to (d-1)$) in the cone hierarchy $\mathcal{C}$ (leading, e.g., for large boxes, to a decrease by a factor of one-half in the radial and angular cone spans: $\Delta_{s,d-1} = \frac{1}{2}\Delta_{s,d}$, $\Delta_{\theta,d-1} = \frac{1}{2}\Delta_{\theta,d}$ and $\Delta_{\varphi,d-1} = \frac{1}{2}\Delta_{\varphi,d}$; see Section 3.3). In view of the interpolation properties of the analytic factor, the interpolation error and cost per point resulting from this conical interpolation setup remains unchanged from one level to the next as the box tree is traversed towards its root level $d = 1$. The situation is even more favorable in the small-box case. And, owing to analyticity at infinity, interpolation for arbitrarily far regions within each cone segment can be achieved on the basis of a finite amount of interpolation data. In all, this strategy reduces the computational cost, by commingling the effect of large numbers of sources into a small number of interpolation parameters. A recursive strategy, in which cone segment interpolation data at level d is also exploited to

obtain the corresponding cone-segment interpolation data at level $(d-1)$, finally, yields the optimal $\mathcal{O}(N \log N)$ approach.

The properties of the factored Green function, which underlie the proposed IFGF algorithm, additionally provide certain perspectives concerning various algorithmic components of other acceleration approaches. In particular, the analyticity properties of the analytic factor, which are established in Theorem 4, in conjunction with the classical polynomial interpolation bound presented in Theorem 3, and the IFGF spherical-coordinate interpolation strategy, clearly imply the property of low-rank approximability which underlies some of the ideas associated with the butterfly methods [39–41] and directional FMM [32]. The directional FMM approach, further, relies on a "directional factorization" which, in the context of the present interpolation-based viewpoint, can be interpreted as facilitating interpolation. For the directional factorization to produce beneficial effects it is necessary for the differences of source and observation points to lie on a line asymptotically parallel to the vector between the centers of the source and target boxes. This requirement is satisfied in the directional FMM approach through its "parabolic scaling", according to which the distance to the observation set is required to be the square of the size of the source box. The IFGF factorization is not directional, however, and it does not require use of the parabolic scaling: the IFGF approach interpolates analytic-factor contributions at linearly-growing distances from the source box.

In a related context we mention the recently introduced approach [42], which incorporates in an $\mathcal{H}^2$-matrix setting some of the main ideas associated with the directional FMM algorithm [32]. Like the IFGF method, the approach relies on interpolation of a factored form of the Green function—but using the directional factorization instead of the IFGF factorization. The method yields a full LU decomposition of the discrete integral operator, but it does so under significant computing costs and memory requirements, both for precomputation, and per individual solution.

It is also useful to compare the IFGF approach to other acceleration methods from a purely algorithmic point of view. The FMM-based approaches [31, 32, 35, 55] entail two passes over the three-dimensional acceleration tree, one in the upward direction, the other one downward. In the upward pass of the original FMM methods, for example, the algorithm commingles contributions from larger and larger numbers of sources via correspondingly growing spherical-harmonics expansions, which are sequentially translated to certain spherical coordinate systems and then recombined,

as the algorithm progresses up the tree via application of a sequence of so-called M2M translation operators (see, e.g., [32]). In the downward FMM pass, the algorithm then re-translates and localizes the spherical-harmonics expansions to smaller and smaller boxes via related M2L and L2L translation operators (e.g., [32]). The algorithm is finally completed by evaluation of surface point values at the end of the downward pass. The IFGF algorithm, in contrast, progresses simultaneously along two tree-like structures, the box tree and the cone interpolation hierarchy, and it produces evaluations at the required observation points, via interpolation, at all stages of the acceleration process (but only in a neighborhood of each source box at each stage). In particular, the IFGF method does not utilize high-order expansions of the kinds used in other acceleration methods—and, thus, it avoids use of Fast Fourier Transforms (FFTs) which are almost invariably utilized in the FMM to manipulate the necessary spherical harmonics expansions. (Reference [14, Sec. 7] mentions two alternatives which, however, it discards as less efficient than an FFT-based procedure.)

As indicated above, the IFGF algorithm, which relies on interpolation by means of Chebyshev expansions of relatively low degree, does not require the use of FFTs—a fact that provides significant benefits in the distributed memory context. As a counterpoint, however, the low degree Chebyshev approximations used by the IFGF method do not yield the spectral accuracy resulting from the high-order expansions used by other methods. A version of the IFGF method which enjoys spectral accuracy could be obtained simply by replacing its use of low-order Chebyshev interpolation by Chebyshev interpolation of higher and higher orders on cone segments of fixed size as the hierarchies are traversed toward the root $d = 1$. Such a direct approach, however, entails a computing cost which increases quadratically as the Chebyshev expansion order grows—thus degrading the performance of the IFGF method. But the needed evaluation of high-order Chebyshev expansions on arbitrary three-dimensional grids can be performed by means of FFT-based interpolation methods similar to those utilized in [37, Sec. 3.1] and [57, Remark 7]. This approach, which is not pursued in this thesis, would lead to a spectrally convergent version of the method, which still runs on essentially linear computing time and memory. And, despite reverting to the use of FFTs, the strategy may perform well in a parallel setting, since the number of the involved FFTs and their sizes would be essentially constant from one level in the octree structure to the next.

It is also relevant to contrast the algorithmic aspects in the IFGF approach to those used in the butterfly approaches [39–41]. Unlike the interpolation-based IFGF method, which does not rely on the use of linear-algebra factorizations, the butterfly approaches are based on low-rank factorizations of various high-dimensional sub-matrices of the overall system matrix. Certain recent versions of the butterfly methods reduce linear-algebra computational cost by means of an interpolation process in high-dimensional space in a process which can easily be justified on the basis of the analytic properties of the factored Green function described in Section 3.1. As in the IFGF approach, further, the data structure inherent in the butterfly approach [40, 41] is organized on the basis of two separate tree structures that are traversed in opposite directions, one ascending and the other descending, as the algorithm progresses. In the method [41] the source and observation cubes are paired in such a way that the product their sizes remains constant—which evokes the IFGF's cone-and-box sizing condition, according to which the angles scale inversely with the cone span angles. These two selection criteria are indeed related, as the interpolability by polynomials used in the IFGF approach has direct implications on the rank of the interpolated values. But, in a significant distinction, the IFGF method can be applied to a wide range of scattering kernels, including the Maxwell, Helmholtz, Laplace and elasticity kernels among others, and including smooth as well as non-smooth kernels. The butterfly approaches [39, 41], in contrast, only apply to Fourier integral operators with smooth kernels. The earlier butterfly contribution [40] does apply to Maxwell problems, but its accuracy, specifically in the low-frequency near-singular interaction regime, has not been studied in detail.

Since the emergence of parallel computers, the parallelization of accelerated Green function methods has been the subject of a significant literature, which is mostly devoted to tackling a particular difficulty, namely, the "parallelization bottleneck"—which manifests itself under various related guises [34, 44–51], and which almost invariably concerns uses of the hard-to-parallelize [54] FFT algorithm. In the case of the multilevel FMM, the parallelization bottleneck arises in the evaluation of translation operators associated with the upper part of the octree structure, which leads to low parallel efficiency [32, 44, 49]. In the directional FMM [44] the low efficiency in the upper octree is alleviated as a result of the parabolic scaling utilized; however, the parallelization strategy does suffer from hard limitations in the number of parallel tasks that, in the cases considered in that reference, lead to a "leveling off" of the parallel scaling at 256 or 512 cores [44, Secs. 3.6, 4.2], depending on the geometry under consideration. Reference [45] identifies the part

of the FMM relying on FFTs as a parallelization bottleneck which arises from FFT-related "lowest arithmetic intensity" and "bandwidth contention." In [46, 47], in turn, a hybrid octree storage strategy is used, which stores a complete set of tree nodes for a certain number of "full" levels in each process, and which reduces the communication in the upper octree levels. Those articles demonstrate the treatment of problems containing very large numbers of discretization points on up to $2,560$ processes, but they restrict their illustration of the algorithm's parallel efficiency to a limited strong scaling test from 1 process (sequential) to 64 processes. In contrast to this hybrid octree-storage strategy, reference [48] simultaneously partitions boxes (clusters) and field values representing the radiating and incoming fields of each box. This approach leads to increased efficiency compared to a parallelization purely based on the boxes (clusters), but the communication in the translation step still poses a bottleneck, resulting in as little as 30% parallel efficiency from one (sequential) core to 128 cores.

Reference [58], in turn, presents scaling results for the parallel BEMFMM implementation of the FMM algorithm, for wave scattering problems on up to 196,608 cores on 6,144 compute nodes, and for problems with up to 2.3 billion degrees of freedom (DOF) and approximately 1,389 wavelengths in size (or, in the nomenclature of Table 2 in [58], a sphere two-meters in diameter illuminated at the frequency of $f = 238.086$ KHz, under the assumption of a 343m/s speed of sound). Like the implementations mentioned above, the results in [58] indicate a deterioration of the strong-scaling for growing numbers of cores, as manifested by a flattening of the strong-scaling speedup curves presented as the numbers of cores increase. The weak scaling curve presented in [58] indicate a high weak-scaling efficiency, however, with up to 95% efficiency for weak scaling between 32 and 131,072 cores. Comparison with BEMFMM and IFGF weak scaling results presents some challenges on a number of counts. On one hand the contribution [58] does not mention a crucial element in judging parallelization quality, namely, memory usage: even though memory duplication may be relied upon in a parallel algorithm to maximize parallel efficiency, no indications are provided in that paper about the amount of memory used in any of the runs presented. Further, under closer examination, the computing times indicated in these curves appeared to be high, and we thus decided to perform a direct comparison of the performance of our IFGF implementation with the BEMFMM implementation on the basis of the freely available BEMFMM open-source download provided by the authors. By necessity, our tests were limited to a test example consisting of a sphere containing approximately 360,000 DOF,

which is the largest test case provided with the BEMFMM test code, and we selected a sphere of acoustic diameter of 16λ for this experiment. We run both algorithms in the 30 available nodes in our cluster, Wavefield, each one of which contains 56 computing cores. Our observations are as follows. The BEMFMM run for the test case considered required 20 secs. in a single node, and 5 secs. in all 30 nodes, with a speedup factor of 4 going from 1 to thirty nodes. The IFGF run, in turn, required 1.6 secs. in a single node, and 0.122 secs in the thirty node cluster, with a speedup factor of 13 going from 1 to 30 nodes. Thus, the IFGF runs in one and 30 nodes were faster than the BEMFMM runs by factors of 12.5 and 40 in the 1-node and 30-node runs, respectively, with an IFGF speedup over three times higher than that provided by BEMFMM going from 1 to 30 nodes. As an additional point of contact with reference [58], it is worth mentioning that, in our 1,680 core cluster, and on the basis of approximately 4 TB of memory, a sphere 1,389 λ in diameter (reported as $f = 238.086$KHz at a speed of sound of $343m/s$ in [58]; cf. Section 5.8 for details) with 2.12 billion DOF was run in a computing time of 2,380 seconds (Table 5.25 in Section 5.8), which, with a 0.5% near-field error (which may be compared to the only error indicator reported in [58, Table 2] for this test case, which amounts to 20%, as well as the 3% near-field solution error reported in the same table of that paper for significantly smaller problems), is a factor of approximately 46 times longer than the time reported in [58], for the same number of DOFs and sphere size, on a computer 78 times larger (containing 131,072 cores) and on the basis of an unspecified amount of memory. Additional test cases for large sphere problems are presented in Section 5.8.

Following a different approach, to avoid the communication bottleneck in the upper multilevel FMM octree entirely, references [34, 49] utilize a single-level Fast Multipole strategy. While this method significantly simplifies the algorithm and minimizes the required communication in a parallel setting, it does give rise to a sub-optimal asymptotic computational cost (e.g., $O(N^{3/2})$ in [34] or, exploiting the FFT, $O(N^{4/3}\log^{2/3} N)$ in [49]), and, while resulting in good parallel scaling up to 512 processes in the $O(N^{3/2})$ algorithm [34], as in the case of [44], the parallel efficiency does level off beyond 512 processes. Direct FFT methods, in turn, present alternatives to the various FMM strategies, including, for example, the Adaptive Integral Method [38] (AIM) and the sparse-FFT method [37]. Like the single-level FMM algorithms, these FFT methods exhibit sub-optimal algorithmic complexity (of orders $O(N^{3/2})$ and $O(N^{4/3})$, respectively, and, owing to their strong reliance on FFTs, they also suffer from reduced parallel efficiency, as shown and discussed

for the AIM in, e.g., [50, 51]. (A parallel version of the algorithm [37], which has been developed by the authors, has not been published, but we report here that, as may have been expected, the overall parallel efficiency of the method suffers from the typical FFT-related degradation.)

Finally, we mention parallel methods proposed for non-singular [41] and low-frequency [59, 60] problems which, albeit important and interesting, do not incur some of the main challenges associated with the singular and high-frequency kernels considered in this thesis. Thus, although not applicable to singular Green function kernels such as the ones considered here, the Butterfly Method [41] does provide an acceleration technique for Fourier integral operators which, based on linear-algebra constructs instead of the hierarchical interpolation underlying the IFGF approach, incorporates a parallelization strategy that is somewhat reminiscent of the proposed IFGF parallelization approach. The Blue Gene/Q implementation [41] of the Butterfly parallel algorithm demonstrates excellent results in terms of parallel scaling to a large number of cores. The parallel FMM method presented in [60], which is restricted to box geometries and to the Laplace and low-frequency Helmholtz problems, shows impressive scaling up to 299,008 cores on 18,688 nodes. Similarly, the parallel Barnes-Hut tree code [59] for the low-frequency singular problem provides excellent scaling up to 294,912 cores with up to 2,048,000,000 particles.

The parallel IFGF strategy introduced in this thesis is based on adequate partitioning of the interpolations performed on each level of the underlying octree structure, which facilitates the spatial decomposition of the surface discretization points. As discussed in Section 3.6, the number of interpolations performed on each level is large and approximately constant (as a function of the octree level). The decomposition and distribution of the interpolation data is based on a total order in the set of spherical cone segments representing the interpolation domains, which is an extension of a domain decomposition based on a Morton curve to the box-cone data structure inherent in the IFGF approach. The usage of space-filling curves for the representation of octree structures underlying the various acceleration methods is not a novel concept [58, 60, 61]. However, the extension of space-filling curves to the present box-cone structure of the IFGF method to achieve the desired efficiency has not been reported before. In view of its strong reliance on the IFGF's box-cone structure, the proposed parallelization strategy is therefore not applicable to other acceleration methods such as the FMM. The present parallel IFGF implementation on a 30-node (1,680-core) HPC cluster with Infiniband interconnect, delivers per-

fect $O(N \log N)$ performance on all 1,680 cores. And, demonstrating high (albeit imperfect) strong and weak parallel efficiencies, unlike other methods, it does not suffer from scaling limitations, under either weak scaling or strong scaling tests, as the number of processing cores grow—conceivably, as argued in Sections 4.2 and 5.6, up to arbitrarily large numbers of cores.

1.3 Content and layout of this thesis

This thesis consists of a total of six chapters. The present Chapter 1 provides an overview of the integral-equation formulations of scattering theory, it reviews previously proposed acceleration techniques in the area, and it presents the main motivations and goals for the development of the proposed IFGF method. Chapter 2 then provides an overview of known methods and concepts relevant to the discussion of the IFGF method and integral equations in general. It includes an overview of the GMRES algorithm for the iterative solution of systems of linear equations, a brief summary of Chebyshev interpolation techniques, and an overview of the nomenclature utilized in the area of high performance computing (HPC). Readers familiar with boundary integral equations may choose to skip these first two chapters. The actual description of the novel IFGF method starts in Chapter 3, where the IFGF theoretical basis is introduced and a serial IFGF algorithm is presented. Chapter 4 introduces a hybrid MPI-OpenMP IFGF parallelization strategy suitable for implementation on large computer cluster systems. The serial and parallel IFGF algorithms are numerically validated in Chapter 5 on the basis of several geometries and test cases. Finally, Chapter 6 presents a number of concluding remarks, and it provides an outline of possible future research projects and open questions closely related to the IFGF method.

Chapter 2

PRELIMINARIES

This chapter briefly reviews background concepts which are highly relevant in the context of integral equations and the IFGF method proposed in this thesis. In particular, as discussed above, the IFGF and other acceleration methods are used in practice in conjunction with iterative approaches for the solution of integral equation problems. Thus, in Section 2.1 we review the iterative linear-algebra solver that is preferred in this context, namely, the GMRES algorithm. Section 2.2, in turn, reviews one- and three-dimensional Chebyshev polynomial approximation, which form the basis of the IFGF interpolation strategy. Finally, Section 2.3 reviews concepts and nomenclature in the field of high performance computing which are used, in particular, for the presentation of the proposed IFGF parallelization strategy in Chapter 4.

2.1 GMRES

As indicated above, the present thesis is concerned with the fast evaluation of matrix-vector products of the form $A_N v_N$, where $A_N \in \mathbb{C}^{N\times N}$ represents some discretization of an integral operator (cf. Section 1.1) and $v_N \in \mathbb{C}^N$ denotes some vector, which occurs in the iterative solution of discrete integral equations of the form $A_N \varphi_N = f_N$. To provide context we briefly mention the *Generalized Minimal Residual* (GMRES) algorithm [62], which is the algorithm typically utilized for the iterative solution of discretized integral equations. The use of the GMRES algorithm and other iterative *Krylov subspace* linear equation solvers in the context of the integral equation problems under consideration motivate our treatment of the fundamental problem considered in this thesis, namely, the accelerated evaluation of discrete integral operators. The details of the GMRES algorithm, which do not impact upon the innovations introduced in this thesis, are not discussed here in any detail. For quick reference we include the pseudocode 1 of the GMRES algorithm for real matrices $A \in \mathbb{R}^{N\times N}$ (cf. [16, Algorithm C.4]), and we additionally refer to [62], [27, Sec. 8.7.2] and [16, Sec. C.3.2] in this regard. Lines 1 and 6 in Algorithm 1 display the matrix-vector product which is accelerated by the IFGF algorithm: a direct evaluation of this product requires $O(N^2)$ arithmetic operations—a requirement which, in the context of the high-frequency scattering problems relevant in this thesis, is often computationally prohibitive. As discussed in Chapter 1, the proposed IFGF method enables the evaluation of the matrix-vector product in a linearithmic ($O(N \log N)$) number of operations, and it provides a number of important advantages over other available acceleration methodologies.

2.2 Chebyshev interpolation

The IFGF method is based upon a hierarchical interpolation strategy of a certain factored form of the Green function. More precisely, the interpolation of the factored Green function is facilitated in piece-wise fashion in certain spherical coordinate systems resulting in the IFGF *cone segments*, as presented in Section 3.5. Theoretically, any interpolation method with sufficient accuracy would suffice to achieve the desired asymptotic acceleration, but, throughout this thesis, we utilize a three-dimensional Chebyshev interpolation procedure in view of its accuracy and efficiency. For a self-contained presentation, the present section therefore briefly reviews Chebyshev polynomials in one and three dimensions and discusses some possibilities for an efficient practical implementation of Chebyshev-based interpola-

Algorithm 1 GMRES

Require: $A \in \mathbb{R}^{N\times N}$, $f \in \mathbb{R}^N$, $x^0 \in \mathbb{R}^N$, $\epsilon > 0$

$r^0 = f - Ax^0$
$\rho^0 = \left\|r^0\right\|$
$v^1 = r^0/\left\|r^0\right\|$
$p^0 = \rho^0$
for $k = 0, \ldots, N-2$ **do**
    $w^k = Av^k$
    $\hat{v}^{k+1} = w^k$
    for $\ell = 0, \ldots, k$ **do**
        $\beta^{k,\ell} = w^k \cdot v^\ell$
        $\hat{v}^{k+1} = \hat{v}^{k+1} - h^{k,\ell} v^\ell$
    end for
    $\beta^{k+1,k} = \left\|\hat{v}^{k+1}\right\|$
    if $\beta^{k+1,k} = 0$ **then** Leave "for"-loop
    end if
    $v^{k+1} = \hat{v}^{k+1}/\beta^{k+1,k}$
    for $\ell = 0, \ldots, k-1$ **do**
        $\tilde{\beta}^{k,\ell} = a^\ell \beta^{k,\ell} + b^\ell \beta^{k,\ell+1}$
        $\tilde{\beta}^{k,\ell+1} = -b^l \beta^{k,\ell} + a^\ell \beta^{k,\ell+1}$
    end for
    $a^k = \beta^{k,k}/\sqrt{(\beta^{k,k})^2 + (\beta^{k,k+1})^2}$
    $b^k = \beta^{k,k+1}/\sqrt{(\beta^{k,k})^2 + (\beta^{k,k+1})^2}$
    $\tilde{\beta}^{k,k} = \sqrt{(\beta^{k,k})^2 + (\beta^{k,k+1})^2}$
    $p^{k+1} = -b^k p^k$, $p^k = a^k p^k$, $\rho^{k+1} = |p^{k+1}|$
    if $\rho^{k+1} < \epsilon\rho^0$ **then** Leave "for"-loop
    end if
end for
$x^{\text{solution}} = x^0$
for $\ell = k, k-1, \ldots, 0$ **do**
    $\alpha^\ell = \frac{1}{\beta^{\ell,\ell}}\left(p^\ell - \sum_{j=\ell+1}^{k} \beta^{\ell,j}\alpha^j\right)$
    $x^{\text{solution}} = x^{\text{solution}} - \alpha^\ell v^\ell$
end for
return x^{solution}

tion procedures. For a more thorough introduction to Chebyshev interpolation, we refer to [30, 63].

For a given function $u : [-1,1] \to \mathbb{C}$ over the reference interval $[-1,1]$, the one-dimensional interpolation polynomial $I_n^{\text{ref}} u$ of accuracy order n produced via Chebyshev interpolation is given by the expression

$$I_n^{\text{ref}} u(x) = \sum_{i=0}^{n-1} a_i T_i(x), \quad x \in [-1,1], \tag{2.1}$$

where $T_i(x) = \cos(i \arccos(x))$ denotes the i-th Chebyshev polynomial of the first kind, and where, letting

$$x_k = \cos\left(\frac{2k+1}{2n}\pi\right), \quad \text{and} \quad \alpha_i = \begin{cases} 2 & i \neq 0 \\ 1 & i = 0, \end{cases} \tag{2.2}$$

the coefficients $a_i \in \mathbb{C}$ are given by

$$a_i = \frac{\alpha_i}{n} \sum_{k=0}^{n-1} u(x_k) T_i(x_k) \tag{2.3}$$

(see [30, (5.8.7) and (5.8.8)]). Chebyshev interpolation for functions defined on arbitrary intervals $[a,b]$ are obtained via a linear re-scaling to the reference interval $[-1,1]$; for notational simplicity, the corresponding interpolating polynomial in the interval $[a,b]$ is denoted by $I_n u$, without explicit reference to the interpolation interval $[a,b]$.

As is known ([28, Sec. 7.1], [64]), the one-dimensional Chebyshev interpolation error $|u(x) - I_n u(x)|$ in the interval $[a,b]$ satisfies the bound

$$|u(x) - I_n u(x)| \leq \frac{(b-a)^n}{2^{2n-1} n!} \left\| \frac{\partial^n u}{\partial x^n} \right\|_\infty, \tag{2.4}$$

where

$$\left\| \frac{\partial^n u}{\partial x^n} \right\|_\infty := \sup_{c \in (a,b)} \left| \frac{\partial^n u}{\partial x^n}(c) \right| \tag{2.5}$$

denotes the supremum norm of the n-th partial derivative.

In the context of the IFGF method, the generalization

$$I_n^{\text{ref},3} u(x,y,z) = \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \sum_{k=0}^{n_z-1} a_{i,j,k} T_i(x) T_j(y) T_k(z), \quad (x,y,z) \in [-1,1]^3, \tag{2.6}$$

of (2.1) to three dimensions is of interest, where, as in the one-dimensional case (2.3), using once again (2.2), we obtain

$$a_{i,j,k} = \frac{\alpha_i \alpha_j \alpha_k}{n_x n_y n_z} \sum_{m=0}^{n_x-1} \sum_{n=0}^{n_y-1} \sum_{o=0}^{n_z-1} u(x_m, x_n, x_o) T_i(x_m) T_j(x_n) T_k(x_o), \tag{2.7}$$

for $i = 0, \ldots, n_x - 1$, $j = 0, \ldots, n_y - 1$, and $k = 0, \ldots, n_z - 1$. A general version of the one-dimensional error estimate (2.4) is deferred to Section 3.3, in view of to its close relation with the theoretical basis of the IFGF method.

The direct computation of (2.7) and (2.6) is costly due to the evaluation of the triple sums. The repeated evaluation of polynomials $I_n^{\mathrm{ref},3} u(x, y, z)$ in (2.6), in particular, is the most cost intensive part of the IFGF method, and, thus, algorithms for the accelerated evaluation of these polynomials can reduce the computational effort of the overall IFGF method significantly. In the present implementation of the IFGF method, (2.6) is evaluated naively due to the non-uniformity of the targets and the small expansion sizes. Accelerated evaluation algorithms based on, e.g., small non-uniform FFTs (cf. [63, Sec. 10], [37, Sec. 3.1], [57, Remark 7]) are currently under investigation to further enhance the performance of the IFGF method. On the other hand, the coefficients (2.7) are currently evaluated with an accelerated "partial summation" algorithm as shown in [63, Sec. 10.2].

2.3 HPC basics

The proposed IFGF algorithm, the parallel IFGF algorithm presented in Chapter 4 in particular, is designed for implementation in modern HPC *cluster* systems. The present section reviews relevant hardware and software concepts and nomenclature utilized throughout this thesis; more detailed descriptions and alternative hardware designs can be found, e.g., in [65–67]. A modern computer cluster consists of multiple compute *nodes*. Each node contains its own memory space, and thus the memory in the cluster is distributed between the nodes. In particular, access to memory in other compute nodes requires explicit data communication, which is typically performed via the *message passing interface* (MPI) [66, Sec. 8]; the performance of algorithmic implementations for cluster systems can therefore significantly benefit from careful engineering of MPI-based inter-process data communications.

Each node typically comprises one or a few *multi-core processors*, each one of which, as the name suggests, contains multiple computing *cores*. The compute nodes typically are so-called *shared memory machines* (SMMs), where each core

within the node can access all of the memory in the node. To efficiently make use of more than a single core, certain specialized programming techniques are required, e.g., the Intel Threading Building Blocks (TBB) library, the C++ standard threading model, MPI, or the OpenMP programming interface. Modern compute nodes usually follow a *non-uniform memory access* (NUMA) design (in contrast to uniform memory access (UMA)), where the access times to the shared memory depend on the locality of the memory with respect to the multi-core processor accessing it. This design typically results in one or more NUMA nodes per compute node, where memory access to other NUMA nodes on the same compute node is usually significantly slower than access to memory local to the processor. All of the tests presented in this thesis were conducted on a small cluster consisting of thirty nodes connected via an InfiniBand interconnect, each one of which contains four fourteen-core NUMA nodes; additional details concerning the hardware used are provided in Section 5.1.

On the basis of the functions and synchronization capabilities provided by MPI, a program can be launched as a set of multiple *processes* (which are identified in what follows by their corresponding integer-valued *rank* within the group of all processes launched by a given program). One of the main roles of the MPI standard is to allow the programmer to orchestrate the data communications between the ranks. Note that, at runtime, an MPI rank can be assigned, or *pinned*, to various kinds of hardware units, such as, e.g., a single core, a NUMA node, a complete compute node, or various combinations of cores and/or nodes.

Chapter 3

THE INTERPOLATED FACTORED GREEN FUNCTION METHOD

As discussed in Chapter 1, the IFGF method [68] provides an accelerated algorithm, requiring $O(N \log N)$ operations, for the approximate numerical evaluation of discrete integral operators (cf. Section 1.1) of the form

$$I(x_\ell) \coloneqq \sum_{\substack{m=1 \\ m\neq\ell}}^{N} a_m G(x_\ell, x_m), \quad \ell = 1, \dots, N, \tag{3.1}$$

for distinct points $x_\ell \in \Gamma$ on a surface $\Gamma \subset \mathbb{R}^3$, and for given complex coefficients $a_m \in \mathbb{C}$, where the function $G : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{C}$ denotes a Green function for some partial differential equation (or derivatives thereof, as discussed in Section 1.1), such as the acoustic Green function (1.8) associated with the Helmholtz equation as well as those associated with the Laplace, Stokes, and elasticity equations, among others. In other words, the IFGF method generates approximations $I_{\text{acc}}(x_\ell) \approx I(x_\ell)$, $\ell = 1, \dots, N$, of the discrete integral operator operator (3.1) in $O(N \log N)$ operations instead of $O(N^2)$. For ease of the notation, in what follows, $\Gamma_N \coloneqq \{x_1, \dots, x_N\} \subset \Gamma$ denotes the set of surface discretization points. To achieve its $O(N \log N)$ computational complexity, the IFGF method is based on the following main ideas, which are described in detail in the remainder of this chapter. The first one of these main ideas concerns the use of a factorization of the Green function G into a so-called *centered factor*, which is an easily evaluated common factor, and a so-called *analytic factor*, which is under certain assumptions slowly oscillatory and analytic up to and including infinity, and, thus, easily interpolated (see Sections 3.1-3.3). Secondly, as a result of the aforementioned factorization of the Green function inherent in the IFGF method, an octree-based hierarchical partitioning, denoted by $\mathcal{B}$ in what follows, of the surface discretization points Γ_N into axis-aligned, equi-sized, and pairwise disjoint *boxes*, as described in Section 3.4, is used to facilitate the discrete operator evaluation (3.1) through certain pairwise interactions of these boxes. Finally, and most importantly, the pairwise interactions of boxes in the box-octree structure $\mathcal{B}$ is facilitated through the evaluation and accumulation of the fields emitted by the point sources contained in each box in an iterative and hierarchical fashion utilizing Chebyshev interpolation. More precisely, the IFGF

interpolation procedure is a piece-wise Chebyshev interpolation procedure in certain box-centered spherical coordinate systems, where the interpolation domains are represented (in real space) by a hierarchy $\mathcal{C}$ of spherical *cone segments*. The details concerning the definition of these cone segments and the Chebyshev interpolation performed on them, including a suggested refinement strategy to optimize their sizes in dependence of their position in the underlying box-octree structure, are presented in Sections 3.3 and 3.5. After the presentation of these fundamental elements of the IFGF approach, Section 3.6 summarizes the complete IFGF algorithm, before the algorithmic complexity of the resulting algorithm is analyzed in Section 3.7.

3.1 Factorization of the Green function

For the presentation of the IFGF factorization, we first introduce the box, source-point, and target-point notations we use in what follows. To that end, for given $H > 0$ and $x = ((x)_1, (x)_2, (x)_3)^T \in \mathbb{R}^3$, we define an *axis aligned box*—a Cartesian product of one-dimensional half-open intervals—as follows.

Definition 4 (Box)**.** *Let $H > 0$ and let $x = (x_1, x_2, x_3) \in \mathbb{R}^3$. A* box $B(x, H)$ centered *at x of* size *(or* side*) $H > 0$ is defined as the Cartesian product of one-dimensional, half-open intervals of the following form.*

$$B(x, H) \coloneqq \left[x_1 - \frac{H}{2}, x_1 + \frac{H}{2}\right) \times \left[x_2 - \frac{H}{2}, x_2 + \frac{H}{2}\right) \times \left[x_3 - \frac{H}{2}, x_3 + \frac{H}{2}\right).$$

The radius *h of a box $B(x, H)$, in turn, is defined as the largest Euclidean distance of any point in the box to its center.*

$$h \coloneqq \max_{x \in B(x=0,H)} |x| = \frac{\sqrt{3}}{2} H. \tag{3.2}$$

For a given *source box* $B(x_S, H)$ of side H and centered at a given point $x_S = ((x_S)_1, (x_S)_2, (x_S)_3)^T \in \mathbb{R}^3$, we use the enumeration $x_1^S, \ldots, x_{N_S}^S \in B(x_S, H) \cap \Gamma_N$ ($N_S \leq N$ and, possibly, $N_S = 0$) of all source points x_m, $m = 1, \ldots, N$, which are contained in $B(x_S, H)$; the corresponding source coefficients a_m are denoted by $a_\ell^S \in \{a_1, \ldots, a_N\}$, $\ell = 1, \ldots, N_S$. A given set of N_T surface target points, at arbitrary positions outside $B(x_S, H)$, are denoted by $x_1^T, \ldots, x_{N_T}^T \in \Gamma_N \setminus B(x_S, H)$. Then, letting $I_S(x)$ denote the field generated at a point x by all point sources contained in $B(x_S, H)$, we will consider, in particular, the problem of evaluation of the local operator

$$I_S(x_\ell^T) \coloneqq \sum_{m=1}^{N_S} a_m^S G(x_\ell^T, x_m^S), \qquad \ell = 1, \ldots, N_T. \tag{3.3}$$

A sketch of this setup is presented in Figure 3.1.

To achieve the desired acceleration of the discrete operator (3.1), the IFGF approach utilizes a certain factorization of the Green function G which leads to efficient evaluation of the field I_S in equation (3.3) by means of numerical methods based on polynomial interpolation.

The IFGF factorization for x' in the box $B(x_S, H)$ (centered at x_S) takes the form

$$G(x, x') = G(x, x_S) g_S(x, x'). \tag{3.4}$$

Throughout this thesis, the functions $G(x, x_S)$ and g_S are called the *centered factor* and the *analytic factor*, respectively. Clearly, for a fixed given center x_S, the centered factor depends only on x: it is independent of x'. As shown in Section 3.2, in turn, the analytic factor is *analytic up to and including infinity* in the x variable for each fixed value of x' (which, in particular, makes $g_S(x, x')$ slowly oscillatory and asymptotically constant as a function of x as $|x| \to \infty$), with oscillations as a function of x that, for $x' \in B(x_S, H)$, increase linearly with the box size H.

Using the factorization (3.4), the field I_S generated by point sources placed within the source box $B(x_S, H)$ at any point $x \in \mathbb{R}^3$ may be expressed in the form

$$\begin{aligned} I_S(x) &:= \sum_{m=1}^{N_S} a_m^S G(x, x_m^S) = G(x, x_S) F_S(x) \quad \text{where} \\ F_S(x) &:= \sum_{m=1}^{N_S} a_m^S g_S(x, x_m^S). \end{aligned} \tag{3.5}$$

The desired IFGF accelerated evaluation of the operator (3.3) is achieved via interpolation of the function $F_S(x)$, which, as a linear combination of analytic factors, is itself analytic at infinity. The singular and oscillatory character of the function F_S, which determine the cost required for its accurate interpolation, can be characterized in terms of the analytic properties, mentioned above, of the factor g_S. A study of these analytic and interpolation properties is presented in Sections 3.2 and 3.3.

On the basis of the aforementioned analytic properties the algorithm evaluates all the sums in equation (3.3) by first obtaining values of the function F_S at a small number $P \in \mathbb{N}$ of points $p_i \in \mathbb{R}^3$, $i = 1, \ldots, P$, from which the necessary I_S values (at all the target points $x_1^T, \ldots, x_{N_T}^T$) are rapidly and accurately obtained by interpolation. At a cost of $O(PN_S + PN_T)$ operations, the interpolation-based algorithm yields useful acceleration provided $P \ll \min\{N_S, N_T\}$. Section 3.6 shows that adequate utilization of these elementary ideas leads to a multi-level algorithm which applies the forward map (3.1) for general surfaces at a total cost of $O(N \log N)$ operations. The algorithm and a study of its computational cost are presented in Sections 3.6 and 3.7, respectively.

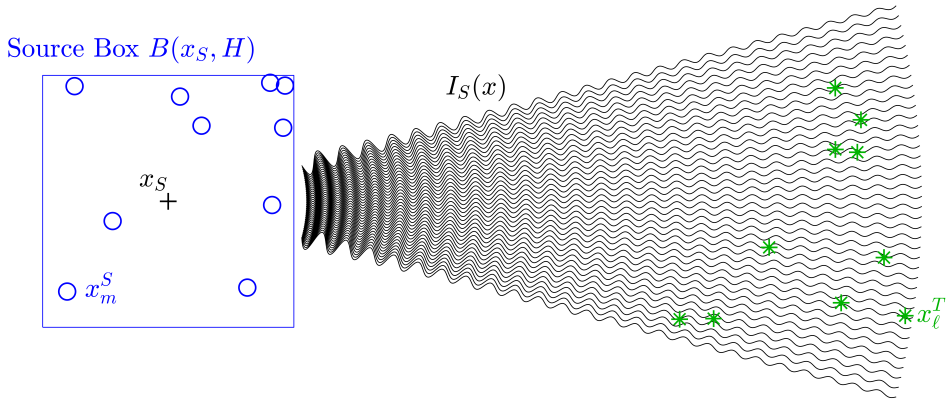


Figure 3.1: Two-dimensional illustration of a source box B_S containing multiple source points $x' \in \Gamma_N$ (blue circles) and certain illustrative target points (green stars). The black wavy lines represent the field I_S generated by the point sources in the box B_S.

3.2 Analyticity

In order to analyze the above introduced *analytic factor*, certain notations and conventions are introduced. On one hand, for notational simplicity, but without loss of generality, throughout the remainder of this section it is assumed that the factorization is centered at the origin, i.e., $x_S = 0$; the extension to the general $x_S \neq 0$ case is, of course, straightforward due to the translation invariance of the Green function under consideration. Incorporating the convention $x_S = 0$, then, for $0 < \eta < 1$, the following sets denoting the *analyticity domain* of the analytic factor are considered in the analysis of the factorization.

Definition 5 (Analyticity domain)**.** *Let $B = B(0, H)$ denote an origin-centered box of side $H > 0$, as per Definition 4. Let $\eta > 0$. Then, the* analyticity domain A_η^H *of the analytic factor of a field emitted by the box B is defined as the subset of the set*

$$A_\eta := \{(x, x') \in \mathbb{R}^3 \times \mathbb{R}^3 \ : \ |x'| \leq \eta |x|\}$$

given by

$$A_\eta^H := A_\eta \cap \left(\mathbb{R}^3 \times B\right). \tag{3.6}$$

Clearly, A_η^H is the subset of pairs in A_η such that x' is restricted to a particular box $B(0, H)$. Theorem 4 below implies that, on the basis of an appropriate change of variables which adequately accounts for the analyticity of the analytic factor g_S up to and including infinity, this factor can be accurately evaluated for $(x, x') \in A_\eta^H$

by means of a straightforward interpolation rule based on an interpolation mesh in spherical coordinates, which is finite and sparse along the radial direction.

As indicated above, the analytic properties of the factor g_S play a pivotal role in the proposed algorithm. Under the $x_S = 0$ convention established above, the factors in equation (3.4) become

$$G(x,0) = \frac{e^{\imath\kappa|x|}}{4\pi|x|} \quad \text{and} \quad g_S(x,x') = \frac{|x|}{|x-x'|} e^{\imath\kappa(|x-x'|-|x|)}. \tag{3.7}$$

In order to analyze the properties of the factor g_S, we introduce the spherical coordinate parametrization

$$\tilde{\mathbf{x}}(r,\theta,\varphi) \coloneqq \begin{pmatrix} r\sin\theta\cos\varphi \\ r\sin\theta\sin\varphi \\ r\cos\theta \end{pmatrix}, \qquad 0 \le r < \infty,\ 0 \le \theta \le \pi,\ 0 \le \varphi < 2\pi, \tag{3.8}$$

and note that (3.7) may be re-expressed in the form

$$g_S(x,x') = \frac{1}{4\pi\left|\frac{x}{r} - \frac{x'}{r}\right|} \exp\left(\imath\kappa r\left(\left|\frac{x}{r} - \frac{x'}{r}\right| - 1\right)\right). \tag{3.9}$$

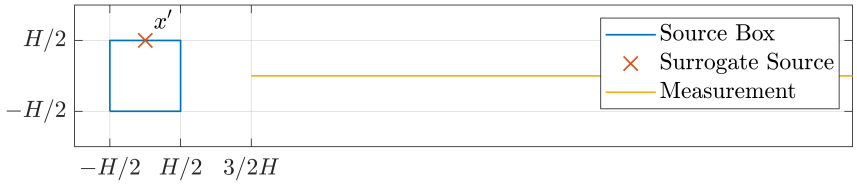
The effectiveness of the proposed factorization is illustrated in Figures 3.2a, 3.2b, and 3.2c, where the oscillatory character of the analytic factor g_S and the Green function (1.8) without factorization are compared, as a function of r, for several wavenumbers κ. The slowly-oscillatory character of the factor g_S, even for acoustically large source boxes $B(x_S, H)$ as large as twenty wavelengths $\lambda = 2\pi/\kappa$ ($H = 20\lambda$) and starting as close as just $3H/2$ away from the center of the source box, is clearly visible in Figure 3.2c; much faster oscillations are observed in Figure 3.2b, even for source boxes as small as two wavelengths in size ($H = 2\lambda$). Only the real part is depicted in Figures 3.2a, 3.2b, and 3.2c, but, clearly, the imaginary part displays the same behavior.

In addition to the factorization (3.5), the proposed strategy relies on the use of the singularity resolving change of variables

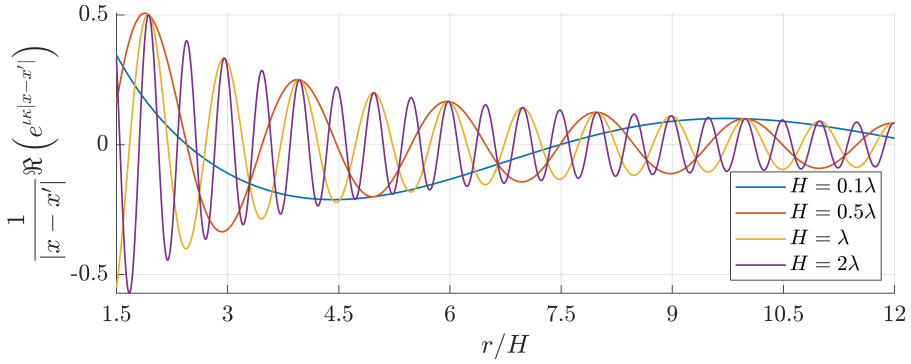
$$s \coloneqq \frac{h}{r}, \qquad \mathbf{x}(s,\theta,\varphi) \coloneqq \tilde{\mathbf{x}}(r,\theta,\varphi), \tag{3.10}$$

where, once again, $r = |x|$ denotes the radius in spherical coordinates and where h denotes the radius of the source box, as in Definition 4. Using these notations, equation (3.9) may be re-expressed in the form

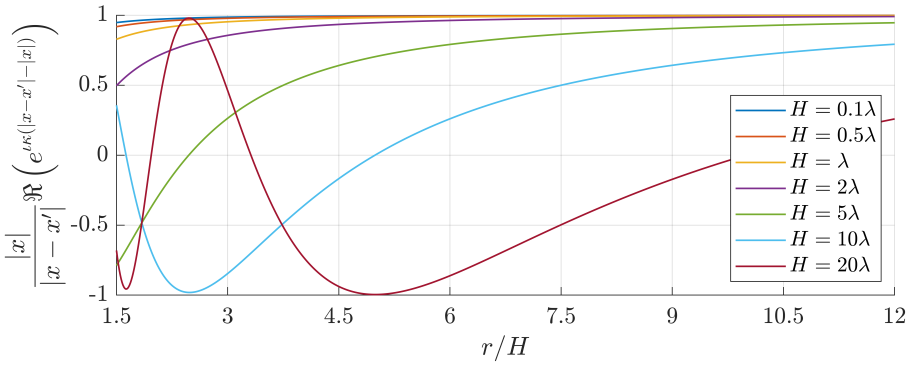
$$g_S(x,x') = \frac{1}{4\pi\left|\frac{x}{r} - \frac{x'}{h}s\right|} \exp\left(\imath\kappa r\left(\left|\frac{x}{r} - \frac{x'}{h}s\right| - 1\right)\right). \tag{3.11}$$



(a) Test setup. The Surrogate Source position x' gives rise to the fastest possible oscillations along the Measurement line, among all possible source positions within the Source Box.



(b) Real part of the Green function G in equation (1.8) (without factorization), along the Measurement line depicted in Figure 3.2a, for boxes of various acoustic sizes H.



(c) Real part of the analytic factor g_S (equation (3.7)) along the Measurement line depicted in Figure 3.2a, for boxes of various acoustic sizes H.

Figure 3.2: Surrogate Source factorization test, set up as illustrated in Figure 3.2a. Figure 3.2c shows that the analytic factor g_S oscillates much more slowly, even for $H = 20\lambda$, than the unfactored Green function does for the much smaller values of H considered in Figure 3.2b.

Remark 4. *While the source point x and its norm r depend on s, the quantity x/r is independent of r and therefore also of s.*

The introduction of the variable s gives rise to several algorithmic advantages, all of which stem from the analyticity properties of the function g_S—as presented in Lemma 1 below and Theorem 4 in Section 3.3. Briefly, these results establish that, for any fixed values $H > 0$ and η satisfying $0 < \eta < 1$, the function g_S is analytic for $(x, x') \in A_\eta^H$, with x-derivatives that are bounded up to and including $|x| = \infty$. As a result (as shown in Section 3.3) the s change of variables translates the problem of interpolation of g_S over an infinite r interval into a problem of interpolation of an analytic function of the variable s over a compact interval in the s variable. The relevant H-dependent analyticity domains for the function g_S for each fixed value of H are described in the following lemma.

Lemma 1. *Let $x' \in B(x_S, H)$ and let $x_0 = \tilde{\mathbf{x}}(r_0, \theta_0, \varphi_0) = \mathbf{x}(s_0, \theta_0, \varphi_0)$ ($s_0 = h/r_0$) be such that $(x_0, x') \in A_\eta^H$. Then g_S is an analytic function of x around x_0 and also an analytic function of (s, θ, φ) around $(s_0, \theta_0, \varphi_0)$. Further, the function g_S is an analytic function of (s, θ, φ) (resp. (r, θ, φ)) for $0 \le \theta \le \pi$, $0 \le \varphi < 2\pi$, and for s in a neighborhood of $s_0 = 0$ (resp. for r in a neighborhood of $r_0 = \infty$, including $r = r_0 = \infty$).*

Proof. The claimed analyticity of the function g_S around $x_0 = \mathbf{x}(s_0, \theta_0, \varphi_0)$ (and, thus, the analyticity of g_S around $(s_0, \theta_0, \varphi_0)$) is immediate since, under the assumed hypothesis, the quantity

$$\left| \frac{x}{r} - \frac{x'}{h} s \right|, \tag{3.12}$$

does not vanish in a neighborhood of $x = x_0$. Analyticity around $s_0 = 0$ ($r_0 = \infty$) follows similarly since the quantity (3.12) does not vanish around $s = s_0 = 0$. □

Corollary 1. *Let $H > 0$ be given. Then for all $x' \in B(x_S, H)$, the function $g_S(\mathbf{x}(s, \theta, \varphi), x')$ is an analytic function of (s, θ, φ) for $0 \le s < 1$, $0 \le \theta \le \pi$ and $0 \le \varphi < 2\pi$.*

Proof. Take $\eta \in (0, 1)$. Then, for $0 \le s \le \eta$, we have $(\mathbf{x}(s, \theta, \varphi), x') \in A_\eta^H$. The analyticity for $0 \le s \le \eta$ follows from Lemma 1, and since $\eta \in (0, 1)$ is arbitrary, the lemma follows. □

For a given $x' \in \mathbb{R}^3$, Corollary 1 reduces the problem of interpolation of the function $g_S(x, x')$ in the x variable to a problem of interpolation of a re-parametrized form of the function g_S over a bounded domain—provided that $(x, x') \in A_\eta^H$, or, in other words, provided that x is separated from x' by a factor of at least η, for some $\eta < 1$. In the IFGF algorithm presented in Section 3.6, side-H boxes $B(x_S, H)$ containing sources x' are considered, with target points x at a distance no less than H away from $B(x_S, H)$. Clearly, a point (x, x') in such a configuration necessarily belongs to A_η^H with $\eta = \sqrt{3}/3$. Importantly, as demonstrated in the following section, the interpolation quality of the algorithm does not degrade as source boxes of increasingly large side H are used, as is done in the proposed multi-level IFGF algorithm (with a single box size at each level), leading to a computing cost per level which is independent of the level box size H.

3.3 Interpolation procedure

On the basis of the discussion presented in the previous Section 3.2, the present section concerns the problem of interpolation of the analytic factor g_S in the variables (s, θ, φ). For efficiency, piece-wise Chebyshev interpolation (see Section 2.2) in each one of these variables is used, over interpolation intervals of respective lengths Δ_s, Δ_θ and Δ_φ, where, for a certain positive integer n_C, angular coordinate intervals of size

$$\Delta_\theta = \Delta_\varphi = \frac{\pi}{n_C},$$

are utilized. Defining

$$\theta_k \coloneqq k\Delta_\theta, \quad (k = 0, \ldots, n_C - 1) \quad \text{and} \quad \varphi_\ell \coloneqq \ell\Delta_\varphi, \quad (\ell = 0, \ldots, 2n_C - 1),$$

as well as

$$\begin{aligned} E_j^\varphi &\coloneqq [\varphi_{j-1}, \varphi_j) \quad \text{and} \\ E_{i,j}^\theta &\coloneqq \begin{cases} [\theta_{n_C-1}, \pi] & \text{for} \quad i = n_C,\ j = 2n_C \\ (0, \Delta_\theta) & \text{for} \quad i = 1,\ j > 1 \\ [\theta_{i-1}, \theta_i) & \text{otherwise,} \end{cases} \end{aligned} \tag{3.13}$$

we thus obtain the mutually disjoint *interpolation cones*

$$\begin{aligned} \tilde{C}_{i,j} \coloneqq \left\{ x = \tilde{\mathbf{x}}(r, \theta, \varphi) \ :\ r \in (0, \infty),\ \theta \in E_{i,j}^\theta,\ \varphi \in E_j^\varphi \right\}, \\ (i = 1, \ldots, n_C, j = 1, \ldots, 2n_C), \end{aligned} \tag{3.14}$$

centered at $x_S = (0, 0, 0)^T$.

Remark 5. *Definition* (3.14) *ensures that the cone segments cover all of* $\mathbb{R}^3$ *(except for the origin) and are pairwise disjoint, i.e.,*

$$\bigcup_{\substack{1=1,\dots,n_C \\ j=1,\dots,2n_C}} \tilde{C}_{i,j} = \mathbb{R}^3 \setminus \{0\} \qquad \textit{and}$$

$$\tilde{C}_{i,j} \cap \tilde{C}_{k,l} = \emptyset \quad \textit{for} \quad (i,j) \neq (k,l).$$

The proposed interpolation strategy additionally relies on a number $n_s \in \mathbb{N}$ of disjoint radial interpolation intervals E_k^s, $k = 1, \dots, n_s$, of size $\Delta_s = \eta/n_s$, within the IFGF s-variable radial interpolation domain $[0, \eta]$ (with $\eta = \sqrt{3}/3$, see Section 3.2). Thus, in all, the approach utilizes an overall number $N_C := n_s \times n_C \times 2n_C$ of interpolation domains

$$E_\gamma := E_{\gamma_1}^s \times E_{\gamma_2}^\theta \times E_{\gamma_3}^\varphi, \tag{3.15}$$

which we call *cone domains*, with $\gamma = (\gamma_1, \gamma_2, \gamma_3) \in \{1, \dots, n_s\} \times \{1, \dots, n_C\} \times \{1, \dots, 2n_C\}$. Under the parametrization $\mathbf{x}$ in equation (3.10), the cone domains yield the *cone segment* sets

$$C_\gamma := \{x = \mathbf{x}(s, \theta, \varphi) \,:\, (s, \theta, \varphi) \in E_\gamma\}. \tag{3.16}$$

Remark 6. *By definition, the cone segments are mutually disjoint.*

A two-dimensional illustration of the cone domains and associated cone segments is provided in Figure 3.3.

The desired interpolation strategy then relies on the use of a fixed number $P = P_{\text{ang}}^2 P_s$ of interpolation points for each cone segment C_γ, where P_{ang} (resp. P_s) denotes the number of Chebyshev interpolation points per interval used for each angular variable (resp. for the radial variable s). For each cone segment, the proposed interpolation approach proceeds by breaking up the problem into a sequence of one-dimensional Chebyshev interpolation problems of accuracy orders P_s and P_{ang}, as described in Section 2.2, along each one of the three coordinate directions s, θ and φ. This spherical Chebyshev interpolation procedure is described in what follows, and an associated error estimate is presented which is then used to guide the selection of cone segment sizes.

The desired error estimate for the nested Chebyshev interpolation procedure within a cone segment (3.16) (or, more precisely, within the cone domains (3.15)) is provided by the following theorem.

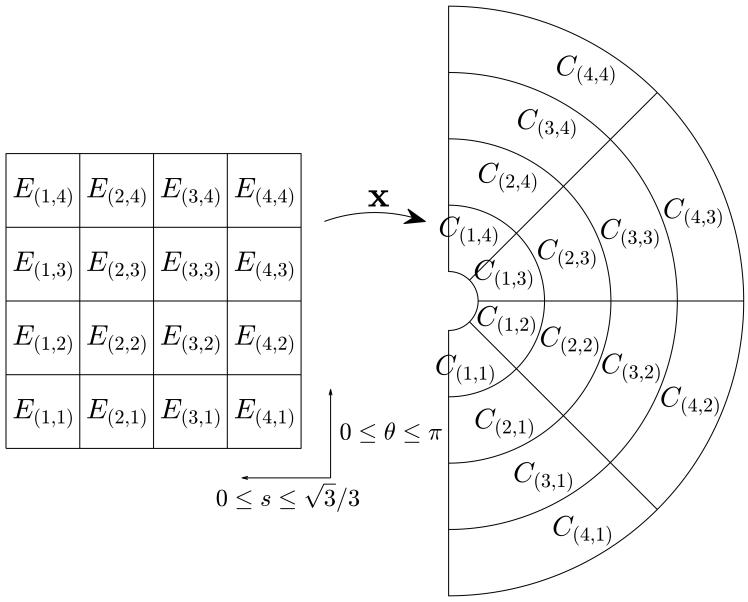


Figure 3.3: Schematic two-dimensional illustration of a set of cone domains E_γ, together with the associated cone segments C_γ that result under the parametrization (3.10). For the sake of simplicity, the illustration shows constant cone-segment radial sizes (in the r variable), but the actual radial sizes are constant in the s variable (equation (3.10)), instead. Thus, increasingly large real-space cone segments are used as the distance of the interpolation cone segments to the origin grows.

Theorem 3. *Let $I^s_{P_s}$, $I^\theta_{P_{ang}}$, and $I^\varphi_{P_{ang}}$ denote the Chebyshev interpolation operators of accuracy orders P_s in the variable s and P_{ang} in the angular variables θ and φ, over intervals E^s, E^θ, and E^φ of lengths Δ_s, Δ_θ, and Δ_φ in the variables s, θ, and φ, respectively. Then, for each arbitrary but fixed point $x' \in \mathbb{R}^3$, the error arising from nested interpolation of the function $g_S(\mathbf{x}(s,\theta,\varphi), x')$ (cf. equation* (3.10)*) in the variables (s,θ,φ) satisfies the estimate*

$$|g_S(\mathbf{x}(s,\theta,\varphi),x') - I^\varphi_{P_{ang}} I^\theta_{P_{ang}} I^s_{P_s} g_S(\mathbf{x}(s,\theta,\varphi),x')| \le$$
$$C\left[(\Delta_s)^{P_s}\left\|\frac{\partial^{P_s} g_S}{\partial s^{P_s}}\right\|_\infty + (\Delta_\theta)^{P_{ang}}\left\|\frac{\partial^{P_{ang}} g_S}{\partial \theta^{P_{ang}}}\right\|_\infty + (\Delta_\varphi)^{P_{ang}}\left\|\frac{\partial^{P_{ang}} g_S}{\partial \varphi^{P_{ang}}}\right\|_\infty\right], \quad (3.17)$$

for some constant C depending only on P_s and P_{ang}, where the supremum-norm expressions are shorthands for the supremum norm defined by

$$\left\|\frac{\partial^n g_S}{\partial \xi^n}\right\|_\infty := \sup_{\substack{\tilde{s}\in E^s \\ \tilde{\theta}\in E^\theta \\ \tilde{\varphi}\in E^\varphi}} \left|\frac{\partial^n g_S}{\partial \xi^n}(\mathbf{x}(\tilde{s},\tilde{\theta},\tilde{\varphi}),x')\right|$$

for $\xi = s$, θ, or φ.

Proof. The proof is only presented for a double-nested interpolation procedure; the extension to the triple-nested method is entirely analogous. Suppressing, for readability, the explicit functional dependence on the variables x and x', use of the triangle inequality and the error estimate (2.4) yields

$$\begin{aligned}|g_S - I^{\theta}_{P_{\text{ang}}} I^{s}_{P_s} g_S| &\leq |f - I^{s}_{P_s} g_S| + |I^{\theta}_{P_{\text{ang}}} I^{s}_{P_s} g_S - I^{s}_{P_s} g_S| \\ &\leq C_1 (\Delta_s)^{P_s} \left\| \frac{\partial^{P_s} g_S}{\partial s^{P_s}} \right\|_{\infty} + C_2 (\Delta_\theta)^{P_{\text{ang}}} \left\| \frac{\partial^{P_{\text{ang}}} I^{s}_{P_s} g_S}{\partial \theta^{P_{\text{ang}}}} \right\|_{\infty},\end{aligned}$$

where C_1 and C_2 are constants depending on P_s and P_{ang}, respectively. In order to estimate the second term on the right-hand side in terms of derivatives of g_S, we utilize equation (2.3) in the shifted arguments corresponding to the s-interpolation interval (a, b):

$$I^{s}_{P_s} g_S = \sum_{i=0}^{P_s - 1} a^{s}_{i}(\theta) T_i \left(2 \frac{s-a}{b-a} - 1 \right), \quad (b = a + \Delta_s).$$

Differentiation with respect to θ and use of the relations (2.1) and (2.3) then yield

$$\left\| \frac{\partial^{P_{\text{ang}}} I^{s}_{P_s} g_S}{\partial \theta^{P_{\text{ang}}}} \right\|_{\infty} \leq P_s \max_{i=1,\ldots,P_s-1} \left\| \frac{\partial^{P_{\text{ang}}} a^{s}_{i}}{\partial \theta^{P_{\text{ang}}}} \right\|_{\infty} \leq C_3 \left\| \frac{\partial^{P_{\text{ang}}} g_S}{\partial \theta^{P_{\text{ang}}}} \right\|_{\infty},$$

as it may be checked, for a certain constant C_3 depending on P_s, by employing the triangle inequality and the L^∞ bound $\|T_i\|_\infty \leq 1$ ($i \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$). The more general error estimate (3.17) follows by a direct extension of this argument to the triple-nested case, and the proof is thus complete. □

The analysis presented in what follows, including Lemmas 2 through 4 and Theorem 4, yields bounds for the partial derivatives in (3.17) in terms of the acoustic size κH of the source box $B(x_S, H)$. Subsequently, these bounds are used, together with the error estimate (3.17), to determine suitable choices of the cone domain sizes Δ_s, Δ_θ, and Δ_φ, ensuring that the errors resulting from the triple-nested interpolation process lie below a prescribed error tolerance. Leading to Theorem 4, the next three lemmas provide estimates, in terms of the box size H, of the n-th order derivatives ($n \in \mathbb{N}$) of certain functions related to $g_S(\mathbf{x}(s, \theta, \varphi), x')$, with respect to each one of the variables s, θ, and φ and every $x' \in B(x_S, H)$.

Lemma 2. *Under the change of variables $x = \mathbf{x}(s, \theta, \varphi)$ in* (3.10)*, for all $n \in \mathbb{N}$ and for either $\xi = \theta$ or $\xi = \varphi$, we have*

$$\frac{\partial^n}{\partial \xi^n} |x - x'| = \sum \frac{c(m_1, \ldots, m_n)}{|x - x'|^{2k-1}} \prod_{j=1}^{n} \left\langle \frac{\partial^j x}{\partial \xi^j}, x' \right\rangle^{m_j},$$

where the outer sum is taken over all n-tuples $(m_1, \ldots, m_n) \in \mathbb{N}_0^n$ *such that*

$$\sum_{j=1}^{n} j m_j = n,$$

where $k := \sum_{i=1}^{n} m_i$*, where* $c(m_1, \ldots, m_n) \in \mathbb{R}$ *denote constants independent of* x*,* x' *and* ξ*, and where* $\langle \cdot, \cdot \rangle$ *denotes the Euclidean inner product on* $\mathbb{R}^3$*.*

Proof. The proof follows from Faà di Bruno's formula [69] applied to $f(g(x)) = |x - x'|$, where $f(x) = \sqrt{x}$ and $g(x) = \langle x, x \rangle - 2\langle x, x' \rangle + \langle x', x' \rangle$. Indeed, noting that

$$\frac{d^k f(x)}{dx^k} = c_1(k) \frac{1}{f(x)^{2k-1}},$$

for some constant $c_1(k)$, and that, since $\langle \frac{\partial x}{\partial \xi}, x \rangle = 0$ for $\xi = \theta$ and $\xi = \varphi$,

$$\frac{\partial^i g(x(\xi))}{d\xi^i} = c_2(i) \left\langle \frac{\partial^i x}{\partial \xi^i}, x' \right\rangle,$$

for some constant $c_2(i)$, an application of Faà di Bruno's formula directly yields the desired result. □

Lemma 3. *Let* $H > 0$ *and* $\eta \in (0, 1)$ *be given. Then, under the change of variables* $x = \mathbf{x}(s, \theta, \varphi)$ *in* (3.10)*, the exponent in the right-hand exponential in* (3.7) *satisfies*

$$\frac{\partial^n}{\partial \xi^n} \left(|x - x'| - |x| \right) \leq C(\eta, n) H,$$

for all $(x, x') \in A_\eta^H$*, for all* $n \in \mathbb{N}_0$*, and for* $\xi = s$*,* $\xi = \theta$ *and* $\xi = \varphi$*, where* $C(\eta, n)$ *is a certain real constant that depends on* η *and* n*, but which is independent of* H*.*

Proof. Expressing the exponent in (3.7) in terms of s yields

$$|x - x'| - |x| = \frac{h}{s} \left(\left| \frac{x}{r} - \frac{x'}{h} s \right| - 1 \right) =: h g(s), \tag{3.18}$$

where our standing assumption $x_S = 0$ and notation $|x| = r$ have been used (so that, in particular, x/r is independent of r and therefore also independent of s), and where the angular dependence of the function g has been suppressed. Clearly, $g(s)$ is an analytic function of s for $s \in \left[0, h/|x'|\right)$ and, thus, since $\eta < 1$, for s in the compact interval $\left[0, \eta \cdot h/|x'|\right]$. It follows that g and each one of its derivatives with respect to s is uniformly bounded for all $s \in \left[0, \eta \cdot h/|x'|\right]$ and (as shown by a simple re-examination of the discussion above) for all H and for

all values of x/r and x'/h under consideration. Since at the point (x, x') we have $s = h/|x| = |x'|/|x|| \cdot h/|x' \leq \eta \cdot h/|x'|$, using (3.2) once again, the desired $\xi = s$ estimate

$$\frac{\partial^n}{\partial s^n}\left(hg(s)\right) \leq C(\eta, n)H,$$

follows, for some constant $C(\eta, n)$.

Turning to the angular variables, we only consider the case $\xi = \theta$; the case $\xi = \varphi$ can be treated similarly. Using Lemma 2 for $\xi = \theta$, the Cauchy-Schwarz inequality and the assumption $(x, x') \in A_\eta^H$, we obtain

$$\begin{aligned}
\left|\frac{\partial^n \left(|x - x'| - |x|\right)}{\partial \theta^n}\right| &= \left|\frac{\partial^n \left(|x - x'|\right)}{\partial \theta^n}\right| \\
&= \left|\sum \frac{c(m_1, \ldots, m_n)}{|x - x'|^{2k-1}} \prod_{j=1}^{n} \left\langle \frac{\partial^j x}{\partial \xi^j}, x' \right\rangle^{m_j}\right| \\
&\leq \sum \frac{|c(m_1, \ldots, m_n)|}{|x - x'|^{2k-1}} \prod_{j=1}^{n} \left|\frac{\partial^j x}{\partial \xi^j}\right|^{m_j} |x'|^{m_j} \\
&\leq \sum_{k=1}^{n} \hat{C}(\eta, n) \frac{1}{r^{2k-1}} r^k \, |x'|^k \\
&\leq \tilde{C}(\eta, n) \, |x'| \\
&\leq C(\eta, n)H,
\end{aligned}$$

where the same notation as in Lemma 2 was used. The constant $C(\eta, n)$ has been suitably adjusted. The proof is now complete. □

Lemma 4. *Let $H > 0$ and $\eta \in (0, 1)$ be given. Then, under the change of variables $x = \mathbf{x}(s, \theta, \varphi)$ in* (3.10)*, for all $(x, x') \in A_\eta^H$, for all $n \in \mathbb{N}_0$, and for $\xi = s$, $\xi = \theta$ and $\xi = \varphi$, we have*

$$\left|\frac{\partial^n}{\partial \xi^n} e^{\imath\kappa(|x-x'|-|x|)}\right| \leq \tilde{M}(\eta, n) \left(\kappa H\right)^n,$$

where $\tilde{M}(\eta, n)$ is a certain real constant that depends on η and n, but which is independent of H.

Proof. Using Faà di Bruno's formula [69] yields

$$\frac{\partial^n}{\partial \xi^n} e^{\imath\kappa(|x-x'|-|x|)} = \sum c(m_1, \ldots, m_n) e^{\imath\kappa(|x-x'|-|x|)} \prod_{j=1}^{n} \left(\imath\kappa \frac{\partial^j \left(|x - x'| - |x|\right)}{\partial \xi^j}\right)^{m_j},$$

where the sum is taken over all n-tuples $(m_1, \ldots, m_n) \in \mathbb{N}_0^n$ such that

$$\sum_{j=1}^{n} j m_j = n,$$

and where $c(m_1, \ldots, m_n)$ are certain constants which depend on $m_1, \ldots, m_n$. Using the triangle inequality and Lemma 3 then completes the proof. □

The desired bounds on derivatives of the function g_S are presented in the following theorem.

Theorem 4. *Let $H > 0$ and $\eta \in (0, 1)$ be given. Then, under the change of variables $x = \mathbf{x}(s, \theta, \varphi)$ in (3.10), for all $(x, x') \in A_\eta^H$, for all $n \in \mathbb{N}_0$, and for $\xi = s$, $\xi = \theta$ and $\xi = \varphi$, we have*

$$\left| \frac{\partial^n g_S}{\partial \xi^n} \right| \leq M(\eta, n) \max\{(\kappa H)^n, 1\},$$

where $M(\eta, n)$ is a certain real constant that depends on η and n, but which is independent of H.

Proof. The quotient on the right-hand side of (3.7) may be re-expressed in the form

$$\frac{|x|}{|x - x'|} = \frac{1}{\left| \frac{x}{r} - \frac{x'}{h} s \right|}, \tag{3.19}$$

where x/r is independent of r and therefore also independent of s. An analyticity argument similar to the one used in the proof of Lemma 3 shows that this quotient, as well as each one of its derivatives with respect to s, is uniformly bounded for s throughout the interval $\left[0, \eta \cdot h/|x'|\right]$, for all $H > 0$, and for all relevant values of x/r and x'/h.

In order to obtain the desired estimates, we now utilize Leibniz' differentiation rule, which yields

$$\begin{aligned} \left| \frac{\partial^n g_S(x, x')}{\partial \xi^n} \right| &= \left| \sum_{i=0}^{n} \binom{n}{i} \frac{\partial^{n-i}}{\partial \xi^{n-i}} \left(\frac{|x|}{|x - x'|} \right) \frac{\partial^i}{\partial \xi^i} \left(e^{\imath \kappa (|x - x'| - |x|)} \right) \right| \\ &\leq C(\eta, n) \sum_{i=0}^{n} \frac{\partial^i}{\partial \xi^i} e^{\imath \kappa (|x - x'| - |x|)}, \end{aligned}$$

for some constant $C(\eta, n)$ that depends on η and n, but which is independent of H. Applying Lemma 4 and suitably adjusting constants, the result follows. □

In view of the bound (3.17), Theorem 4 shows that the interpolation error remains uniformly small provided that the interpolation interval sizes Δ_s, Δ_θ, and Δ_φ are held constant for $\kappa H < 1$ and are taken to decrease like $O(1/(\kappa H))$ as the box sizes κH grow when $\kappa H \geq 1$.

This observation motivates the main strategy in the IFGF algorithm. As the algorithm progresses from one level to the next, the box sizes are doubled, from H to $2H$, and the cone segment interpolation interval lengths Δ_s, Δ_θ, and Δ_φ are either kept constant or decreased by a factor of $1/2$ (depending on whether $\kappa H < 1$ or $\kappa H \geq 1$, respectively)—while the interpolation error, at a fixed number of degrees of freedom per cone segment, remains uniformly bounded. The resulting hierarchy of boxes and cone segments is embodied in two different but inter-related hierarchical structures: the box octree and a hierarchy of cone segments. In the box octree, each box contains eight equi-sized child boxes. In the cone segment hierarchy, similarly, each cone segment (spanning certain angular and radial intervals) spawns *up to* eight child segments. The $\kappa H \to \infty$ limit then is approached as the box tree structure is traversed from children to parents and the accompanying cone segment structure is traversed from parents to children. This hierarchical strategy and associated structures are described in detail in the following Sections 3.4 and 3.5.

The properties of the proposed interpolation strategy, as implied by Theorem 4 (in presence of Theorem 3), are illustrated by the blue dash-dot error curves presented on the upper plot in Figure 3.4. For reference, this figure also includes error curves corresponding to various related interpolation strategies, as described below. In this demonstration, the field generated by one thousand sources randomly placed within a source box $B(x_S, H)$ of acoustic size κH is interpolated to one thousand points randomly placed within a cone segment of interval lengths Δ_s, Δ_θ, and Δ_φ proportional to $\min\{1, 1/(\kappa H)\}$—which, in accordance with Theorems 3 and 4, ensures essentially constant errors. All curves in Figure 3.4 report errors relative to the maximum absolute value of the exact one-thousand source field value within the relevant cone segment. The target cone segment used is symmetrically located around the x axis, and it lies within the r range $3H/2 \leq r \leq 3H/2 + \Delta_r$, for the value

$$\Delta_r = \frac{9H\Delta_s}{2\sqrt{3}(1 - \sqrt{3}\Delta_s)}$$

corresponding to a given value of Δ_s. It is useful to note that, depending on the values of θ and φ, the distance from the closest possible singularity position to the left endpoint of the interpolation interval could vary from a distance of H to a distance of

$\frac{\sqrt{3}(\sqrt{3}-1)}{2}H \approx 0.634H$; cf. Figure 3.2a. In all cases the interpolations were produced by means of Chebyshev expansions of degree two and four (with numerical accuracy of orders $P_s = 3$ and $P_{\text{ang}} = 5$) in the radial and angular directions, respectively. The (κH-dependent) radial interpolation interval sizes Δ_s were selected as follows: starting with the value $\Delta_s = \sqrt{3}/3$ for $\kappa H = 10^{-1}$, Δ_s was varied proportionally to $1/(\kappa H)$ (resp. $\min\{1, 1/(\kappa H)\}$) in the top (resp. bottom) plot as κH increases. (Note that the value $\Delta_s = \sqrt{3}/3$, which corresponds to the infinite-length interval going from $r = 3H/2$ to $r = \infty$, is the maximum possible value of Δ_s along an interval on the x axis whose distance to the source box is not smaller than one box-size H. In particular, the errors presented for $\kappa H = 10^{-1}$ correspond to interpolation, using a finite number of intervals, along the entire rightward x semi-axis starting at $x = 3H/2$.) The corresponding angular interpolation lengths $\Delta_\theta = \Delta_\varphi$ were set to $\pi/4$ for the initial $\kappa H = 10^{-1}$ value, and they were then varied like the radial interval proportionally to $1/(\kappa H)$ (resp. $\min\{1, 1/(\kappa H)\}$) in the top (resp. bottom) plot.

As indicated above, the figure shows various interpolation results, including results for interpolation in the variable r without factorization (thus interpolating the Green function (1.8) directly), with exponential factorization (factoring only $\exp(\iota\kappa|x|)$ and interpolating $\exp(\iota\kappa(|x - x'| - |x|)/r)$, with exponential and denominator factorization (called full factorization, factoring the centered factor interpolating the analytic factor as in (3.7)), and, finally, for the interpolation in the s variable also under full factorization. It can be seen that the exponential factorization is beneficial for the interpolation strategy in the *high-frequency regime* (κH large) while the factorization of the denominator and the use of the s change of variables is beneficial for the interpolation in the *low-frequency regime* (κH small). Importantly, the bottom plot in Figure 3.4 confirms that, as predicted by theory, constant interval sizes in all three variables (s, θ, φ) suffice to ensure a constant error in the low-frequency regime. Thus, the overall strategy leads to constant errors for $0 \leq \kappa H < \infty$. Figure 3.4 also emphasizes the significance of the factorization of the denominator, i.e., the removal of the singularity, without which interpolation with significant accuracy would be only achievable using a prohibitively large number of interpolation points. And, it also shows that the change of variables from the r variable to the s variable leads to a selection of interpolation points leading to improved accuracies for small values of κH.

Theorem 4 also holds for the special $\kappa = 0$ case of the Green function for the Laplace equation. In view of its independent importance, the result is presented,

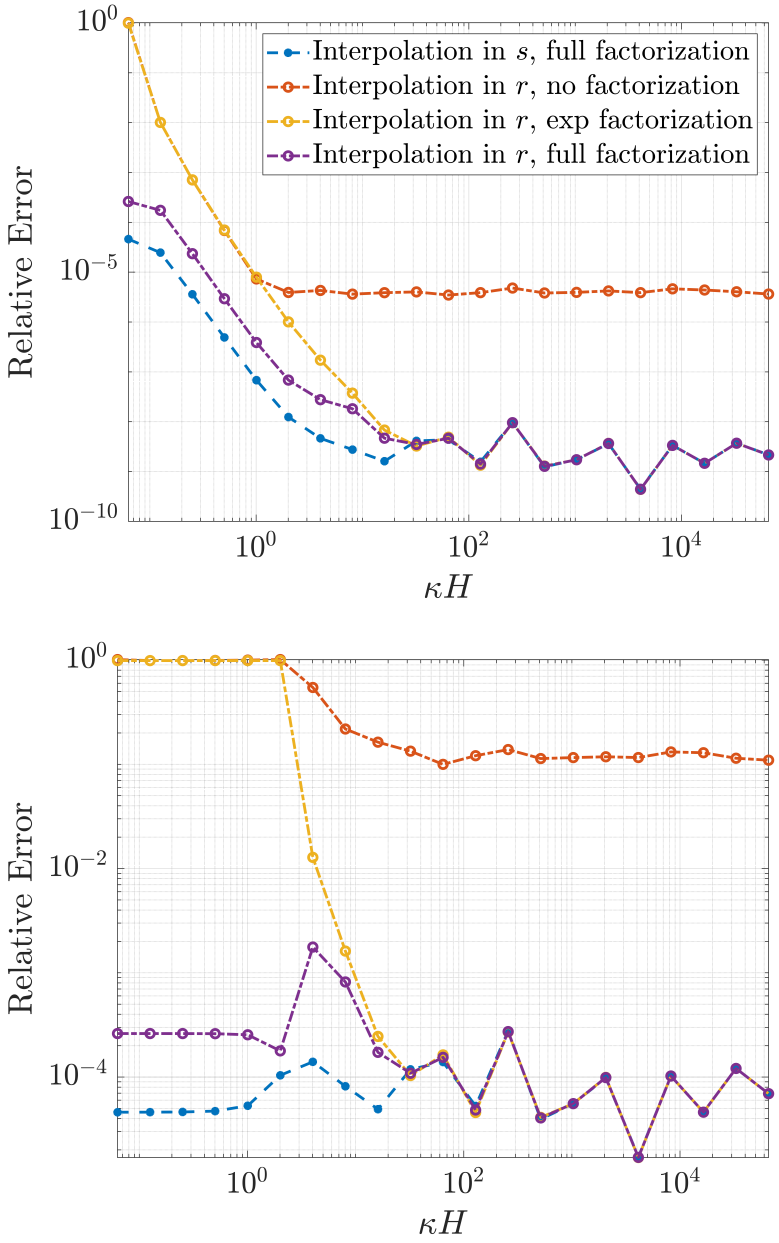


Figure 3.4: Numerical investigation of Theorem 4 showing the overall interpolation error for various Green function factorizations and two different cone segment refinement strategies. Top graph: Errors resulting from use of interpolation intervals of sizes Δ_s, Δ_θ and Δ_φ proportional to $1/(\kappa H)$—which suffices to capture the oscillatory behavior for large κH, but which under-resolves the singularity that arises for small κH values, for which the Green function singular point $x = x'$ is approached. Bottom graph: Errors resulting from use of interpolation interval sizes Δ_s, Δ_θ and Δ_φ that remain constant for small κH (< 1), and which decrease like $1/(\kappa H)$ for large κH (> 1), resulting in essentially uniform accuracy for all box sizes provided the full IFGF factorization is used. Note that the combined use of full factorization and interpolation in the s variable, yields the best (essentially uniform) approximations.

in Corollary 2, explicitly for the Laplace case, without reference to the Helmholtz kernel.

Corollary 2. *Let $G^{\Delta}(x, x') = 1/|x - x'|$ denote the Green function of the three dimensional Laplace equation and let $g_S^{\Delta}(x, x') = |x|/|x - x'|$ denote the analytic kernel (cf. equations* (3.4) *and* (3.7) *with $\kappa = 0$). Additionally, let $H > 0$ and $\eta \in (0, 1)$ be given. Then, under the change of variables $x = \mathbf{x}(s, \theta, \varphi)$ in* (3.10)*, for all $(x, x') \in A_\eta^H$, for all $n \in \mathbb{N}_0$, and for $\xi = s$, $\xi = \theta$ and $\xi = \varphi$, we have*

$$\left| \frac{\partial^n g_S^{\Delta}}{\partial \xi^n} \right| \leq M(\eta, n), \tag{3.20}$$

where $M(\eta, n)$ is a certain real constant that depends on η and n, but which is independent of H.

Corollary 2 shows that an even simpler and more efficient strategy can be used for the selection of the cone segment sizes in the Laplace case. Indeed, in view of Theorem 3, the corollary tells us that (as illustrated in Table 5.8) a constant number of cone segments per box, independent of the box size H, suffices to maintain a fixed accuracy as the box size H grows (as is also the case for the Helmholtz equation for small values of κ). As discussed in Section 3.7 and numerically verified in Section 5.4, this reduction in complexity leads to significant additional efficiency for the Laplace case.

Noting that Theorem 4 implies, in particular, that the function g_S and all its partial derivatives with respect to the variable s are bounded as $s \to 0$, below in this section we compare the interpolation properties in the s and r variables, but this time in the case in which the source box is fixed and $s \to 0$ ($r \to \infty$). To do this we rely in part on an upper bound on the derivatives of g_S with respect to the variable r, which is presented in Corollary 3.

Corollary 3. *Let $H > 0$ and $\eta \in (0, 1)$ be given. Then, under the change of variables $x = \mathbf{x}(s, \theta, \varphi)$ in* (3.10) *and for all $(x, x') \in A_\eta^H$, for all $n \in \mathbb{N}_0$ we have*

$$\left| \frac{\partial^n g_S}{\partial r^n} \right| \leq C_r(n, \kappa, H) \frac{1}{r^n} \sum_{m \in I} \left(\frac{h}{r} \right)^m,$$

where I denotes a subset of $\{1, \ldots, n\}$ including 1.

Proof. Follows directly using Theorem 4 and applying Faà di Bruno's formula to the composition $g_S(s(r), \theta, \varphi)$. □

Theorem 3, Theorem 4, and Corollary 3 show that, for any fixed value κH of the acoustic source box size, the error arising from interpolation using n interpolation points in the s variable (resp. the r variable) behaves like $(\Delta_s)^n$ (resp. $(\Delta_r)^n/r^{n+1}$). Additionally, as is easily checked, the increments Δ_s and Δ_r are related by the identity

$$\Delta_r = \frac{r_0^2 \Delta_s}{h - r_0 \Delta_s}, \tag{3.21}$$

where h and r_0 denote the source box radius (3.2) and the left endpoint of a given interpolation interval $r_0 \leq r \leq r_0 + \Delta_r$, respectively. These results and estimates lead to several simple but important conclusions. On one hand, for a given box size κH, a partition of the s-interpolation interval $[0, \eta]$ on the basis of a finite number of equi-sized intervals of fixed size Δ_s (on each one of which s-interpolation is to be performed) provide a natural and essentially optimal methodology for interpolation of the uniformly analytic function g_S up to the order of accuracy desired. Secondly, such covering of the s interpolation domain $[0, \eta]$ by a finite number of intervals of size Δ_s is mapped, via equation (3.10), to a covering of a complete semi-axis in the r variable and, thus, one of the resulting r intervals must be infinitely large—leading to large interpolation errors in the r variable. Finally, values of Δ_r leading to constant interpolation error in the r variable necessarily requires use of infinitely many interpolation intervals and is therefore significantly less efficient than the proposed s interpolation approach.

Figure 3.5 displays interpolation errors for both the s- and r-interpolation strategies, for increasing values of the left endpoint r_0 and a constant source box one wavelength in side. The interval Δ_s is kept constant and Δ_r is taken per equation (3.21). The rightmost points in Figure 3.5 are close to the singular point $r_0 = h/\Delta_s$ of the right-hand side in (3.21). The advantages of the s-variable interpolation procedure are clearly demonstrated by this figure.

3.4 Box octree structure

The above presented factorization of the Green function and the resulting analyticity properties of the analytic factor in its analyticity domain (see Definition 5), together with the radial interpolation strategy presented in Section 3.3, give rise to the unique box-octree structure inherent in the IFGF method, as described in what follows. For a comprehensive introduction of this box-cone structure, the boxes and cone segments in the context of the IFGF method are introduced separately in the present section and the following Section 3.5, respectively. This separation of the presentation is

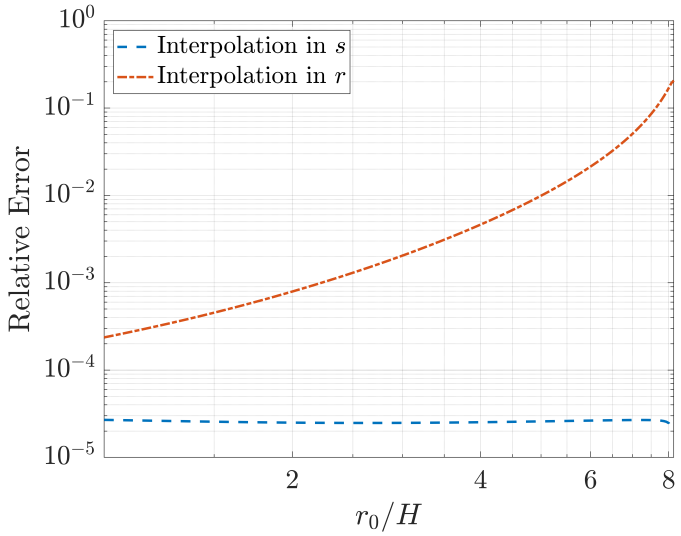


Figure 3.5: Comparison of the errors resulting from r- and s-based interpolation strategies for the problem of interpolation of the analytic factor g_S in the interval $[r_0, r_0 + \Delta_r)$, as a function of r_0. Clearly, the equi-spaced s discretization used is optimally suited for the interpolation problem at hand.

purely for the sake of readability; in practice, the boxes and cone segments form a cohesive whole data structure. For details regarding octree structures in general, we refer to, e.g., [20, Sec. 5.3] or [70].

Based on Definition 4, the box-octree structure underlying the IFGF method can be defined as a hierarchy of disjoint boxes, as shown in the following Definition 6.

Definition 6 (IFGF box-octree)**.** *Let $\Gamma_N = \{x_1, \ldots, x_N\} \subset \mathbb{R}^3$, $N > 1$, be a given set of distinct surface discretization points. A D-leveled ($D \in \mathbb{N}$)* octree structure $\mathcal{B} = \mathcal{B}(D, \Gamma_N)$ *for the partitioning of Γ_N is defined iteratively as follows. Let $x^1_{(1,1,1)} \in \mathbb{R}^3$ and $H_1 > 0$ be such that the box $B^1_{(1,1,1)} := B(x^1_{(1,1,1)}, H_1)$ (cf. Definition 4) satisfies*

$$\Gamma_N \subset B^1_{(1,1,1)},$$

where $H_1 = \min\{H > 0 \ : \ \Gamma_N \subset B(x, H), x \in \mathbb{R}^3\}$, and $x^1_{(1,1,1)}$ the corresponding point where this minimum is achieved.

For $d = 2, \ldots, D$, and $\mathbf{k} = (k_1, k_2, k_3) \in \{1, \ldots, 2^{d-1}\}^3 =: \mathcal{K}^d_B$ the boxes $B^d_{\mathbf{k}} := B(x^d_{\mathbf{k}}, H_d)$ are defined in terms of their sides H_d and their centers $x^d_{\mathbf{k}}$, which are, in turn, defined as follows.

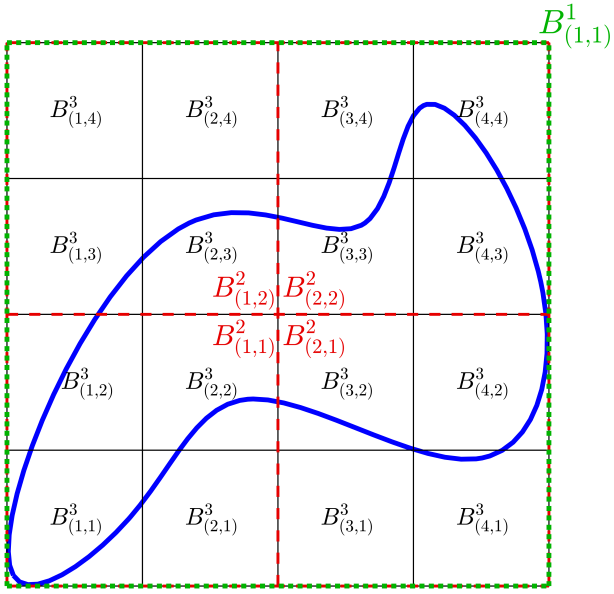


Figure 3.6: A scatterer, in blue, and three levels of a two-dimensional analog of the associated box tree, with the highest level box $B^1_{(1,1)}$ in green, four $d = 2$ level boxes in red, and sixteen $d = 3$ level boxes, in black.

$$\mathcal{K}_B^d \coloneqq \{1, \ldots, 2^{d-1}\}^3,$$
$$H_d \coloneqq \frac{H_{d-1}}{2},$$
$$x_{\mathbf{k}}^d = x_{(k_1,k_2,k_3)}^d \coloneqq x_{(1,1,1)}^1 + (k_1 H_d - H_1, k_2 H_d - H_1, k_3 H_d - H_1).$$

With the above notations for the index set $\mathcal{K}_B^d$, the box sides H_d, and the box-centers $x_{\mathbf{k}}^d$, the box-octree $\mathcal{B}$ can defined as the set of all boxes $B_{\mathbf{k}}^d$, as follows

$$\mathcal{B} \coloneqq \{B_{\mathbf{k}}^d \ : \ d = 1, \ldots, D, \text{ and } \mathbf{k} \in \mathcal{K}_B^d\}.$$

Note that, by definition, on every level d, $1 \leq d \leq D$, there are $N_B^d \coloneqq 2^{d-1}$ boxes in each coordinate direction for a total of $(N_B^d)^3$ boxes on any level d in the octree structure $\mathcal{B}$.

A two-dimensional illustration of a 3-leveled box-octree is depicted in Figure 3.6. The above Definition 6 of the box-octree, for a given set of points Γ_N, utilizes the unique *axis-aligned minimum bounding-box* [71] of Γ_N as the initial box to uniquely define $x_{(1,1,1)}^1$ and H_1, and, thus, results in a unique D-leveled octree structure $\mathcal{B}$ for Γ_N due to the assumptions that Γ_N consists of at least two distinct points. In

fact, the usage of the minimum bounding box is solely for the sake of concreteness in what follows. Any different choice of initial bounding box—e.g., to facilitate a more advantageous partition of Γ_N—may be employed for practical purposes.

The IFGF algorithm iterates through the octree levels, starting from level D (the smallest box-level) and ending at level 3, to accumulate the field contributions of point sources within each box on each level. These fields are defined analogously to (3.3), as follows.

Definition 7. *Let $\mathcal{B}$ denote a D-leveled box-octree for the surface discretization Γ_N and let $B^d_{\mathbf{k}} \in \mathcal{B}$ denote a level-d box. The* field emitted by point sources placed at the surface discretization points $x' \in B^d_{\mathbf{k}}$ *evaluated at a point $x \in \mathbb{R}^3$ is denoted by $I^d_{\mathbf{k}}(x)$ and defined as follows.*

$$I^d_{\mathbf{k}}(x) \coloneqq \sum_{x' \in \left(B^d_{\mathbf{k}} \cap \Gamma_N\right) \setminus \{x\}} a(x') G(x, x'), \tag{3.22}$$

where $a(x')$ denotes the coefficient in sum (3.1) *associated with the point x'.*

Similarly, the notion of the analytic factor and the centered factor, as in (3.7), is suitably extended in the context of the octree structure as follows.

Definition 8 (Centered and analytic factor)**.** *Let $\mathcal{B}$ denote a box-octree and let $B^d_{\mathbf{k}} \in \mathcal{B}$ denote a box in the box-octree. The IFGF factorization of the Green function G for $x' \in B^d_{\mathbf{k}}$ takes the form*

$$G(x, x') = G(x, x^d_{\mathbf{k}}) g^d_{\mathbf{k}}(x, x'). \tag{3.23}$$

The functions $G(\cdot, x^d_{\mathbf{k}})$ and $g^d_{\mathbf{k}}(\cdot, x')$ are called the centered factor *and the* analytic factor*, respectively.*

Using the factorization (3.23), the field $I^d_{\mathbf{k}}$ in (3.22) generated by point sources placed within the box $B^d_{\mathbf{k}}$ at any point $x \in \mathbb{R}^3$ may be expressed, analogously to (3.5), in the form

$$\begin{aligned} I^d_{\mathbf{k}}(x) &= \sum_{x' \in \left(B^d_{\mathbf{k}} \cap \Gamma_N\right) \setminus \{x\}} a(x') G(x, x') = G(x, x^d_{\mathbf{k}}) F^d_{\mathbf{k}}(x), \quad \text{where} \\ F^d_{\mathbf{k}}(x) &\coloneqq \sum_{x' \in \left(B^d_{\mathbf{k}} \cap \Gamma_N\right) \setminus \{x\}} a(x') g^d_{\mathbf{k}}(x, x'). \end{aligned} \tag{3.24}$$

An *octree-level* consists of all boxes of a given size as per Definition 6 and is indicated by the superscript in the notation of the boxes and box-centers in the same definition. The relative position in x, y, and z direction of a box $B^d_{\mathbf{k}}$ on any level d in the octree structure is indicated by the subscript multi-index $\mathbf{k}$.

Typically only a small fraction of the boxes on a given level d intersect the discrete surface Γ_N; the set of all such *level-d relevant boxes* is denoted by $\mathcal{R}^d_B$ and is rigorously defined in Definition 9.

Definition 9 (Relevant boxes)**.** *Let $\mathcal{B} = \mathcal{B}(D, \Gamma_N)$ denote a D-level octree structure for the surface discretization Γ_N. The* set of level-d relevant boxes*, $\mathcal{R}^d_B$, $1 \le d \le D$, of $\mathcal{B}$ consists of all level-d boxes $B^d_{\mathbf{k}} \in \mathcal{B}$ with non-empty intersection with the discrete surface Γ_N.*

$$\mathcal{R}^d_B \coloneqq \{B^d_{\mathbf{k}} \in \mathcal{B} \,:\, \mathbf{k} \in \mathcal{K}^d_B, B^d_{\mathbf{k}} \cap \Gamma_N \neq \emptyset\}.$$

The set of all relevant boxes *in the box-octree structure $\mathcal{B}$ is defined as the union over all sets of level-d relevant boxes and denoted by $\mathcal{R}_B$:*

$$\mathcal{R}_B \coloneqq \bigcup_{d=1,\dots,D} \mathcal{R}^d_B.$$

Clearly, for each $d = 1, \dots, D$, out of the $(N^d_B)^3$ level-d boxes in any given octree structure $\mathcal{B}$, only $O\left((N^d_B)^2\right)$ are relevant boxes as $d \to \infty$, since Γ_N is a set of points on a two-dimensional surface Γ—a fact that plays an important role in the evaluation of the computational cost of the IFGF method. The set $\mathcal{N}B^d_{\mathbf{k}} \subset \mathcal{R}^d_B$ of *neighboring boxes* of a given box $B^d_{\mathbf{k}}$ is defined as the set of all relevant level-d boxes $B^d_{\mathbf{a}}$ such that $\mathbf{a}$ differs from $\mathbf{k}$, in absolute value, by an integer not larger than one, in each one of the three coordinate directions: $\|\mathbf{a} - \mathbf{k}\|_\infty \le 1$. The *neighboring points* $\mathcal{U}B^d_{\mathbf{k}} \subset \mathbb{R}^3$ of $B^d_{\mathbf{k}}$, in turn, is defined as the set of points in the boxes neighboring $B^d_{\mathbf{k}}$. These two concepts are formalized in Definition 10.

Definition 10 (Neighbor boxes)**.** *Let $\mathcal{B}$ denote a D-leveled box-octree for the surface discretization Γ_N and let $B^d_{\mathbf{k}} \in \mathcal{B}$ denote a level-d box ($1 \le d \le D$). Let $\|\cdot\|_\infty : \mathbb{R}^3 \to \mathbb{R}$ denote the classical maximum norm. The set of* neighbor boxes $\mathcal{N}B^d_{\mathbf{k}}$ *and the associated set of* neighbor points $\mathcal{U}B^d_{\mathbf{k}}$ *of the box $B^d_{\mathbf{k}}$ are defined as follows.*

$$\begin{aligned} \mathcal{N}B^d_{\mathbf{k}} &\coloneqq \left\{B^d_{\mathbf{a}} \in \mathcal{R}^d_B \,:\, \|\mathbf{a} - \mathbf{k}\|_\infty \le 1\right\}, \\ \mathcal{U}B^d_{\mathbf{k}} &\coloneqq \bigcup_{B \in \mathcal{N}B^d_{\mathbf{k}}} B \cap \Gamma_N. \end{aligned} \tag{3.25}$$

Remark 7. *As per the above definition, a box $B_{\mathbf{k}}^d$ is a neighbor to itself.*

An important aspect of the proposed hierarchical algorithm concerns the application of IFGF interpolation methods to obtain field values for groups of sources within a box $B_{\mathbf{k}}^d$ at points farther than one box away (and thus outside the neighborhood of $B_{\mathbf{k}}^d$, where either direct summation ($d = D$) or interpolation from $(d+1)$-level boxes ($(D-1) \geq d \geq 1$) is applied), but that are not sufficiently far from the source box $B_{\mathbf{k}}^d$ to be handled by the next level, $(d-1)$, in the interpolation hierarchy, and which must therefore be handled as part of the d-level interpolation process. The associated *cousin box* concept is defined in terms of the hierarchical parent-child relationship in the octree $\mathcal{B}$, wherein the definitions of *parent box* $\mathcal{P}B_{\mathbf{k}}^d \in \mathcal{R}_B^{d-1}$ and the set $\mathcal{C}B_{\mathbf{k}}^d \subset \mathcal{R}_B^{d+1}$ of *child boxes* of the box $B_{\mathbf{k}}^d$ are stated in Definitions 11 and 12, respectively.

Definition 11 (Parent box)**.** *Let $\mathcal{B}$ denote a D-leveled box-octree for the surface discretization Γ_N and let $B_{\mathbf{k}}^d \in \mathcal{B}$, for $2 \leq d \leq D$. The* parent box $\mathcal{P}B_{\boldsymbol{k}}^d$ *of the box $B_{\mathbf{k}}^d$ is defined as follows.*

$$\mathcal{P}B_{\boldsymbol{k}}^d \coloneqq B_{\mathbf{a}}^{d-1} \quad (\mathbf{a} \in \mathcal{K}_B^{d-1}),$$
$$\text{where } B_{\boldsymbol{a}}^{d-1} \text{ is the unique level } (d-1) \text{ box satisfying} \quad B_{\mathbf{k}}^d \subset B_{\boldsymbol{a}}^{d-1}.$$

Definition 12 (Child boxes)**.** *Let $\mathcal{B}$ denote a D-leveled box-octree for the surface discretization Γ_N and let $B_{\mathbf{k}}^d \in \mathcal{B}$, for $1 \leq d \leq D-1$. The* children of the box $B_{\mathbf{k}}^d$, $\mathcal{C}B_{\mathbf{k}}^d$, *are defined as follows.*

$$\mathcal{C}B_{\mathbf{k}}^d \coloneqq \left\{ B_{\mathbf{a}}^{d+1} \in \mathcal{R}_B^{d+1} \,:\, \mathcal{P}B_{\mathbf{a}}^{d+1} = B_{\mathbf{k}}^d \right\}.$$

These definitions of the *parent box* and the *child boxes* lead to the notion of *cousin boxes* of a level-$(d+1)$ box $B_{\mathbf{k}}^{d+1}$ ($1 \leq d \leq D-1$), namely, non-neighboring $(d+1)$-level boxes which are nevertheless children of neighboring d-level boxes. Similarly to the neighboring boxes and the neighboring points, the *cousin boxes* $\mathcal{M}B_{\mathbf{k}}^d$ and associated *cousin points* $\mathcal{V}B_{\mathbf{k}}^d$ are stated in Definition 13.

Definition 13 (Cousin boxes)**.** *Let $\mathcal{B}$ denote a D-leveled box-octree for the surface discretization Γ_N and let $B_{\mathbf{k}}^d \in \mathcal{B}$ denote a level-d box ($1 \leq d \leq D$). The set of* cousin boxes $\mathcal{M}B_{\mathbf{k}}^d$ *and the associated set of* cousin points $\mathcal{V}B_{\mathbf{k}}^d$ *of the box $B_{\mathbf{k}}^d$ are defined as follows.*

$$\begin{aligned} \mathcal{M}B_{\mathbf{k}}^d &\coloneqq \left(\mathcal{R}_B^d \setminus \mathcal{N}B_{\mathbf{k}}^d\right) \cap \mathcal{C}\mathcal{N}\mathcal{P}B_{\mathbf{k}}^d, \\ \mathcal{V}B_{\mathbf{k}}^d &\coloneqq \bigcup_{B \in \mathcal{M}B_{\mathbf{k}}^d} B \cap \Gamma_N. \end{aligned} \tag{3.26}$$

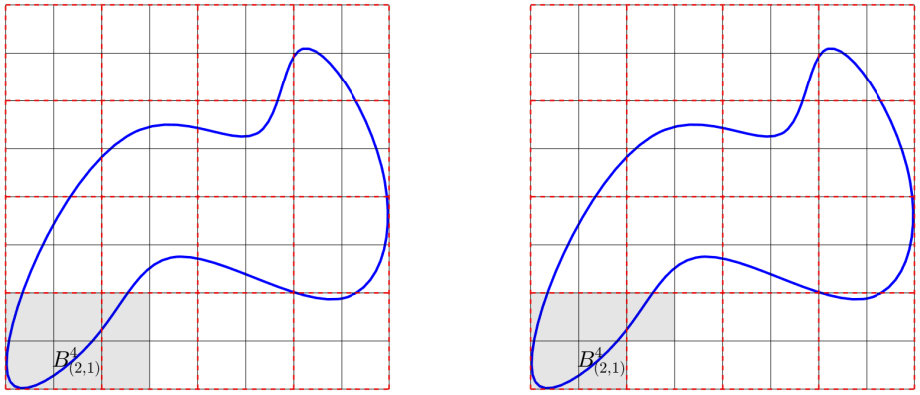


Figure 3.7: Two-dimensional illustration of the neighbors of the fourth-level box $B^4_{(2,1)}$. The left panel shows all possible neighbors of the box $B^4_{(2,1)}$ in gray. The right panel shows the actual neighbor boxes, as in Definition 10, resulting from the intersection with an exemplary scatterer (blue curve) in gray.

Similarly to the above concept of cousin boxes of a box, the set of level-d cousin boxes of a point $x \in \Gamma_N$, $\mathcal{M}^d(x)$, *is given by*

$$\mathcal{M}^d(x) \coloneqq \left\{ B^d_{\mathbf{k}} \in \mathcal{R}^d_B : x \in \mathcal{V} B^d_{\mathbf{k}} \right\}. \tag{3.27}$$

The concept of cousin boxes is illustrated in Figure 3.8 for a two-dimensional example, wherein the cousins of the level-4 box $B^4_{(2,1)}$ are shown in gray in the right panel.

Remark 8. *By definition, two side-H cousin boxes are at a distance that is no larger than $3H$ from each other. It follows that all the cousin boxes of a given level-d box are contained in the set of $6 \times 6 \times 6$ level-d boxes contained in the $3 \times 3 \times 3$ level-$(d-1)$ neighbors of the parent box.*

In view of Remark 8, the number of cousin boxes of a given box is bounded by the constant $6^3 - 3^3 = 189$ (namely, the number of children of the parent's neighbors which are not neighbors of the given box), which is independent of the level d and the number N of surface discretization points—a fact that is exploited in the complexity analysis of the IFGF method.

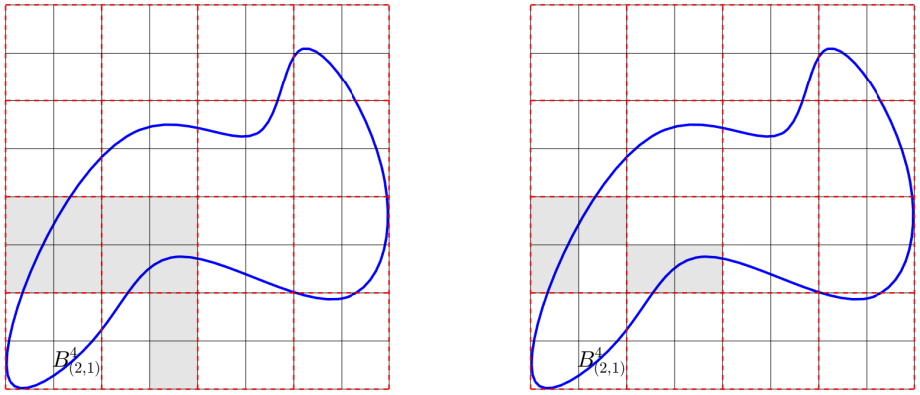


Figure 3.8: Two-dimensional illustration of the cousins of the fourth-level box $B^4_{(2,1)}$. The left panel shows all children of the parent's neighbors of the box $B^4_{(2,1)}$ in gray. The right panel shows the actual cousin boxes, as in Definition 13, resulting from the intersection with an exemplary scatterer (blue curve) in gray.

3.5 Cone segments

On the basis of the discussion presented in the previous Section 3.2, which presented the analyticity properties of the analytic factor (3.5) in the (s, θ, φ) coordinate system, and the cone segment notion introduced in (3.13)-(3.16), the present section discusses the (s, θ, φ) coordinate transformation in the context of the full, hierarchical IFGF method in more detail, thus, finalizing the description of the unique box-cone structure inherent in the IFGF method. As indicated above, the IFGF interpolation procedure consists of piece-wise interpolation in the (s, θ, φ) system over interpolation intervals of size $\Delta_{s,d}$, $\Delta_{\theta,d}$, and $\Delta_{\varphi,d}$, respectively, which depend on the level d in the underlying box-octree structure $\mathcal{B}$.

Definition 14 (Angular interpolation intervals)**.** *Let $n_{C,d} \in \mathbb{N}$ be a given positive integer. Let*

$$\Delta_{\theta,d} = \Delta_{\varphi,d} := \frac{\pi}{n_{C,d}},$$

and

$$\theta_{k,d} := k\Delta_{\theta,d}, \quad k = 0, \ldots, n_{C,d} - 1,$$
$$\varphi_{\ell,d} := \ell\Delta_{\varphi,d}, \quad \ell = 0, \ldots, 2n_{C,d} - 1.$$

The angular interpolation intervals *are then defined as*

$$E_j^{\varphi;d} \coloneqq [\varphi_{j-1,d}, \varphi_{j,d}) \quad \textit{and}$$
$$E_{i,j}^{\theta;d} \coloneqq \begin{cases} [\theta_{n_{C,d}-1}, \pi] & \textit{for} \quad i = n_{C,d},\ j = 2n_{C,d} \\ (0, \Delta_{\theta,d}) & \textit{for} \quad i = 1,\ j > 1 \\ [\theta_{i-1,d}, \theta_{i,d}) & \textit{otherwise,} \end{cases} \tag{3.28}$$

The proposed interpolation strategy additionally relies on a number $n_{s,d} \in \mathbb{N}$ of disjoint *radial interpolation intervals* $E_k^{s;d}$.

Definition 15 (Radial interpolation intervals)**.** *Let $n_{s,d} \in \mathbb{N}$ denote a positive integer and let $\eta = \sqrt{3}/3$ be as in Section 3.2. Further, let*

$$\Delta_{s,d} \coloneqq \frac{\eta}{n_{s,d}}.$$

The radial interpolation intervals *of size $\Delta_{s,d}$ are then defined as*

$$E_k^{s;d} = [(k-1)\Delta_{s,d}, k\Delta_{s,d}) \subset [0, \sqrt{3}/3], \quad k = 1, \dots, n_{s,d}.$$

Thus, in all, the IFGF approach utilizes an overall number $N_{C,d} \coloneqq n_{s,d} \times n_{C,d} \times 2n_{C,d}$ of *interpolation domains*.

Definition 16 (Interpolation domain)**.** *Let $n_{s,d}, n_{C,d} \in \mathbb{N}$ denote positive integers. Let $\gamma = (\gamma_1, \gamma_2, \gamma_3) \in \{1, \dots, n_{s,d}\} \times \{1, \dots, n_{C,d}\} \times \{1, \dots, 2n_{C,d}\} =: \mathcal{K}_C^d$, where $\mathcal{K}_C^d$ denotes the* index set of the cone segments. *The* interpolation domains *are defined as the Cartesian product of the interpolation intervals in the (s, θ, φ) system.*

$$E_\gamma^d \coloneqq E_{\gamma_1}^{s;d} \times E_{\gamma_2}^{\theta;d} \times E_{\gamma_3}^{\varphi;d} \subset [0, \sqrt{3}/3] \times [0, \pi] \times [0, 2\pi), \tag{3.29}$$

Note that in Definition 16, the dependency of γ on the level d was dropped in the notation, but made explicit with the superscript in the notation of the interpolation domain E_γ^d.

Since the parametrization $\mathbf{x}$ in (3.10) depends on the box size $H = H_d$, and thus, on the level d, the following notation for the d-level parametrization is used

$$\mathbf{x}^d(s, \theta, \varphi) = \mathbf{x}(\frac{\sqrt{3}H_d}{2r}, \theta, \varphi), \tag{3.30}$$

which coincides with the expression (3.10) with $H = H_d$.

Under the parametrization $\mathbf{x}^d$ in Equation (3.30), the level-d interpolation domains yield the origin-centered real-space *cone segments* C_γ^d, as defined in what follows.

Definition 17 (Origin-centered cone segments)**.** *Let* $\mathbf{x}^d : [0,\eta]\times[0,\pi]\times[0,2\pi) \to \mathbb{R}^3$ *denote the parametrization introduced in* (3.30)*. The* origin-centered cone segments *are defined as the image of the interpolation domains* E_γ^d *introduced in Definition 16 under the parametrization* $\mathbf{x}^d$*:*

$$C_\gamma^d := \{x = \mathbf{x}^d(s,\theta,\varphi) \ : \ (s,\theta,\varphi) \in E_\gamma^d\}. \tag{3.31}$$

A two-dimensional illustration of the interpolation domains and associated cone segments is provided in Figure 3.3.

While the origin-centered cone segments are not utilized in the IFGF algorithm, they allow an elegant definition of the actually utilized box-centered cone segments $C_{\mathbf{k},\gamma}^d$, as follows.

Definition 18 (Box-centered cone segments)**.** *Let* $\mathcal{B}$ *denote a* D*-leveled box-octree and let* $B_{\mathbf{k}}^d \in \mathcal{B}$*,* $1 \le d \le D$*,* $\mathbf{k} \in \mathcal{K}_B^d$ *be a box in the octree structure of side* H_d *and center* $x_{\mathbf{k}}^d$*. Further, let* C_γ^d*,* $\gamma \in \mathcal{K}_C^d$*, be the origin-centered cone segments according to Definition 17. The cone segments* $C_{\mathbf{k},\gamma}^d$ centered at the box $B_{\mathbf{k}}^d$ *are then defined as follows.*

$$C_{\mathbf{k},\gamma}^d := C_\gamma^d + x_{\mathbf{k}}^d = \{x + x_{\mathbf{k}}^d \ : \ x \in C_\gamma^d\}.$$

An illustration of a two-dimensional example of the cone segments and their naming scheme can be found in Figure 3.9. As indicated above, the box-octree $\mathcal{B}$ is accompanied by a cone segment hierarchy $\mathcal{C}$, which consists of all the cone segments co-centered with boxes contained in the box-octree $\mathcal{B}$.

Definition 19 (Cone segment hierarchy)**.** *Let* $\mathcal{B}$ *denote a* D*-leveled box-octree for the surface discretization* Γ_N*. Let* $n_{c,d}, n_{s,d} \in \mathbb{N}$ *be given for* $1 \le d \le D$*, and let the index set* $\mathcal{K}_B^d$ *and* $\mathcal{K}_C^d$ *be as in Definitions 6 and 16, respectively. The cone segment hierarchy* $\mathcal{C}$ *is defined as the set of all box-centered cone segments (cf. Definition 18):*

$$\mathcal{C} := \{C_{\mathbf{k},\gamma}^d \ : \ 1 \le d \le D, \mathbf{k} \in \mathcal{K}_B^d, \gamma \in \mathcal{K}_C^d\}.$$

As discussed in Section 3.1, the cone segments $C_{\mathbf{k};\gamma}^d$, which are part of the IFGF interpolation strategy, are used to effect piece-wise Chebyshev interpolation in the spherical coordinate system (s,θ,φ). The interpolation approach, which is based on the use of discrete Chebyshev expansions, relies on the use of a set $\mathcal{X}C_{\mathbf{k};\gamma}^d$ for each relevant cone segment $C_{\mathbf{k};\gamma}^d$ containing $P = P_s \times (P_{\text{ang}})^2$ Chebyshev *interpolation points* for all $\mathbf{k} \in \mathcal{K}_B^d$ and $\gamma \in \mathcal{K}_C^d$, $d = 1,\dots,D$:

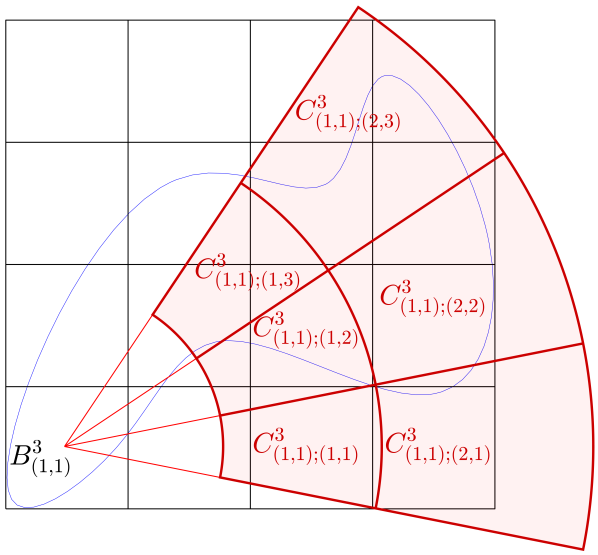


Figure 3.9: Two-dimensional illustrative sketch of the naming scheme used for box-centered cone segments $C^d_{\mathbf{k};\gamma}$ (based on the level-3 box $B^3_{(1,1)}$).

Definition 20 (Interpolation points)**.** *Let $\mathcal{B} = \mathcal{B}(D, \Gamma_N)$ denote a box-octree with cone segment hierarchy C. Further, let $C^d_{\mathbf{k};\gamma} \in C$ and let $\mathbf{x}^d$ be the level-d parametrization introduced in* (3.30) *and let $P_s, P_{ang} \in \mathbb{N}$. The number $P = P_s \times P_{ang} \times P_{ang}$ Chebyshev interpolation points associated with $C^d_{\mathbf{k}}$ are given as follows.*

$$\begin{aligned} \mathcal{X}C^d_{\mathbf{k};\gamma} = \{x \in C^d_{\mathbf{k};\gamma} \,:\, x = \mathbf{x}^d(s_k, \theta_i, \varphi_j) + x^d_{\mathbf{k}}, \\ 1 \le k \le P_s, 1 \le i \le P_{ang}, 1 \le j \le P_{ang}\}, \end{aligned} \tag{3.32}$$

where s_k, θ_i and φ_j denote Chebyshev nodes in the intervals $E^{s;d}_{\gamma_1}$, $E^{\theta;d}_{\gamma_2,\gamma_3}$ and $E^{\varphi;d}_{\gamma_3}$, respectively, and where $x^d_{\mathbf{k}}$ denotes the center of the box $B^d_{\mathbf{k}}$.

A two-dimensional illustration of 3×3 Chebyshev interpolation points within a single cone segment can be found in Figure 3.10.

Clearly, per Definition 18, cone segments are closely related to the box from which they originate. This relation is emphasized by the following concept of *co-centered* boxes and cone segments.

Definition 21 (co-centered boxes and cone segments)**.** *A box $B^d_{\mathbf{k}}$ and a cone segment $C^d_{\mathbf{k},\gamma}$ are said to be* co-centered *if the cone segment is centered at the box center $x^d_{\mathbf{k}}$, as per Definition 18. Note that co-centered structures share the same superscript d*

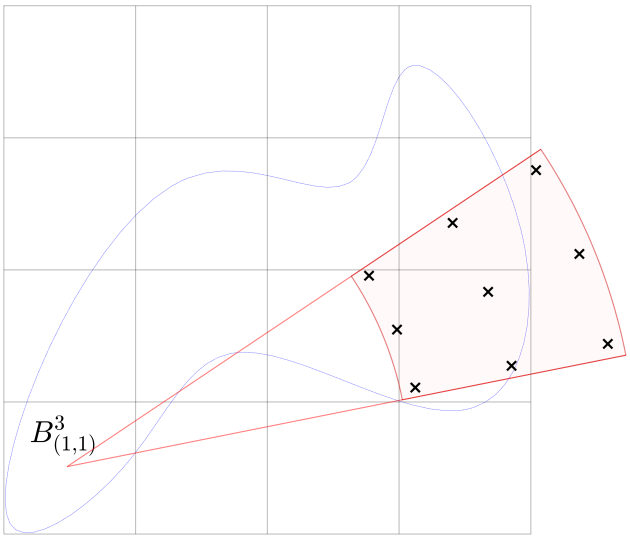


Figure 3.10: Two-dimensional illustration of 3×3 Chebyshev interpolation points associated with the cone segment $C^3_{(1,1);(2,2)}$ in Figure 3.9.

and subscript multi-index $\mathbf{k}$*. The unique relevant level-*d *box* $B^d_{\mathbf{k}}$ *co-centered with the relevant level-*d *cone segment* $C^d_{\mathbf{k};\gamma}$ *is given by*

$$\mathcal{R}_B C^d_{\mathbf{k};\gamma} \coloneqq B^d_{\mathbf{k}}.$$

Further, two cone segments $C_1, C_2 \in \mathcal{C}$ *are called* co-centered *if they are co-centered with the same box, i.e., if* $\mathcal{R}_B C_1 = \mathcal{R}_B C_2$.

Further, analogously to the relevant boxes in the underlying box-octree $\mathcal{B}$, to achieve the desired $\mathcal{O}(N \log N)$ algorithmic complexity, the IFGF method only considers so-called *relevant cone segments* $\mathcal{R}^d_C$ on each level d of the octree structure, i.e., cone segments that, in some way, contribute to the computation of the result. These relevant cone segments are defined as follows.

Definition 22 (Relevant cone segment)**.** *Let* $\mathcal{B} = \mathcal{B}(D, \Gamma_N)$ *denote a* D*-leveled box-octree for the surface discretization* Γ_N*. The* relevant cone segments $\mathcal{R}_C B^d_{\mathbf{k}}$ co-centered with a relevant box $B^d_{\mathbf{k}} \in \mathcal{R}_B$ *are given by*

$$\mathcal{R}_C B^d_{\mathbf{k}} \coloneqq \emptyset, \quad d = 1, 2,$$

$$\mathcal{R}_C B^d_{\mathbf{k}} \coloneqq \Big\{ C^d_{\mathbf{k};\gamma} \,:\, \gamma \in \mathcal{K}^d_C \,,\, C^d_{\mathbf{k};\gamma} \cap \mathcal{V} B^d_{\mathbf{k}} \cap \Gamma_N \neq \emptyset \quad \textit{or}$$

$$C^d_{\mathbf{k};\gamma} \cap \bigcup_{C \in \mathcal{R}_C \mathcal{P} B^d_{\mathbf{k}}} \mathcal{X} C \neq \emptyset \Big\}, \quad d \geq 3.$$

Note that the set of relevant cone segments of a non-relevant box is defined as the empty set. The set of all level-d relevant cone segments*, $\mathcal{R}_C^d$, is further defined as the union of all relevant level-d cone segments*

$$\mathcal{R}_C^d \coloneqq \bigcup_{B \in \mathcal{R}_B^d} \mathcal{R}_C B.$$

Finally, the set of all relevant cone segments*, $\mathcal{R}_C$, is taken as the union of the level-d relevant cone segments over all levels in the box-octree structure*

$$\mathcal{R}_C \coloneqq \bigcup_{d=1,\ldots,D} \mathcal{R}_C^d.$$

Remark 9. *It is important to note that, owing to the placement of the discretization points Γ_N on a two-dimensional surface Γ in three-dimensional space, and due to the cone segment refinement strategy discussed above, the number of relevant boxes is reduced by a factor of $1/4$ as the level is advanced from level $(d+1)$ to level d (at least, asymptotically as $d \to \infty$). Similarly, under the cone segment refinement strategy proposed in view of Theorem 4, the overall number of relevant cone segments per box is increased by a factor of four as the box size is doubled (in the high-frequency regime), so that the total number of relevant cone segments remains essentially constant: $|\mathcal{R}_C^d| \sim |\mathcal{R}_C^{d+1}|$ for all $d = 1, \ldots, D-1$, where $|\mathcal{R}_C^d|$ denotes the total number of relevant cone segments on level d.*

Unlike the box partitioning process, which starts from a single box and proceeds from one level to the next by subdividing each parent box into $2 \times 2 \times 2 = 8$ child boxes (with refinement factors equal to two in each one of the Cartesian coordinate directions, resulting in a number 8^{d-1} boxes at level d), the cone segment partitioning approach proceeds iteratively upwards the tree, starting from the two $d = (D+1)$ initial cone domains

$$E_{(1,1,1)}^{D+1} = [0, \sqrt{3}/3] \times [0, \pi] \times [0, \pi) \quad \text{and}$$
$$E_{(1,1,2)}^{D+1} = [0, \sqrt{3}/3] \times [0, \pi] \times [\pi, 2\pi).$$

(The initial cone domains are only introduced as the initiators of the partitioning process; actual interpolations are only performed from cone domains E_γ^d with $D \geq d \geq 1$.) Thus, starting at level $d = D$ and moving inductively downward to $d = 1$, the cone domains at level d are obtained, from those at level $(d+1)$, by refining each level-$(d+1)$ cone domain by level-dependent refinement factors a_d, i.e., the

number of cone segments in radial and angular directions from one level to the next is taken as $n_{s,d-1} = n_{s,d}/a_d$ and $n_{C,d-1} = n_{C,d}/a_d$. As discussed in Section 3.3, the refinement factors are taken to satisfy $a_d = 1$ or $a_d = 2$ for $D \geq d \geq 2$, but the initial refinement value a_{D+1} is an arbitrary positive integer value.

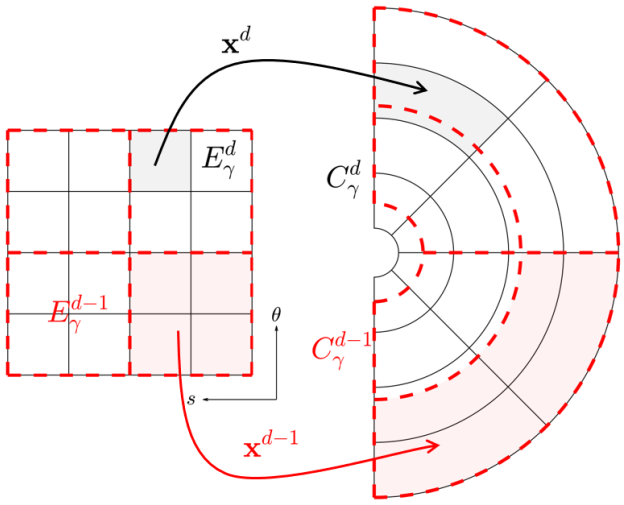
Remark 10. *As indicated later in this thesis, in Chapter 5, the initial refinement values are chosen to achieve* $1 \times 2 \times 4$ *cone segments in the s, θ and φ variables, respectively. These values were empirically determined and, together with a suitable choice of the number of levels D, yield good performance for the targeted* 10^{-3} *accuracy shown in the numerical tests in this thesis.*

After the computation of the values $n_{s,d}$ and $n_{C,d}$, for $d = D, \ldots, 3$, the algorithm proceeds by determining the relevant cone segments in a downward pass starting from level $d = 3$ to level $d = D$, according to Definition 22. As described above, the resulting hierarchy of boxes and cone segments is embodied in two different but inter-related hierarchical structures: the box octree $\mathcal{B}$ and a hierarchy of cone segments C. In the box octree each box contains eight equi-sized child boxes. In the cone segment hierarchy, similarly, each cone segment (spanning certain angular and radial intervals) spawns *up to* eight child segments. The $\kappa H \to \infty$ limit then is approached as the box tree structure is traversed from children to parents (for a sufficiently large number D of levels in the box octree) and the accompanying cone segment structure is traversed from parents to children. This hierarchical strategy and associated structures are described in more in detail in Sections 3.3 and 3.6.

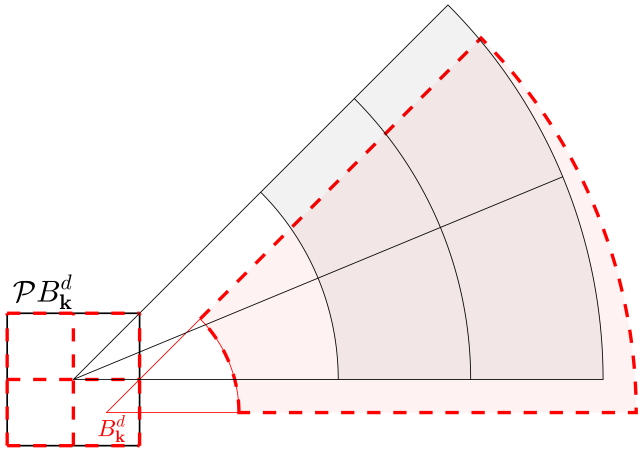
A two-dimensional multi-level setup of the cone segments with a refinement factor $a_d = 2$ and the effect of the parametrization is illustrated in Figure 3.11a. Figure 3.11b, in turn, depicts a two-dimensional sketch of the hierarchical relation of cone segments centered at a box $B^d_{\mathbf{k}}$ and its parent box $\mathcal{P}B^d_{\mathbf{k}}$.

3.6 The IFGF algorithm

The IFGF algorithm consists of two main components, namely, precomputation and operator evaluation. The precomputation stage, which is typically performed only once prior to a series of operator evaluations (that may be required, e.g., as part of an iterative linear-algebra solver for a discrete operator equation), initializes the box octree $\mathcal{B}$ and cone structure C and, in particular, it flags the relevant boxes $\mathcal{R}_B$ and cone segments $\mathcal{R}_C$. The relevant boxes at each level d ($1 \leq d \leq D$) are determined, at a cost of $O(N)$ operations, by evaluation of the integer parts of the quotients of



(a) Two-dimensional illustration of the multi-level cone domains E^d_γ and origin-centered cone segments C^d_γ for two subsequent levels, shown in black and red, respectively.



(b) Two-dimensional illustration of box-centered cone segments, namely, a single $B^d_{\mathbf{k}}$-centered cone segment at level d (in red) and the four (eight in three dimensions) corresponding $\mathcal{P}B^d_{\mathbf{k}}$-centered refined child cone segments at level $d-1$ depicted (in black).

Figure 3.11: Two-dimensional illustration of the hierarchical cone domain structure in (s, θ) space, and corresponding origin-centered and box-centered cone segments.

the coordinates of each point $x \in \Gamma_N$ by the level-d box-size H_d—resulting in an overall cost of $O(N \log N)$ operations for the determination of the relevant boxes at all $D = O(\log N)$ levels. Turning to the determination of relevant cone segments, we first note that, since there are no cousin boxes for any box in either level $d = 1$ (there is only one box in this level) or level $d = 2$ (all boxes are neighbors in this level), by Definition 22, there are also no relevant cone segments in levels $d = 1$ and $d = 2$. To determine the relevant cone segments at level $d = 3$, in turn, the algorithm loops over all relevant boxes $B^3_{\mathbf{k}} \in \mathcal{R}^3_B$, and then over all cousin target points $x \in \Gamma_N \cap \mathcal{V}B^3_{\mathbf{k}}$ of $B^3_{\mathbf{k}}$, and it labels as a relevant cone segment the unique cone segment which contains x. (Noting that, per Definition 18, the cone segments associated with a given relevant box are mutually disjoint, and, consequently, the determination of the cone segment which contains the cousin point x is accomplished at $O(1)$ cost by means of simple arithmetic operations in spherical coordinates.) For the consecutive levels $d = 4, \ldots, D$, the same procedure as for level $d = 3$ is used to determine the relevant cone segments arising from cousin points. In contrast to level $d = 3$, however, for levels $d = 4, \ldots, D$ the relevant cone segments $\mathcal{R}_C\mathcal{P}B^d_{\mathbf{k}}$ associated with the parent box $\mathcal{P}B^d_{\mathbf{k}} \in \mathcal{R}^{d-1}_B$ of a level-d relevant box $B^d_{\mathbf{k}} \in \mathcal{R}^d_B$ also play a role in the determination of the relevant cone segments of the box $B^d_{\mathbf{k}}$. More precisely, for $d \geq 4$ the algorithm additionally loops over all relevant cone segments $C \in \mathcal{R}_C\mathcal{P}B^d_{\mathbf{k}}$ centered at the parent box and all associated interpolation points $x \in \mathcal{X}C$ and, as with the cousin points, flags as relevant the unique cone segment $C^d_{\mathbf{k}}$ associated with the box $B^d_{\mathbf{k}}$ that includes the interpolation point x (cf. Definition 22).

Once the box and cone segment structures $\mathcal{B}$ and C have been initialized, and the corresponding sets of relevant boxes $\mathcal{R}_B$ and relevant cone segments $\mathcal{R}_C$ have been determined, the IFGF algorithm proceeds to the operator evaluation stage. The algorithm thus starts at the initial level D by evaluating directly the expression (3.24) with $d = D$ for the analytic factor $F^D_{\mathbf{k}}$ (which contains contributions from all point sources contained in $B^D_{\mathbf{k}}$) for all level-D relevant boxes $B^D_{\mathbf{k}} \in \mathcal{R}^D_B$ at all the spherical-coordinate interpolation points $x \in \mathcal{X}C^D_{\mathbf{k};\gamma}$ (Definition 20) of all associated relevant cone segments $C^D_{\mathbf{k};\gamma} \in \mathcal{R}_C B^D_{\mathbf{k}}$ co-centered with $B^D_{\mathbf{k}}$. All the associated level-D spherical-coordinate interpolation polynomials $I_P C^D_{\mathbf{k};\gamma}$ of degree $(P-1)$ are then obtained through a direct computation of the coefficients (2.7). The stage D of the algorithm continues by using those level-D interpolants to evaluate the analytic factor $F^D_{\mathbf{k}}(x)$ for all level-D relevant boxes $B^D_{\mathbf{k}}$ through evaluation of the interpolants $I_P C^D_{\mathbf{k};\gamma}(x)$, after which the field values $I^D_{\mathbf{k}}(x)$ are generated via multiplication by the

centered factor at (i) All cousin target points $x \in \Gamma_N \cap \mathcal{V}B^D_{\mathbf{k}}$, and (ii) All parent level interpolation points $x \in \mathcal{X}C^{D-1}_{\tilde{\mathbf{k}};\tilde{\gamma}}$ for all $C^{D-1}_{\tilde{\mathbf{k}};\tilde{\gamma}} \in \mathcal{R}^{D-1}_C$. Finally, the stage D of the algorithm is completed with the generation of the level-$(D-1)$ interpolants $I_P C^{D-1}_{\tilde{\mathbf{k}};\tilde{\gamma}}$ from these point values at the level-$(D-1)$ interpolation points $\mathcal{X}C^{D-1}_{\tilde{\mathbf{k}};\tilde{\gamma}}$ by dividing them by the corresponding level-$(D-1)$ centered factor and utilization of (2.7).

Note that, under the cousin condition $x \in \Gamma_N \cap \mathcal{V}B^D_{\mathbf{k}}$, the variable s takes values on the compact subset $[0, \eta]$ ($\eta = \sqrt{3}/3 < 1$) of the analyticity domain $0 \leq s < 1$ guaranteed by Corollary 1, and, thus, the error-control estimates provided in Theorem 4 guarantee that the required accuracy tolerance is met at the cousin-point interpolation step. Additionally, each cousin target point $x \in \Gamma_N \cap \mathcal{V}B^D_{\mathbf{k}}$ lies within exactly one relevant cone segment $C^D_{\mathbf{k};\gamma} \in \mathcal{R}_C B^D_{\mathbf{k}}$. It follows that the evaluation of the analytic factors (3.24) at a point x for all source boxes $B^D_{\mathbf{k}}$ for which x is a level-D cousin is an $O(1)$ operation—since each surface discretization point $x \in \Gamma_N$ is a cousin point for no more than $189 = 6^3 - 3^3$ boxes (according to Definition 13 and the explanation following it). Therefore, the evaluation of analytic-factor cousin-box contributions at all N surface discretization points requires $O(N)$ operations. This completes the level-D portion of the IFGF algorithm.

At the completion of the level-D stage the field $I^D_{\mathbf{k}}(x)$ generated by each relevant box $B^D_{\mathbf{k}}$ has been evaluated at all cousin surface discretization points $x \in \Gamma_N \cap \mathcal{V}B^D_{\mathbf{k}}$, but field values at surface points farther away from sources, $x \in \Gamma_N \setminus \left(\mathcal{U}B^D_{\mathbf{k}} \cup \mathcal{V}B^D_{\mathbf{k}}\right)$, still need to be obtained; these are produced at stages $d = D-1, \ldots, 3$. (The evaluation process is indeed completed at level $d = 3$ since by construction, we have $\mathcal{U}B^3_{\mathbf{k}} \cup \mathcal{V}B^3_{\mathbf{k}} \supset \Gamma_N$ for any $\mathbf{k} \in \mathcal{K}^3_B$.) In the same manner as the stage D, for each relevant box $B^d_{\mathbf{k}} \in \mathcal{R}^d_B$, the level-$d$ stage of the algorithm ($(D-1) \geq d \geq 3$) proceeds by utilizing the previously (in the level-$d+1$ stage) calculated level-d spherical-coordinate interpolants $I_P C^d_{\mathbf{k};\gamma}$ for each one of the level-d relevant boxes $B^d_{\mathbf{k}} \in CB^{d-1}_{\tilde{\mathbf{k}}}$, to evaluate the analytic factor $F^{d-1}_{\tilde{\mathbf{k}}}(x)$ generated by sources contained within $B^{d-1}_{\tilde{\mathbf{k}}}$ at all points x in all the sets $\mathcal{X}C^{d-1}_{\tilde{\mathbf{k}};\tilde{\gamma}}$ of spherical-coordinate interpolation points associated with parent-level relevant cone segments $C^{d-1}_{\tilde{\mathbf{k}};\tilde{\gamma}} \in \mathcal{R}_C B^{d-1}_{\tilde{\mathbf{k}}}$ emanating from $B^{d-1}_{\tilde{\mathbf{k}}}$. These point values are subsequently used to generate the level-$(d-1)$ Chebyshev interpolants through evaluation of the sums (2.7). The level-d stage is then completed by using the necessary level-d interpolants $I_P C^d_{\mathbf{k};\gamma}$ to evaluate, for all level-d relevant boxes $B^d_{\mathbf{k}}$, the analytic factor $F^d_{\mathbf{k}}(x)$ and, by multiplication with the centered factor, the field $I^d_{\mathbf{k}}(x)$, at all cousin target points $x \in \Gamma_N \cap \mathcal{V}B^d_{\mathbf{k}}$. As in the level D case, these level-d interpolations are performed at a cost of $O(N)$

operations for all surface discretization points—since, as in the level-D case, each surface discretization point (i) Is a cousin target point of $O(1)$ boxes, and (ii) Is contained within one cone segment per cousin box. Performing these steps for all stages $d = D, \ldots 3$ evaluates all fields $I^D_{\mathbf{k}}$, $\mathbf{k} \in \mathcal{K}^D_B$, at all points $x \in \Gamma_N \setminus \mathcal{U}B^D_{\mathbf{k}}$ not included in the neighbors of $B^D_{\mathbf{k}}$. Hence, for a full discrete operator evaluation(3.1), the evaluation of the fields $I^D_{\mathbf{k}}(x)$ at level-D neighboring points $x \in \mathcal{U}B^D_{\mathbf{k}}$ is still missing at this point. These evaluations are performed directly in the present context without any special considerations at a cost of $O(N)$ operations. This completes the algorithm.

Remark 11. *For full solver implementations, where the singularity cannot be removed, as in* (3.1)*, specialized algorithms (e.g., [52]) are required to resolve the singularities.*

As indicated in the Introduction, Section 1.2, the IFGF method does not require a downward pass through the box tree structure—of the kind required by FMM approaches—to evaluate the field at the surface discretization points. Instead, as indicated above, in the IFGF algorithm the surface-point evaluation is performed as part of a single (upward) pass through the tree structure, with increasing box sizes H_d and decreasing values of d, as the interpolating polynomials associated with the various relevant cone segments are evaluated at cousin surface points. Thus, the IFGF approach aggregates contributions arising from large numbers of point sources, but, unlike the FMM, it does so using large number of interpolants of a low (and fixed) degree over decreasing angular and radial spans, instead of using expansions of increasingly large order over fixed angular and radial spans.

It is important to note that, in order to achieve the desired acceleration, the algorithm evaluates analytic factors $F^d_{\mathbf{k}}(x)$ arising from a level-d box $B^d_{\mathbf{k}}$, whether at interpolation points x in the subsequent level, or for cousin surface discretization points x, by relying on interpolation based on (previously computed) interpolation polynomials associated with the $(d+1)$-level relevant children boxes of $B^d_{\mathbf{k}}$, instead of directly evaluating $I^d_{\mathbf{k}}(x)$ using equation (3.24). In particular, all interpolation points within relevant cone segments on level d are also targets of the interpolation performed on level $(d+1)$. Evaluation of interpolant at surface discretization points $x \in \Gamma_N$, on the other hand, are restricted to cousin surface points: evaluation at all points farther away are deferred to subsequent larger-box stages of the algorithm.

Of course, the proposed interpolation strategy requires the creation, for each level-d relevant box $B_{\mathbf{k}}^d$, of all level-d cone segments and interpolants necessary to cover both the cousin surface discretization points as well as all of the interpolation points in the relevant cone segments on level $(d-1)$. We emphasize that the interpolation onto interpolation points requires a re-centering procedure consisting of multiplication by the level d centered factors, and division by corresponding level-$(d-1)$ centered factors (cf. equation (3.24)). We note that, in particular, this re-centering procedure (whose need arises as a result of the algorithm's reliance on the coordinate transformation (3.30) but re-centered at the d-level cube centers for varying values of d) causes the set of the children cone segments not to be geometrically contained within the corresponding parent cone segment (cf. Figure 3.11b). The procedure of interpolation onto interpolation points, which is, in fact, an iterated Chebyshev interpolation method, results only in an error accumulation—due to the well-conditioned nature of the Chebyshev transform—proportional to the number D of levels in the octree structure. Thus, based on the relation $D = O(\log N)$, the overall error scales proportional to $\log N$ as the problem size N is increased.

Using the notation described throughout this thesis, the IFGF algorithm described above is summarized in its entirety in what follows.

- Initialization of relevant boxes and relevant cone segments.
  - Determine the sets $\mathcal{R}_B^d$ and $\mathcal{R}_C^d$ for all $d = 1, \ldots, D$.
- Level D: Start the operator evaluation stage.
  - For every D-level box $B_{\mathbf{k}}^D \in \mathcal{R}_B^D$ evaluate the field $I_{\mathbf{k}}^D(x)$ generated by point sources within $B_{\mathbf{k}}^D$ at all neighboring surface discretization points $x \in \mathcal{U}B_{\mathbf{k}}^D$ by direct evaluation of equation (3.24).
  - For every D-level box $B_{\mathbf{k}}^D \in \mathcal{R}_B^D$ evaluate the analytic factor $F_{\mathbf{k}}^D(x)$ at all interpolation points $x \in \mathcal{X}C_{\mathbf{k};\gamma}^D$ for all $C_{\mathbf{k};\gamma}^D \in \mathcal{R}_C B_{\mathbf{k}}^D$ and generate the interpolants $I_P C_{\mathbf{k};\gamma}$.
- For levels $d = D, \ldots, 3$.
  - For every every box $B_{\mathbf{k}}^d$ evaluate the field $I_{\mathbf{k}}^d(x)$ (equation (3.24)) at every surface discretization point x within the cousin boxes of $B_{\mathbf{k}}^d$, $x \in \mathcal{V}B_{\mathbf{k}}^d$, by evaluating the interpolants $I_P C_{\mathbf{k};\gamma}^d$ and multiplying the result by the centered factor $G(x, x_{\mathbf{k}}^d)$.

– For every every box $B_{\mathbf{k}}^d$ determine the parent box $B_{\mathbf{j}}^{d-1} = \mathcal{P}B_{\mathbf{k}}^d$ and, by way of interpolation of the analytic factor $F_{\mathbf{k}}^d$ through the evaluation of the interpolant $I_P C_{\mathbf{k};\gamma}^d$ and re-centering by the smooth factor $G(x, x_{\mathbf{k}}^d)/G(x, x_{\mathbf{j}}^{d-1})$, obtain the values of the parent-box analytic factors $F_{\mathbf{j}}^{d-1}$ at all level-$(d-1)$ interpolation points corresponding to $B_{\mathbf{j}}^{d-1}$— that is to say, at all points $x \in \mathcal{X}C_{\mathbf{j};\gamma}^{d-1}$ for all $C_{\mathbf{j};\gamma}^{d-1} \in \mathcal{R}_C B_{\mathbf{j}}^{d-1}$ (Note: the contributions of all the children of $B_{\mathbf{j}}^{d-1}$ need to be accumulated at this step.). Finally, generate the parent level interpolants $I_P C_{\mathbf{j};\gamma}^{d-1}$ from these point values.

The corresponding pseudocode, which illustrates the discrete operator evaluation without the precomputation stage and without the singular interactions to level-D neighbors, is presented in Algorithm 2.

For a concise description of the overall method and the parallelization strategy presented in the following Chapter 4, the operator evaluation stage of the IFGF method is split into three parts. First, the level-D evaluation of the field at the interpolation points and the subsequent generation of the first set of interpolants on level D. This part of the algorithm is called in what follows the *LevelDEvaluations* and summarized in Algorithm 3. Secondly, the so-called level-d dependent *Interpolation(d)* which denotes the part of the algorithm responsible for interpolation of the fields $I_{\mathbf{k}}^d$ back to the cousin surface discretization points. It is summarized in Algorithm 4. And, finally, the level-d dependent *Propagation(d)*, which, as the names suggests, propagates the interpolants upwards in the box octree structure and generates the parent level interpolants. The *Propagation* function is summarized in Algorithm 5. Utilizing these three functions, Algorithm 2 may be stated in a shortened form as Algorithm 6. A visual representation of this shortened algorithm is displayed in Figure 3.12, which, in contrast to the pseudocode, also includes the level-D neighbor interactions represented by the *LevelDSingularInteractions* function in that figure.

Algorithm 2 IFGF Method

\\Direct evaluations on the lowest level.
for $B_{\mathbf{k}}^D \in \mathcal{R}_B$ **do**
  for $C_{\mathbf{k};\gamma}^D \in \mathcal{R}_C B_{\mathbf{k}}^D$ **do** ▷ Evaluate F at all relevant interpolation points
    Evaluate and store $F_{\mathbf{k}}^D(\mathcal{X}C_{\mathbf{k};\gamma}^D)$
    Generate interpolant $I_P C_{\mathbf{k};\gamma}^D$
  end for
end for

\\Interpolation onto surface discretization points and parent interpolation points.
for $d = D, \ldots, 3$ **do**
  for $B_{\mathbf{k}}^d \in \mathcal{R}_B$ **do**
    for $x \in \mathcal{V}B_{\mathbf{k}}^d$ **do** ▷ Interpolate at cousin surface points
      Determine $C_{\mathbf{k};\alpha}^d$ s.t. $x \in C_{\mathbf{k};\alpha}^d$
      Evaluate and add to result $I_P C_{\mathbf{k};\alpha}^d(x) \times G(x, x_{\mathbf{k}}^d)$
    end for
    if $d > 3$ **then** ▷ Evaluate F on parent interpolation points
      Determine parent $B_{\mathbf{j}}^{d-1} = \mathcal{P}B_{\mathbf{k}}^d$
      for $C_{\mathbf{j};\gamma}^{d-1} \in \mathcal{R}_C B_{\mathbf{j}}^{d-1}$ **do**
        for $x \in \mathcal{X}C_{\mathbf{j};\gamma}^{d-1}$ **do**
          Determine $C_{\mathbf{k};\alpha}^d$ s.t. $x \in C_{\mathbf{k};\alpha}^d$
          Evaluate and add $I_P C_{\mathbf{k};\alpha}^d(x) \times G(x, x_{\mathbf{k}}^d)/G(x, x_{\mathbf{j}}^{d-1})$
        end for
      end for
    end if
  end for ▷ Generate interpolants on parent level
  for $B_{\mathbf{j}}^{d-1} \in \mathcal{R}_B$ **do**
    for $C_{\mathbf{j};\gamma}^{d-1} \in \mathcal{R}_C B_{\mathbf{j}}^{d-1}$ **do**
      Generate interpolant $I_P C_{\mathbf{j};\gamma}^{d-1}$
    end for
  end for
end for

Algorithm 3 LevelDEvaluations

for $B_{\mathbf{k}}^D \in \mathcal{R}_B$ **do**
  for $C_{\mathbf{k};\gamma}^D \in \mathcal{R}_C B_{\mathbf{k}}^D$ **do**
    Evaluate and store $F_{\mathbf{k}}^D(\mathcal{X} C_{\mathbf{k};\gamma}^D)$
    Generate interpolant $I_P C_{\mathbf{k};\gamma}^D$
  end for
end for

Algorithm 4 Interpolation(d)

for $B_{\mathbf{k}}^d \in \mathcal{R}_B$ **do**
  for $x \in \mathcal{V} B_{\mathbf{k}}^d$ **do**
    Determine $C_{\mathbf{k};\alpha}^d$ s.t. $x \in C_{\mathbf{k};\alpha}^d$
    Evaluate and add to result $I_P C_{\mathbf{k};\alpha}^d(x) \times G(x, x_{\mathbf{k}}^d)$
  end for
end for

Algorithm 5 Propagation(d)

for $B_{\mathbf{k}}^d \in \mathcal{R}_B$ **do**
  Determine parent $B_{\mathbf{j}}^{d-1} = \mathcal{P} B_{\mathbf{k}}^d$
  for $C_{\mathbf{j};\gamma}^{d-1} \in \mathcal{R}_C B_{\mathbf{j}}^{d-1}$ **do**
    for $x \in \mathcal{X} C_{\mathbf{j};\gamma}^{d-1}$ **do**
      Determine $C_{\mathbf{k};\alpha}^d$ s.t. $x \in C_{\mathbf{k};\alpha}^d$
      Evaluate and add $I_P C_{\mathbf{k};\alpha}^d(x) \times G(x, x_{\mathbf{k}}^d)/G(x, x_{\mathbf{j}}^{d-1})$
    end for
  end for
end for
for $B_{\mathbf{j}}^{d-1} \in \mathcal{R}_B$ **do**
  for $C_{\mathbf{j};\gamma}^{d-1} \in \mathcal{R}_C B_{\mathbf{j}}^{d-1}$ **do**
    Generate interpolant $I_P C_{\mathbf{j};\gamma}^{d-1}$
  end for
end for

3.7 Complexity analysis

Under the assumption that the wavenumber κ does not grow faster than $O(\sqrt{N})$, which is natural in the surface scattering context assumed in this thesis, we show in what follows that the IFGF Algorithm 2 runs at an asymptotic computational cost of $O(N \log N)$ operations. The complexity estimates presented in this section incorporate the fundamental assumptions inherent throughout this thesis that fixed

Algorithm 6 IFGF Method

```
LevelDEvaluations()

for d = D, ..., 3 do
    Interpolation(d)
    if d > 3 then
        Propagation(d)
    end if
end for
```

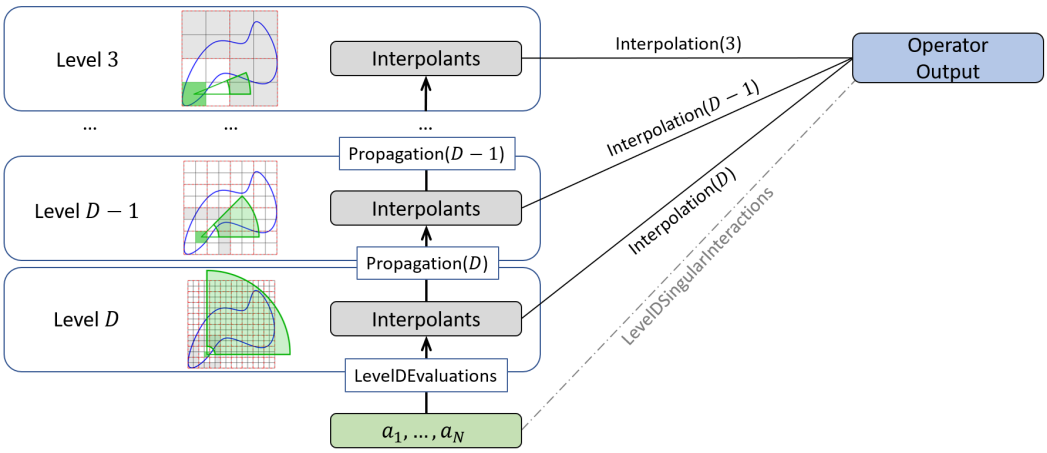


Figure 3.12: Visual representation of the IFGF algorithm, outlined in Algorithm 2, and also expressed in Algorithm 6 in terms of three fundamental functions called *LevelDEvaluations*, *Propagation* and *Interpolation*. Starting from the given coefficients $a_1, \ldots, a_N$ in equation (3.1), the *LevelDEvaluations* function generates the first set of interpolants on level D. The *Interpolation* function interpolates to the surface discretization points and the *Propagation* function facilitates the upwards traversal of the octree structure. Although they are not part of the IFGF algorithm, the interactions between level-D neighbor boxes are represented here by the *LevelDSingularInteractions* function. Note that, unlike other acceleration methods such as the FMM, contributions to the operator output are made at every level, and without a requirement of a downward pass over the octree.

interpolation orders P_s and P_{ang}, and, thus, fixed numbers P of interpolation points per cone segment, are utilized.

For a given choice of interpolation orders P_s and P_{ang}, the algorithm is completely determined once the number D of levels and the numbers $n_{s,D}$ and $n_{C,D}$ of level-D radial and angular interpolation intervals are selected. For a particular configuration, the parameters D, $n_{s,D}$ and $n_{C,D}$ should be chosen in such a way that the overall computational cost is minimized while meeting a given accuracy requirement. An increasing number D of levels reduces the cost of the direct neighbor-evaluations

by performing more of them via interpolation to cousin boxes—which increases the cost of that particular part of the algorithm. The choice of D, $n_{s,D}$ and $n_{C,D}$ should therefore be such that the overall cost of these two steps is minimized while meeting the prescribed accuracy—thus achieving optimal runtime for the overall IFGF method. Note that these selections imply that, for bounded values of $n_{s,D}$ and $n_{C,D}$ (e.g., we consistently use $n_{s,D} = 1$ and $n_{C,D} = 2$ in all of our numerical examples) it follows that $D = \mathcal{O}(\log N)$—since, as it can be easily checked, e.g., increasing $N \to 4N$ and $D \to D + 1$ maintains the aforementioned optimality of the choice of the parameter D. In sum, the IFGF algorithm satisfies the following asymptotics as $\kappa \to \infty$: $\kappa^2 = \mathcal{O}(N)$, $D = \mathcal{O}(\log N)$, $|\mathcal{R}_C B^D_{\mathbf{k}}| = \mathcal{O}(1)$ (for every $B^D_{\mathbf{k}} \in \mathcal{R}^D_B$) and $|\mathcal{R}^D_B| = \mathcal{O}(N)$.

The complexity of the IFGF algorithm equals the number of arithmetic operations performed in Algorithm 2. To evaluate this complexity, we first consider the cost of the level D specific evaluations performed in the "for" loop starting in Line 2. This loop iterates for a total of $\mathcal{O}(N)$ times. The inner loop starting in Line 3, in turn, performs $\mathcal{O}(1)$ iterations. Further, the evaluations of the field in Line 4 and the subsequent generation of the interpolants in Line 5 require $\mathcal{O}(1)$ operations for each cone segment. In total this yields an algorithmic complexity of $\mathcal{O}(N)$ operations for the *LevelDEvaluations* stage of the algorithm.

We consider next the section of the algorithm contained in the loop starting in Line 10, which iterates $\mathcal{O}(\log N)$ times (since $D \sim \log N$). The loop in Line 11, in turn, iterates $\mathcal{O}(N/4^{D-d})$ times, since the number of relevant boxes is asymptotically decreased by a factor of $1/4$ as the algorithm progresses from a given level d to the subsequent level $d-1$. Similarly, the loop in Line 12 performs $\mathcal{O}(4^{D-d})$ iterations—since, as the algorithm progresses from level d to level $d-1$, the side H of the cousin boxes increases by a factor of two, and thus, the number of cousin discrete surface points for each relevant box increases by a factor of four. The interpolation procedure in Line 14, finally, is an $\mathcal{O}(1)$ operation since each point x lies in exactly one cone segment associated with a given box $B^d_{\mathbf{k}}$ (cf. Definition 18 of the cone segments and the previous discussion in Section 3.6) and the interpolation therefore only requires the evaluation of a single fixed order Chebyshev interpolant. A similar count as for the loop in Line 12 holds for the loop in Line 18 which is also run $\mathcal{O}(4^{D-d})$ times since, going from a level d to the parent level $d-1$, the number of relevant cone segments per box increases by a factor four. The "for" loop in Line 19 is performed

$O(1)$ times since the number of interpolation points per cone segment is constant. Altogether, this yields the desired $O(N \log N)$ algorithmic complexity.

In the particular Laplace case $\kappa = 0$, the cost of the algorithm is still $O(N \log N)$ operations, in view of the $O(N \log N)$ cost required by the interpolation to surface points. But owing to the reduced cost of the procedure of interpolation to parent-level interpolation points, which results as a constant number of cone segments per box suffices for $\kappa H_d < 1$ (cf. Section 3.3), the overall $\kappa = 0$ IFGF algorithm is significantly faster than it is for cases in which $\kappa H_d > 1$ for some levels d. In fact, it is expected that an algorithmic complexity of $O(N)$ operations should be achievable by a suitable modification of algorithm in the Laplace case $\kappa = 0$, but this topic is not explored in this thesis at any length.

Finally, we consider the algorithmic complexity of the precomputation stage. According to the first paragraph in Section 3.6, the algorithm corresponding to the determination of relevant cone segments for a single level is executed at a computing cost of $O(N)$ operations. It follows that the full precomputation stage runs at $O(N \log N)$ operations, since the relevant cone segments have to be determined on each and level and the number of levels follows $D = O(\log N)$.

Chapter 4

MASSIVELY PARALLEL IFGF METHOD

The IFGF parallelization scheme proposed in this thesis [72] relies on the use of a hybrid MPI-OpenMP approach. As detailed in Section 4.2, the MPI interface plays two distinct roles in the proposed scheme: it is used to 1) enable distributed-memory parallelization across compute nodes, and 2) in the particular case in which MPI ranks are pinned to NUMA nodes (non-uniform memory access), to distribute work and handle memory access across NUMA nodes within each compute node. Additional details concerning the architecture of the computer used, and, in particular, NUMA nodes, can be found in Sections 5.1 and 2.3. The strategy in point 2) guarantees that memory held by a certain MPI rank is stored within a single NUMA node and can therefore be accessed quickly by all cores within the NUMA node. Moreover, in case 2), access to memory in a different NUMA node within the same compute node is algorithmically effected through MPI in the same manner as access to memory in a different compute node.

The OpenMP parallelization, described in Section 4.1, is used to further distribute the work assigned to each MPI rank. Hence, in the specific hardware implementation demonstrated in this thesis (which is based on the use of compute nodes containing four fourteen-core NUMA nodes), typically four intra-node MPI ranks are used per compute node, each pinned to a single NUMA node, each one of which spawns fourteen OpenMP threads—which, according to our experiments, leads to the best performance achievable without the adverse impact (on, e.g., code complexity, memory requirements, or communication) entailed in pure MPI parallelism within each node. A general discussion on the performance of hybrid MPI-OpenMP approaches can be found in [73–75].

4.1 OpenMP parallelization

Before introducing the proposed OpenMP parallelization scheme, we briefly consider a straightforward OpenMP parallelization strategy which we do not recommend, but which we present for reference. This straightforward and easily implemented strategy results by simply implementing the algorithm depicted in Figure 3.12 by assigning, at each level d, the work associated with groups of relevant boxes to various OpenMP threads (e.g., with a "#pragma omp parallel for" statement), in such a way that each group is handled by a single thread. Equi-distribution of relevant boxes onto the OpenMP threads implies equi-distribution of both the surface discretization points and the computations performed per thread—but only provided 1) the surface discretization points are roughly equi-distributed among the relevant boxes, and, 2) there is a sufficient number of relevant boxes to occupy all OpenMP threads. The difficulties associated with point 1) could be negotiated, in view of the law of large numbers [76, Sec. 13], provided sufficiently many boxes are used, that is to say, provided point 2) is satisfied. In other words, the feasibility of the approach under consideration hinges on the existence of sufficiently many relevant boxes on each level, as required by point 2). Unfortunately, however, for any given discretized surface Γ_N, point 2) is not satisfied at certain levels d in the octree structure, unless only a small number of threads is employed. Noting that, for any surface Γ_N, there are only sixty-four boxes overall on level $d = 3$ of the algorithm (and, in general, even fewer relevant boxes), we see that a definite limit exists on the parallelism achievable by this approach. The method presented in [44] uses this strategy in an MPI context, and it is therefore subject to such a hard limitation on achievable parallelism (although in a somewhat mitigated form, owing to the characteristics of that algorithm, as discussed in Section 1.2). To avoid such limitations, we consider an alternate OpenMP parallelization strategy specifically enabled by the characteristics of the IFGF algorithm, as described in what follows.

The proposed strategy proceeds via parallelization of the three independent programming functions that comprise the IFGF method, namely the *LevelDEvaluations* function, the *Interpolation* function and the *Propagation* function, as introduced in Section 3.6 and illustrated in Figure 3.12. Moreover, these three functions are outlined in Algorithms 3, 4, and 5, respectively. In what follows, we present our strategies for efficient parallelization of each one of these functions separately.

Our approach for an efficient parallelization of the *LevelDEvaluations* function is based on changing the viewpoint from iterating through the level-D relevant boxes

Algorithm 7 Parallel LevelDEvaluations

1: **parallel for** $C^D_{\mathbf{k};\gamma} \in \mathcal{R}^D_C$ **do**
2: Evaluate and store $F^D_{\mathbf{k}}(\mathcal{X}C^D_{\mathbf{k};\gamma})$
3: Generate interpolant $I_P C^D_{\mathbf{k};\gamma}$
4: **end parallel for**

to iterating through the set $\mathcal{R}^D_C$ of all relevant cone segments on level D, introduced in Definition 22. Using this, a parallel version of Algorithm 3 is presented in Algorithm 7. The aforementioned change in viewpoint corresponds to collapsing the two outermost nested loops in Algorithm 3, effectively increasing the number of independent tasks and, consequently, the achievable parallelism. Note that, in a C++ implementation, the "parallel for" construct in Algorithm 7 corresponds to, e.g., a "for" loop preceded by the pragma directive "omp parallel for."

The proposed parallelization of the d-dependent *Propagation* function follows a similar idea as the parallel *LevelDEvaluations* considered above—relying now on iteration over the relevant $(d-1)$ (parent-level) cone segments, which are targets of the interpolation, instead of the relevant level-d boxes emitting the field. This strategy addresses the difficulties arising from the straightforward approach described at the beginning of Section 4.1, for which the number of available tasks to be distributed decreases with the level d and imposes a hard limit on the achievable parallelism. Indeed, in the context of the oscillatory Green functions over two-dimensional surfaces $\Gamma \subset \mathbb{R}^3$ considered in this thesis, for example, wherein the number of relevant cone segments on each level is an approximately constant function of d (see Remark 9), the number of independent tasks available for parallelization remains approximately constant as a function of d. Additionally, the proposed parallel *Propagation* strategy avoids a significant "thread-safety" [77, 78], predicament, that is ubiquitous in the straightforward approach, whereby multiple writes to the same target interpolation point on the parent level take place from different threads. In contrast, the proposed *Propagation* strategy, is by design thread-safe without any additional considerations, since it distributes the targets of the interpolation to the available threads.

Remark 12. *In contrast to the serial implementation of the IFGF method presented in Section 3.6, the practical implementation of this parallel approach requires the algorithm to first determine the relevant box $\mathcal{R}_B C^d_{\mathbf{k};\gamma}$ co-centered with a given relevant cone segment $C^d_{\mathbf{k};\gamma}$ (cf. Definition 21); then to determine the relevant level-$(d+1)$ child boxes $C\mathcal{R}_B C^d_{\mathbf{k};\gamma}$ (cf. Definition 12) of the co-centered box $\mathcal{R}_B C^d_{\mathbf{k};\gamma}$ on*

Algorithm 8 Parallel Propagation(d)

parallel for $C^{d-1}_{\mathbf{j};\gamma} \in \mathcal{R}^{d-1}_C$ **do**
  for $B^d_{\mathbf{k}} \in \mathcal{C}(\mathcal{R}_B C^{d-1}_{\mathbf{j};\gamma})$ **do**
    for $x \in \mathcal{X} C^{d-1}_{\mathbf{j};\gamma}$ **do**
      Determine $C^d_{\mathbf{k};\alpha}$ s.t. $x \in C^d_{\mathbf{k};\alpha}$
      Evaluate and add $I_P C^d_{\mathbf{k};\alpha}(x) \times G(x, x^d_{\mathbf{k}})/G(x, x^{d-1}_{\mathbf{j}})$
    end for
  end for
  Generate interpolant $I_P C^{d-1}_{\mathbf{j};\gamma}$
end parallel for

Algorithm 9 Parallel Interpolation(d)

parallel for $x \in \Gamma_N$ **do**
  for $B^d_{\mathbf{k}} \in \mathcal{M}^d(x)$ **do**
    Determine $C^d_{\mathbf{k};\gamma}$ s.t. $x \in C^d_{\mathbf{k};\gamma}$
    Evaluate $I_P C^d_{\mathbf{k};\gamma}(x) \times G(x, x^d_{\mathbf{k}})$
  end for
end parallel for

level d; and, finally, to find all the interpolants $I_P C$ on the relevant cone segments (Definition 18) $C \in \mathcal{R}_C \mathcal{C} \mathcal{R}_B C^d_{\mathbf{k};\gamma}$ co-centered with the child boxes from which the propagation needs to be enacted.

Utilizing some of the notations in Remark 12, the resulting *Parallel Propagation* algorithm is presented in Algorithm 8.

The proposed parallelization strategy for the third and final IFGF programming function, namely, the *Interpolation* function, relies once again on the strategy used for the *LevelDEvaluations* and *Propagation* functions—which, in the present case, leads to changing the viewpoint from iterating through the relevant boxes to iterating through the surface discretization points that are the target of the interpolation procedure. This approach avoids both, the difficulties mentioned at the beginning of Section 4.1 (concerning the existence of a small number of relevant boxes in the upper levels of the octree structure), as well as thread-safety difficulties similar to those discussed above in the context of the *Propagation* function. Using the definition (3.27), the *Parallel Interpolation* function is stated in Algorithm 9.

In summary, the OpenMP parallelization strategies proposed above for the functions *Parallel LevelDEvaluations*, *Parallel Propagation* and *Parallel Interpolation* are thread-safe by design, and they provide effective work distribution by relying on

iteration over items (relevant cone segments or surface discretization points) that exist in a sufficiently large (and essentially constant) quantities for all levels d, $3 \leq d \leq D$, in the box-octree structure. As a result, the proposed approach effectively eliminates the hard limitation present in the straightforward OpenMP parallelization scheme mentioned at the beginning of this section. Note that the proposed IFGF box-cone parallelization strategy is in general not applicable to other hierarchical acceleration methods, such as, e.g., FMM-type algorithms. Indeed, in contrast to the incremental propagation and surface evaluation approach inherent in the IFGF method, previous acceleration methods rely on the FFT algorithm—which, as discussed in Section 1, leads to inefficiencies in the upper portions of the corresponding octree structures [32, 44].

4.2 MPI parallelization

The proposed MPI parallel IFGF algorithm, which enables both data distribution onto the MPI ranks and efficient communication of data between MPI ranks, is described in detail in Sections 4.2.1 through 4.2.3. The approach mirrors the one proposed in Section 4.1 for the corresponding OpenMP interface. In fact, the MPI parallel scheme is based on slight modifications of the OpenMP parallel Algorithms 7, 8, and 9. As indicated by the theoretical discussion in Section 4.3, the communication overhead is such that the intrinsic IFGF linearithmic complexity previously demonstrated in 3.7 for a single core implementation is preserved on any fixed number N_c of cores; an illustration of this theoretical result on $N_c = 1,680$ cores is presented in the Supplementary Materials Table 5.6. Most importantly, as in the OpenMP case (cf. the last paragraph of Section 4.1), for arbitrarily large numbers D of levels, the MPI IFGF algorithm iterates over items (relevant cone segments or surface discretization points) that exist in a sufficiently large (and essentially constant) quantities for all levels d, $3 \leq d \leq D$, in the box-octree structure. As a result, the strategy results in an overall MPI-OpenMP IFGF parallel scheme without hard limitations on the achievable parallelism as the number of cores grows.

4.2.1 Problem decomposition and data distribution

The distribution of the data required by the IFGF algorithm to the MPI ranks can be summarized as the independent distribution of the set of surface discretization points Γ_N, which are organized on the basis of boxes induced by an octree structure, and the distribution of the set of relevant cone segments on each level $\mathcal{R}_C^d$. Clearly, for an efficient parallel implementation, the distribution used should balance the amount of work performed by each rank while maintaining a minimal memory footprint per rank and while also minimizing the communication between ranks. A concise description of the method used for data distribution to the MPI ranks is presented in what follows, where we let $N_r \in \mathbb{N}$ and $\rho \in \mathbb{N}$ $(1 \leq \rho \leq N_r)$ denote the number of MPI ranks and the index of a specific MPI rank, respectively.

The distribution of the surface discretization points is orchestrated on the basis of an ordering of the set of relevant boxes $\mathcal{R}_B^d$ on each level d, which, in the proposed algorithm, is obtained from a depth-first traversal of the octree structure. This ordering is equivalent to a Morton order of the boxes (as described, e.g., in [58, 60, 61, 79] and depicted by the red "Z"-looking curve in the left panel of Figure 4.1) which, as indicated in [79], can be generated quickly from the positions $\mathbf{k} \in \mathcal{K}_B^D$ of the level-D boxes $B_{\mathbf{k}}^D$ through a bit-interleaving procedure. Ordering the surface discretization points according to the Morton order of the containing level-D boxes also guarantees a Morton order on every other level d, $1 \leq d \leq D-1$. More precisely, at every level d the Morton order introduces a total order $\prec$ on the set of boxes. The ordering of the surface discretization points Γ_N is facilitated by assigning each point $x \in \Gamma_N$ the Morton order of the containing level-D box, which can be computed through a division operation on the coordinates of the point x to get the index $\mathbf{k} \in \mathcal{K}_B^D$ of the containing box with a subsequent bit-interleaving procedure, followed by a simple sorting of the points according to their assigned Morton order. Noting that the map which assigns to each point on Γ_N the Morton order of the containing level-D box is not injective, in order to obtain a total order on all of Γ_N we additionally order in an arbitrary manner subset of points $x \in \Gamma_N$ with the same assigned Morton order. The resulting overall order has the desirable properties that, on every level d, surface discretization points within any given box are contiguous in memory, and that boxes close in real space are also close in memory.

The sorted surface discretization points are distributed to the MPI ranks based on their containing level-D boxes, in such a way that the boxes processed by each each rank are an "interval" set of the form $\{B \in \mathcal{R}_B^D \ : \ B_{\mathbf{k_1}}^D \prec B \prec B_{\mathbf{k_2}}^D\}$, for

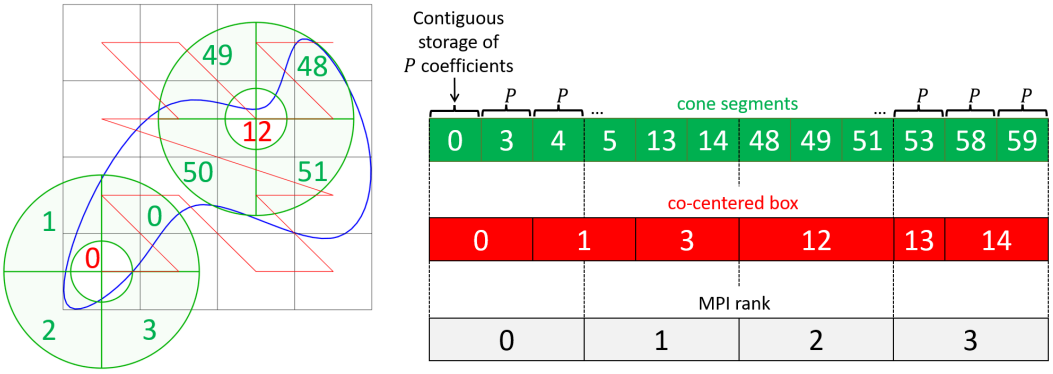


Figure 4.1: Left panel: Two-dimensional example of an ordering of the cone segments based on the Morton order of the boxes on level $d = 3$ with four cone segments per box. The red line indicates the Morton order of the boxes where the red numbers denote the actual Morton code of the containing box. The green numbers denote the ordering of the cone segments in the proposed Morton-based cone-segment ordering. The blue curve denotes a sketch of a scatterer. Right panel: Sketch of a possible cone-segment memory layout, demonstrating the equi-distribution of cone segments among ranks, and emphasizing a central element of the proposed parallelization strategy, namely, that co-centered cone segments may be assigned to different MPI ranks. Note that only relevant boxes and cone segments are stored in memory resulting in some numbers in the ordering being skipped.

suitable choices of $\mathbf{k_1}, \mathbf{k_2} \in \mathcal{K}_B^D$ designed to guarantee that all the boxes on a given rank contain a number of surface discretization points as close as possible to the average value N/N_r. Hence, the smallest boxes in the octree structure represent the smallest "unit" for the distribution of the surface discretization points. The maximum possible deviation in the number of discretization points assigned to a certain MPI rank from the average is therefore given by the maximum number of surface discretization points contained within one level-D box in the octree structure. For reasonable distributions of the discretization points Γ_N on the surface Γ, and for a suitable choice of the number of levels D, this deviation between MPI ranks is typically less than 100 surface discretization points.

The set of *surface discretization points stored in the ρ-th MPI rank*, $1 \leq \rho \leq N_r$, is denoted by $\Gamma_{N,\rho}$. By definition, the subsets $\Gamma_{N,\rho}$ of Γ_N are pairwise disjoint and their union over all MPI ranks $\rho = 1, \ldots, N_r$ equals Γ_N. The distribution of the surface discretization points is used to evenly divide between all MPI ranks the work performed in the *Interpolation* function (OpenMP Algorithm 9). The underlying level-D based distribution of Γ_N is utilized throughout all levels $d = D, \ldots, 3$. Thus, the MPI parallel *Interpolation* function results from the straightforward and

level-independent modification of Line 1 in Algorithm 9, to read $x \in \Gamma_{N,\rho}$ instead of $x \in \Gamma_N$—as shown in Algorithm 11. Naturally, the values of the discrete operator $I(x_\ell)$ in (3.1) computed by the ρ-th MPI rank correspond to points $x_\ell \in \Gamma_{N,\rho}$, and they are therefore also stored in the ρ-th MPI rank. In other words, the set of resulting field values $I(x_\ell)$ is partitioned and stored in the MPI ranks according to the partition utilized for the surface discretization points Γ_N.

The data associated with the level-d relevant cone segments is also distributed to MPI ranks on the basis of a total order—in this case, a total order on the set of level-d cone segments that is based on the Morton order imposed on the level-d boxes, in such a way that co-centered cone segments are close in memory. It should be noted that, for every relevant cone segment $C^d_{\mathbf{k};\gamma} \in R^d_C$, $3 \leq d \leq D$ (see Definition 22), the set of P coefficients that characterize the polynomial interpolants $I_P C^d_{\mathbf{k};\gamma}$ (Section 3.3), which approximate the field $F^d_{\mathbf{k}}$ in (3.24) within the cone segment $C^d_{\mathbf{k};\gamma}$, need to be stored, in appropriately distributed manner, for two consecutive levels. Indeed, for each d, these level-d coefficients are utilized to enable two different interpolation procedures, namely interpolation from level d to interpolation points at the parent-level $(d-1)$ in the *Propagation* function (Line 4 in Algorithm 8), as well as interpolation to the level-d cousin surface discretization points in the *Interpolation* function (Line 3 in Algorithm 9).

The set of level-d relevant cone segments $\mathcal{R}^d_C$ is sorted on the basis of the Morton order induced by the co-centered level-d boxes followed by a suitable sorting of cone segments in each spherical coordinate system—resulting in a total order $\sqsubset$ in the set of all level-d relevant cone segments, as depicted in the left panel of Figure 4.1. (Each set of co-centered cone segments is ordered using the radial direction first, then elevation and finally azimuth, although any other ordering could be used.) Finally, at each level d ($d = D, \ldots, 3$), approximately equi-sized and pair-wise disjoint intervals of relevant cone segments C of the form $\left\{C \in \mathcal{R}^d_C \,:\, C^d_{\mathbf{k_1};\gamma_1} \sqsubset C \sqsubset C^d_{\mathbf{k_2};\gamma_2}\right\}$, for some $\mathbf{k_1}, \mathbf{k_2} \in \mathcal{K}^d_B$ and $\gamma_1, \gamma_2 \in \mathcal{K}^d_C$ (i.e., disjoint intervals of *contiguous* cone segments), are distributed to the MPI ranks, as illustrated in the right panel of Figure 4.1. Note that the specific assignment of cone segments to MPI ranks is solely determined by the order $\sqsubset$ and the number of MPI ranks and cone segments, and it does not otherwise relate to the underlying box tree. In particular, as suggested in the right panel of Figure 4.1, co-centered cone segments may be assigned to different MPI ranks—which induces a flexibility that leads to excellent load-balancing and, therefore, high parallelization efficiency. As is the case for the relevant boxes,

Algorithm 10 MPI Parallel LevelDEvaluations

parallel for $C^D_{\mathbf{k};\gamma} \in \mathcal{R}^D_{C,\rho}$ **do**
    Evaluate and store $F^D_{\mathbf{k}}(\mathcal{X}C^D_{\mathbf{k};\gamma})$
    Generate interpolant $I_P C^D_{\mathbf{k};\gamma}$
end parallel for

Algorithm 11 MPI Parallel Interpolation(d)

parallel for $x \in \Gamma_{N,\rho}$ **do**
    for $B^d_{\mathbf{k}} \in \mathcal{M}^d(x)$ **do**
        Determine $C^d_{\mathbf{k};\gamma}$ s.t. $x \in C^d_{\mathbf{k};\gamma}$
        Evaluate $I_P C^d_{\mathbf{k};\gamma}(x) \times G(x, x^d_{\mathbf{k}})$
    end for
end parallel for

Algorithm 12 MPI Parallel Propagation(d)

parallel for $C^{d-1}_{\mathbf{j};\gamma} \in \mathcal{R}^{d-1}_{C,\rho}$ **do**
    for $B^d_{\mathbf{k}} \in C(\mathcal{R}_B C^{d-1}_{\mathbf{j};\gamma})$ **do**
        for $x \in \mathcal{X}C^{d-1}_{\mathbf{j};\gamma}$ **do**
            Determine $C^d_{\mathbf{k};\alpha}$ s.t. $x \in C^d_{\mathbf{k};\alpha}$
            Evaluate and add $I_P C^d_{\mathbf{k};\alpha}(x) \times G(x, x^d_{\mathbf{k}})/G(x, x^{d-1}_{\mathbf{j}})$
        end for
    end for
    Generate interpolant $I_P C^{d-1}_{\mathbf{j};\gamma}$
end parallel for

the proposed ordering of the relevant cone segments implies that cone segments which are close in real space (i.e., co-centered with the same box and pointing in the same direction or co-centered with boxes which are close in real space) are also close in memory, and, in particular, are likely to be stored within the same MPI rank. Analogously to the notation introduced above for the distributed surface discretization points, the relevant level-d cone segments assigned to a MPI rank with index ρ, $1 \leq \rho \leq N_r$, are denoted by $\mathcal{R}^d_{C,\rho}$. The MPI-capable algorithm is thus obtained by adjusting the loops in the first lines in Algorithms 7 and 8 to only iterate over the level-d relevant cone segments $\mathcal{R}^d_{C,\rho}$ stored in the current rank ρ, as shown in the MPI parallel Algorithms 10 and 12, instead of iterating over all relevant cone segments on level d.

4.2.2 Practical implementation of the box-cone data structures

A C++ implementation of the parallel IFGF box-cone data structures described above, which enables a linearithmic memory and time complexity, is described in detail in what follows.

In the proposed implementation, the geometry Γ_N is stored in three separate arrays X_1, X_2, and X_3 (either C style arrays or std::vector) of size N for the x_1, x_2, and x_3 components of the surface discretization points $(x_1, x_2, x_3) = x \in \Gamma_N$, resulting in a *structure of arrays* (SoA) memory layout [80], which is beneficial as it leads to increased floating-point performance under automatic vectorization on the basis of *single instruction, multiple data* (SIMD) hardware [66, Sec. 2.7] generally available in modern processors. As mentioned above in Section 4.2.1, each one of the three arrays is sorted according to the Morton order of the boxes. Similarly, the real and imaginary parts of the resulting field values $I(x_\ell)$, $1 \leq \ell \leq N$, are stored as two independent arrays, $I_{\Re}$ and $I_{\Im}$, of size N. The order of these field values coincides with the order imposed on the surface discretization points such that $I(x_\ell) = I_{\Re}[k] + \imath I_{\Im}[k]$ at a given point x_ℓ is stored at the same position k in the arrays $I_{\Re}$ and $I_{\Im}$ as the corresponding surface discretization point $x_\ell = (X_1[k], X_2[k], X_3[k])$ in the arrays X_1, X_2 and X_3.

The algorithm enacts the box-octree inherent in the IFGF solver in the form of a *linear octree structure* (cf. [70, 81]). In particular, the proposed linear octree only includes data associated with relevant boxes, and it does not store any information about non-relevant boxes, to avoid $O(N^{3/2})$ memory requirements, as described in detail in what follows. Relevant boxes in the linear octree are represented, on each level $d = 3, \ldots, D$, by the box index $\mathbf{k} \in \mathcal{K}_B^d$ (as described above in Section 3.4) and the equivalent Morton order. Each box stores the position in the arrays X_1, X_2, and X_3 of the first surface discretization point contained in the box in addition to the number of discretization points in the box in a hash map (cf. [70, Section 11]) with average $O(1)$ time and memory complexity for read access (e.g., a std::unordered_map), where the Morton order of the box is utilized as the key. Thus, given a Morton order of a box, the discretization points contained within the box can be determined in $O(1)$ time and memory complexity. Conversely, given any surface discretization point $x \in \Gamma_N$, the three-dimensional index $\mathbf{k} \in \mathcal{K}_B^d$ (for every level $d = 3, \ldots, D$) of the box $B_{\mathbf{k}}^d$ containing the point x and the associated Morton order can be determined through simple division and bit-interleaving, as described in Section 4.2.1, in $O(1)$ time and memory. Overall, this guarantees a true $O(N \log N)$ time and memory

implementation by avoiding the storage of any information regarding non-relevant boxes. Note that the linear octree structure described above is essentially the same as the one presented in [79].

Similarly, for each level d, the relevant cone segments $C^d_{\mathbf{k};\gamma}$, or, more precisely, the real and imaginary parts of the P coefficients representing the interpolants $I_P C^d_{\mathbf{k};\gamma}$ on the relevant cone segments $C^d_{\mathbf{k};\gamma}$, are stored separately in two one-dimensional arrays per rank (following the above partition of the cone segments to the ranks). To associate the three-dimensional cone segment index $\gamma \in \mathcal{K}^d_C$ with the actual coefficients, a hash map for each relevant box $B^d_{\mathbf{k}}$ is used, where the *value* of the hash map is an index pointing to the first of the coefficients $I_P C^d_{\mathbf{k};\gamma}$ in the array of coefficients mentioned above, and where $\gamma \in \mathcal{K}^d_C$ is the *key* of the hash map. (Note that the three-dimensional cone segment index γ, which runs over both relevant and non-relevant cone segments, corresponds to the relative position of the cone segment in the spherical coordinate system centered at the box center.) The usage of a hash map circumvents the storage of any non-relevant cone segment data while maintaining the association with the three-dimensional index γ that allows an easy identification of the cone segment based on its relative position in spherical coordinates. Thus, for a given Cartesian point $x \in \mathbb{R}^3$, this data structure can be used to locate the interpolant $I_P C^d_{\mathbf{k};\gamma}$ for the relevant cone segment $C^d_{\mathbf{k};\gamma} \in R^d_C$ containing the point x through a transformation of x to spherical coordinates centered at the origin of the cone segment $C^d_{\mathbf{k};\gamma}$, a division to get the cone segment index γ and a look-up in the hash map to get the coefficients of the interpolating polynomial. The association of any point with the relevant cone segment containing it can therefore be achieved on average in $O(1)$ time and memory. This is required in the *Interpolation* and *Propagation* function to facilitate the interpolation to cousin surface discretization points and parent-level interpolation points, respectively.

Note that, for increased performance, the hash maps described above and stored on a given rank ρ are required to contain all associations between boxes, cone segments, discretization points and interpolation coefficients utilized by the current rank ρ at any point in the algorithm. In particular, if certain surface discretization points or interpolant coefficients are stored on a different rank $\tilde{\rho} \neq \rho$, but are required in the current rank ρ, the above hash maps are utilized to find the data and, consequently, enable the communication of that data through MPI. While this produces some data duplication, analogously to "halo regions" [66, Sec. 9.6] employed in grid-based

methods, the memory duplicated in the parallel IFGF method is limited to surface discretization points and interpolant coefficients of neighbors and cousins.

4.2.3 Data communication

Clearly, for an MPI rank to access data stored in a different rank, explicit communication between the ranks must take place. The proposed solution, which we favor due to the decreased complexity of the implementation it entails, is based on *one-sided* or *remote memory access* (RMA) communication introduced in MPI-2 [65, Section 5], [66, Section 8]—which utilizes a single MPI_Get or MPI_Put call on the origin rank instead of a coupled MPI_Recv-MPI_Send call (or similar functionalities) involving both the origin and the target rank.

The data any MPI rank may require from other MPI ranks is limited to certain interpolants $I_P C^d_{\mathbf{k};\gamma}$. It is therefore sufficient to store the corresponding coefficients in so-called RMA *windows* (in MPI given by MPI_Win and allocated with, e.g., MPI_Win_allocate), which enable the one-sided communication approach. For increased efficiency, the computations and communications are organized among the ranks on the basis of the following two considerations: 1) For each ρ, $1 \leq \rho \leq N_r$, the ρ-th rank asynchronously collects from other ranks all the data (i.e., the coefficients of the interpolants) it requires to perform *Interpolation* or *Propagation* computations assigned to it; and 2) The communications necessary to collect this data are interleaved with the computations in such a way that while the computations by the *Interpolation* function take place, the communication for the next *Propagation* function is performed and vice versa. This approach, which effectively hides the communications behind computations (thus increasing the performance and parallel efficiency), requires every MPI rank to store all data it obtains from other ranks for one full level-d ($3 \leq d \leq D$) *Interpolation* or *Propagation* step while it continues to store the coefficients it has itself generated—which effectively increases the peak memory per rank requirements slightly (by, e.g., 10% or less).

The level-d dependent *CommunicateInterpolationData* (resp. *CommunicatePropagationData*) programming function in Algorithm 13 (resp. Algorithm 14) encapsulates the communications performed by each rank to obtain, from other ranks, the polynomial coefficients it needs to enact the necessary level-d interpolation computations (resp. interpolation computations onto level-$(d-1)$ interpolation points) required by the *Interpolation* (resp. *Propagation*) function. The *LevelDEvaluations* function does not need any communications since the surface discretization points

Algorithm 13 CommunicatePropagationData(d)

parallel for $C^{d-1}_{\mathbf{j};\gamma} \in \mathcal{R}^{d-1}_{C,\rho}$ **do**
  for $B^d_{\mathbf{k}} \in \mathcal{C}(\mathcal{R}_B C^{d-1}_{\mathbf{j};\gamma})$ **do**
    for $x \in \mathcal{X} C^{d-1}_{\mathbf{j};\gamma}$ **do**
      Find $\tilde{\gamma}$ such that $x \in C^d_{\mathbf{k};\tilde{\gamma}} \in \mathcal{R}_C B^d_{\mathbf{k}}$
      Identify the MPI rank ρ on which $I_P C^d_{\mathbf{k};\tilde{\gamma}}$ is stored
      MPI_Get $I_P C^d_{\mathbf{k};\tilde{\gamma}}$ from rank ρ
    end for
  end for
end parallel for

Algorithm 14 CommunicateInterpolationData(d)

parallel for $x \in \Gamma_{N,\rho}$ **do**
  for $B^d_{\mathbf{k}} \in \mathcal{M}^d(x)$ **do**
    Find $\tilde{\gamma}$ such that $x \in C^d_{\mathbf{k};\tilde{\gamma}} \in \mathcal{R}_C B^d_{\mathbf{k}}$
    Identify the MPI rank ρ on which $I_P C^d_{\mathbf{k};\tilde{\gamma}}$ is stored
    MPI_Get $I_P C^d_{\mathbf{k};\tilde{\gamma}}$ from rank ρ
  end for
end parallel for

$x \in \Gamma_N$, which are required in the *LevelDEvaluations* function, but which are not stored as part of Γ_N, ρ (see previous Section 4.2.1), are duplicated to the ρ-th MPI rank. The rank that stores a level-D relevant cone segment, as described at the beginning of Section 4.2, facilitates the evaluation of the field at the interpolation points of that cone segment and the generation of the interpolants independently from every other MPI rank.

Using the functions 10 through 14, the pseudocode for the proposed overall MPI-OpenMP IFGF algorithm is given in Algorithm 15. Note that access to RMA windows is usually asynchronous and requires some form of synchronization to ensure the data transfer is finalized before the communicated data is accessed. Moreover, the call to the *CommunicatePropagationData* in Algorithm 15 requires for the *Propagation* function to have completed in all ranks targeted by the communication function.

Algorithm 15 IFGF Method

```
 1: LevelDEvaluations()
 2: CommunicatePropagationData(D)
 3:
 4: for d = D, ..., 3 do
 5:     CommunicateInterpolationData(d)
 6:     if d > 3 then
 7:         Propagation(d)
 8:         if d > 4 then
 9:             CommunicatePropagationData(d − 1)
10:         end if
11:     end if
12:     Interpolation(d)
13: end for
```

4.3 Parallel linearithmic complexity analysis

Section 3.7 shows that the basic IFGF algorithm runs on a linearithmic ($\mathcal{O}(N \log N)$) number of arithmetic operations. The present section, in turn, shows that the *communication cost* additionally required by the proposed MPI-OpenMP parallel IFGF algorithm also grows linearithmically—thus, establishing that, on a fixed number of cores, the parallel algorithm runs on an linearithmic overall computing time.

To do this, in view of the data distribution strategy described in Section 4.2.1, it suffices to ensure that both the *Interpolation* and *Propagation* functions require a linearithmic communication cost. Inspection of the corresponding Algorithms 11 and 12 (specifically, lines 4 and 5, respectively) shows that these functions, and, thus, the overall parallel IFGF algorithm, only require communication of certain polynomial coefficients—a task that is effected via the communication Algorithms 14 and 13, respectively. Thus, the analysis of the communication cost amounts to counting the number of coefficients that are communicated, including multiple counts for coefficients that are communicated to multiple ranks, as a result of the application of these two communication algorithms within the overall IFGF algorithm.

In order to count the number of communications effected by each one of these algorithms, we proceed as follows. Noting that, since, 1) as indicated in Remark 9, there are $\mathcal{O}(N)$ relevant cone segments per level, each one of which contains $\mathcal{O}(1)$ data (namely, the P coefficients of a single polynomial interpolant); 2) each cone-segment data is stored in exactly one MPI rank (Section 4.2.1); and, as discussed below for

both communication algorithms, 3) each relevant cone segment is communicated to a uniformly bounded number of MPI ranks at each level $d = 3, \ldots, D$; it follows that for each level d ($3 \leq d \leq D$) a total of $O(N)$ coefficients are communicated by each of the communication algorithms 13 and 14 for each one of the $D = \log N$ levels, at a total communication cost of $O(N \log N)$ coefficients by this algorithm, as desired.

It remains for us to show that point 3) above holds for both communication algorithms. In the case of the propagation communication, we note that each relevant cone segment on any level $d = D, \ldots, 4$ is split into eight smaller cone segments on the parent level $(d - 1)$. Thus, for each level-d relevant cone segment, this results in at most K parent-level cone segments (usually $K = 8$, or possibly a slightly higher number owing to the re-centering procedure associated with the *Propagation* function, but most often $K = 1$) that could be targets for the interpolation procedure in the *Propagation* function. In view of point 2) above, each level-d relevant cone segment must thus communicate coefficients to no more than $O(1)$ ranks, and point 3) follows in this case.

In the case of the interpolation communication, finally, relevant cone-segment coefficients need to be communicated to ranks that store surface discretization points included in boxes that are cousins of the box co-centered with the relevant cone segment. First, on the lowest level D, each relevant box has at most $K = 189$ cousin boxes and since, by design, the surface discretization points contained within each one of the smallest boxes are stored in a single MPI rank (Section 4.2.1), it follows that $O(1)$ (at most 189) different MPI ranks require coefficients contained in each relevant cone segment. Further, since each cone segment is partitioned into eight in the transition from a given level d to a subsequent level $(d - 1)$ (so that the number of relevant level-D boxes contained within a level-$(d - 1)$ cone equals approximately one-fourth of the corresponding number for level-d cone segments, since Γ_N is a discretization of a 2D surface), and since, conversely, the number of MPI ranks storing surface discretization points within a cousin box increases by approximately a factor four in the same d-to-$(d - 1)$ transition, the number of communications per relevant cone segment remains essentially constant as a result of the d-to-$(d - 1)$ level transition. It follows that each relevant cone segment is communicated to a $O(1)$ number of MPI ranks for all levels d, thus establishing the validity of point 3) for the interpolation communication function, and completing the proof of linearithmic complexity of the proposed parallel IFGF algorithm.

Chapter 5

NUMERICAL EXAMPLES

We analyze the performance of the proposed IFGF approach to evaluate the discrete operator (3.1) for various N-point surface discretizations. In each case, the tests concern the accelerated evaluation of the full N-point sum (3.1) at each one of the N discretization points $x_\ell \in \Gamma_N$, $\ell = 1, \ldots, N$—which, if evaluated by direct addition, would require a total of $\mathcal{O}(N^2)$ operations. The capabilities of the IFGF method in a serial and a parallel setting are demonstrated in various configurations, including examples for the Helmholtz ($\kappa \neq 0$) and Laplace ($\kappa = 0$) Green functions.

In all the tests where the Helmholtz Green function is used, the number of levels D in the underlying box octree structure is chosen in such a way that the resulting smallest boxes on level D are approximately a quarter wavelength in size ($H_D \approx 0.25\lambda$). Moreover, for the sake of simplicity, the version of the IFGF algorithm described in Section 3.6 does not incorporate an adaptive box octree (which would stop the partitioning process once a given box contains a sufficiently small number of points) but instead always partitions boxes until the prescribed level D is reached. Hence, a box is a leaf in the tree if and only if it is a level-D box. The cone segments, in turn, (see Definition 18) are chosen in such a way that there are eight cone segments ($1 \times 2 \times 4$ segments in the s, θ and φ variables, respectively) associated with each of the smallest boxes on level D and they are refined according to Section 3.3 for the levels $d < D$. Unless stated otherwise, each cone segment is assigned $P = P_s \times P_{\text{ang}} \times P_{\text{ang}}$ interpolation points with $P_s = 3$ and $P_{\text{ang}} = 5$.

The presentation of the numerical results proceeds as follows: First, Section 5.1 introduces relevant background knowledge, including a rigorous definition of the test geometries, an overview of the hardware used for the test, a presentation of the employed error estimation, and several other related concepts. The tests focusing on the theoretical aspects of the IFGF method, i.e., the $\mathcal{O}(N \log N)$ scaling in time and memory and higher order results, are demonstrated in Sections 5.2 and 5.3, respectively, on the basis of three basic geometries Γ, namely, a sphere, an oblate spheroid and a prolate spheroid, as presented in Section 5.1.1. Section 5.4 then demonstrates results generated with a serial IFGF implementation for the special case of the Laplace equation, before a full OpenMP parallel solver is presented and applied to

engineering motivated problems in Section 5.5. In particular, Section 5.5 gives an overview of the implementation of such a solver before demonstrating acoustic scattering results for a sphere geometry, a submarine geometry, and an aircraft nacelle geometry. Finally, Sections 5.6 through 5.8 demonstrate the massively parallelized MPI-enabled IFGF implementation. More precisely, Sections 5.6 and 5.7 show an investigation of the strong and weak parallel scaling capabilities of the parallel IFGF implementation, whereas Section 5.8 demonstrates the largest problems that can be computed with the parallel IFGF method on our hardware on the basis of very large sphere test cases.

5.1 Background for numerical examples

The current section provides details on several aspects of the numerical results presented below in this chapter. In particular, this section provides details on the hardware used for our serial and parallel tests in Section 5.1.2, before introducing the hardware pinning, which is relevant for the parallel test, in Section 5.1.3. Further, the data points shown throughout all tests are briefly explained in Section 5.1.4. A more detailed explanation of the error estimation, in turn, is presented in Section 5.1.5, after which the the strong and weak efficiency and speedup scalability concepts used to quantify the performance of the parallel IFGF method are detailed in Section 5.1.6; briefly, relative to a base core-number N_c^0, the N_c-core run speedup $S_{N_c^0,N_c}$ and the weak and strong efficiencies $E^w_{N_c^0,N_c}$ and $E^s_{N_c^0,N_c}$ are used to characterize the effectiveness of the proposed parallelization schemes by relating computing times and core numbers under weak-scaling tests (in which N_c is increased proportionally to the size N of the discretization Γ_N) and strong-scaling tests (wherein N_c is increased as N is held fixed).

5.1.1 Test geometries

As indicated above, our numerical examples used to demonstrate the properties of the IFGF method focus on three simple geometries: a sphere of radius a, the oblate spheroid $x^2 + y^2 + (z/0.1)^2 = a^2$ and the prolate spheroid $x^2 + y^2 + (z/10)^2 = a^2$. The latter two geometries are depicted in Figure 5.1. In what follows, the diameter (also referred to as the "size") of a geometry Γ is denoted by

$$d \coloneqq d(\Gamma) \coloneqq \max_{x,y\in\Gamma} |x - y|, \tag{5.1}$$

(not to be confused with the level index d introduced in Section 3.4); clearly, we have $d = 2a$ in the case of the sphere and the oblate spheroid geometries and $d = 20a$ for the prolate spheroid geometry. For our examples, we utilize discretizations Γ_N obtained from use of parametrized surface patches covering Γ and equispaced partitioning of the corresponding parameter spaces, as presented in [52].

These relatively simple geometries present the same kinds of challenges, in the context of the IFGF method, that arise in a wide range of real-world problems, including aircraft, lenses and meta-materials (with a point distribution somewhat similar to that in an oblate spheroid), submarines (prolate spheroid), etc. For example, even though the problem of finding a scattering solution for a submarine is much more challenging than the corresponding problem for a spheroid of the

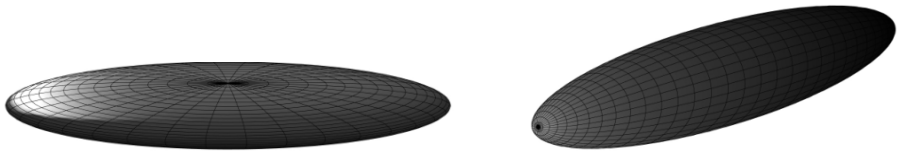


Figure 5.1: Test geometries. Left: Oblate spheroid $x^2 + y^2 + (z/0.1)^2 = a^2$. Right: Prolate spheroid $x^2 + y^2 + (z/10)^2 = a^2$.

same size, in view of the need for accurate integration of singular kernels and adequate representation of the surface Jacobians, the performance of the IFGF method for the evaluation of the discrete operator (3.1) for a submarine should not differ significantly from the corresponding performance on a prolate spheroid of a comparable discretization, point distribution and acoustic size.

5.1.2 Compiler and hardware

All serial tests were performed on a Lenovo X1 Extreme 2018 Laptop with an Intel i7-8750H Processor and 16 GB RAM running Ubuntu 18.04 as operating system. The code is a single core implementation in C++ of Algorithm 2 compiled with the Intel C++ compiler version 19 and without noteworthy effort regarding vectorization.

The parallel IFGF program proposed in Chapter 4, in turn, was also implemented in C++, and the resulting code was compiled with the Intel mpiicpc compiler, version 2021.1, and the Intel MPI library. The following performance-relevant compiler flags were used: "-std=c++20," "-O3," "-ffast-math," "-qopt-zmm-usage=high," "-no-prec-sqrt," "-no-prec-div." All parallel tests were run on our internal *Wavefield* cluster which consists of 30 dual-socket nodes. Each node consists of two Intel Xeon Platinum 8276 processors with 28 cores per processor, i.e., 56 cores per node, and 384 GB of GDDR4 RAM per node. (The Xeon processors we use support hyper-threading, but this capability was not exploited in any of our tests presented in this thesis.) The nodes are connected with HDR Infiniband.

5.1.3 Hardware pinning

For the parallel tests, as indicated in Chapter 4 and Section 2.3, since each compute node in the *Wavefield* cluster consists of four NUMA nodes, we typically run four MPI ranks per node, each pinned to one of these four NUMA nodes through setting the environment variable "I_MPI_PIN_DOMAIN = cache3." The shared memory parallelization with OpenMP (see Section 4.1) is then used for the parallelization within each MPI rank, i.e., within a NUMA node. The parallel scaling within a NUMA node from 1 to 14 cores is investigated below using the OpenMP specific environment variables "OMP_NUM_THREADS=[1-14]," "OMP_PLACES=cores," and "OMP_PROC_BIND=true." The continued scaling, which is achieved with the MPI parallelization, when exceeding 14 cores, is investigated going from one to four MPI ranks (each rank pinned to one NUMA node in the same compute node), which corresponds to the MPI scaling on a single, shared-memory node. Finally, the scaling of the MPI based distributed-memory parallelization is investigated starting from a single node to 16 nodes, where each node is fully utilized with four MPI ranks per node and fourteen cores per rank, as described above. A slightly different hardware pinning is used for the test cases presented in Section 5.8. Instead of pinning one MPI rank to each NUMA node, the test cases in Section 5.8 use a pinning of one MPI rank per compute node each on spawning 54 OpenMP threads. While this pinning is sub-optimal in terms of computing times compared to the one-MPI-rank-per-NUMA-node pinning, it is slightly more beneficial in terms of memory requirements since the amount of data duplication through MPI is minimized.

5.1.4 Data points

In all tests presented below, and in accordance with the notation introduced in the previous sections, N denotes the number of surface discretization points, d the size of the geometry (cf. Section 5.1.1), N_r the number of MPI ranks, and N_c the overall number of cores utilized. Moreover, in the serial runs, the memory is given in the "Memory" column whereas the parallel runs indicate the utilized peak memory per rank in the "Mem/rank" column. The error, as described in detail in the following Section 5.1.5, is presented in the columns labeled ε. Some serial test cases also state the discretization size in "points per wavelengths" which is computed based on the largest equator of the respective geometries and stated in the "PPW" column.

Remark 13. *The PPW have no impact on the accuracy of the IFGF acceleration, since only the discrete operator* (3.1) *is evaluated in the present context, instead*

of an accurate approximation of a full continuous operator. The PPW are only considered here as they provide an indication of the discretization levels that might be used to achieve continuous operator approximations with errors consistent with those displayed in the various tables presented in this section.

Finally, throughout all tests, T denotes the time (in seconds) required for a single application of the IFGF method, i.e., a single evaluation of the discrete operator (3.1), and excludes the precomputation time T_{pre} (which is presented separately in each serial case, and which includes the time required for setup of the data structures and the determination of the relevant boxes and cone segments), but which includes all the other parts of the algorithm presented in Section 3.6, including the direct evaluation at the neighboring surface discretization points on level D.

5.1.5 Numerical error estimation

The errors reported in what follows were computed as the relative L_2 difference ε_M between the full, non-accelerated evaluation of the field $I(x)$, as stated in (3.1), and the IFGF-accelerated evaluation $I_{\text{acc}}(x)$ of (3.1) computed on a randomly chosen subset of $M \leq N$ surface discretization points $x \in \Gamma_N$. This approximate error evaluation is rigorously introduced in the following definition.

Definition 23 (Approximate L_2 error)**.** *Let Γ_N denote a surface discretization and let $x_\ell \in \Gamma_N$ denote the surface discretization points. Further, let $\sigma : \{1, \dots, N\} \to \{1, \dots, N\}$ denote a random permutation and let $M \leq N$ denote some positive integer. The M-*point approximate L_2 error *ε_M of the approximate solution I_{acc} computed by the IFGF method is given by*

$$\varepsilon_M := \sqrt{\frac{\sum_{i=1}^{M} |I(x_{\sigma(i)}) - I_{acc}(x_{\sigma(i)})|^2}{\sum_{i=1}^{M} |I(x_{\sigma(i)})|^2}}. \tag{5.2}$$

In the serial case, $M = 1000$ is chosen. To ensure that $M = 1000$ gives a sufficiently accurate approximation of the error, the exact relative errors ε_N accounting for all N surface discretization points were also obtained for the first three test cases shown in Table 5.1; the results are $\varepsilon_N = 3.56 \cdot 10^{-4}$ ($N = 24576$), $\varepsilon_N = 5.71 \cdot 10^{-4}$ ($N = 98304$) and $\varepsilon_N = 9.28 \cdot 10^{-4}$ ($N = 393216$).

Remark 14. *Exact relative error evaluation for larger values of N is not practical on account of the prohibitive computation times required by the non-accelerated operator evaluation.*

The method is suitably extended to the MPI parallel implementation by using a set of test points x_ℓ that contains a number $M = 1000$ of randomly chosen points on each MPI rank. More precisely, 1000 surface discretization points are randomly chosen on each MPI rank from the distinct set of surface discretization points $\Gamma_{N,\rho}$ each MPI rank ρ ($1 \le \rho \le N_r$) is responsible for, based on the distribution introduced in Section 4.2. The final errors are then accumulated resulting in the overall error estimate

$$\varepsilon := \varepsilon_M \qquad \text{at } M = 1000 \times N_r \text{ points.} \tag{5.3}$$

As a result, the errors are dependent on the number N_r of MPI ranks, which is the reason the shown errors vary slightly as the number of MPI ranks varies (cf. Tables 5.16- 5.18 and 5.19- 5.21).

5.1.6 Weak and strong parallel efficiency concepts

Let $T(N_c, N)$ denote the time required by a run of the parallel IFGF algorithm on an N-point discretization Γ_N, with a given and fixed discretization scheme, of a given surface Γ using N_c cores. Using this notation, for a given N, the strong parallel efficiency $E^s_{N_c^0,N_c}$ that results as the number of cores is increased from N_c^0 to N_c is defined as the quotient of the resulting speedup $S_{N_c^0,N_c}$ to the corresponding ideal speedup value $S^{\text{ideal}}_{N_c^0,N_c}$:

$$S^{\text{ideal}}_{N_c^0,N_c} := \frac{N_c}{N_c^0}, \qquad S_{N_c^0,N_c} := \frac{T(N_c^0,\Gamma_N)}{T(N_c,\Gamma_N)}, \qquad E^s_{N_c^0,N_c} := \frac{S_{N_c^0,N_c}}{S^{\text{ideal}}_{N_c^0,N_c}}.$$

Note that the implicit dependence on N and Γ_N is suppressed in the speedup and efficiency notations.

The weak parallel efficiency $E^w_{N_c^0,N_c} > 0$, in turn, concerns the computing costs that are observed as the numbers N_c of cores are increased proportionally to the problem size N—effectively keeping the number of surface discretization points per core constant—so that as the numbers of cores and discretization points are simultaneously increased from N_c^0 to N_c and from N^0 to N, respectively, the relation

$$N/N^0 = N_c/N_c^0 \tag{5.4}$$

is satisfied. Since the weak scaling concerns varying numbers N of surface discretization points, however, the weak parallel-efficiency concept must correctly account for the linearithmic theoretical scaling of the IFGF algorithm. To do this, we consider the computing time $T(N_c, N)$ required for a run of the algorithm on N_c cores for an N-point discretization of a given, fixed, surface Γ. In view of the linearithmic complexity of the algorithm, perfect weak parallel efficiency would be observed if, for a certain constant K, we had

$$T(N_c, N) = \frac{K}{N_c} N \log N.$$

Thus is to say, under perfect weak parallel scaling, in view of (5.4) we would have

$$\frac{T(N_c, N)}{T(N_c^0, N^0)} = \frac{N_c^0 N \log N}{N_c N^0 \log N^0} = \frac{\log N}{\log N^0}.$$

We therefore define the *weak parallel efficiency* that results as the number of cores is increased from N_c^0 to N_c by

$$E^w_{N_c^0, N_c} := \frac{T(N_c^0, N^0) \log N}{T(N_c, N) \log N^0}.$$

Note that $E^w_{N_c^0, N_c} = 1$ corresponds to perfect weak parallel efficiency, or a weak parallel efficiency of 100%.

5.2 The $N \log N$ scaling

The numerical results shown in this section demonstrate the linearithmic scaling of the serial and the parallel IFGF method.

In particular, the first three tests investigate the scaling of the algorithm as the surface acoustic size is increased and the number of surface discretization points N is increased proportionally to achieve a constant number of points per wavelength. The results of these tests are presented in the Tables 5.1, 5.2, and 5.3 for the aforementioned radius-a sphere, the oblate spheroid and the prolate spheroid (see Section 5.1.1), respectively. The acoustic sizes of the test geometries range from 4 wavelengths to 64 wavelengths in diameter for the normal sphere case, up to 128 wavelengths in size for the case of the oblate spheroid, and up to 512 wavelengths in size for the prolate spheroid.

Several key observations may be drawn from these results. On one hand, we see that, in all cases, the computing and memory costs of the method scale like

N	d	PPW	ε	T_{pre} (s)	T (s)	Memory
24,576	4λ	22.4	$4 \cdot 10^{-4}$	$5.25 \cdot 10^{-1}$	$1.81 \cdot 10^{0}$	25 MB
98,304	8λ		$6 \cdot 10^{-4}$	$3.33 \cdot 10^{0}$	$9.30 \cdot 10^{0}$	80 MB
393,216	16λ		$9 \cdot 10^{-4}$	$1.86 \cdot 10^{1}$	$4.55 \cdot 10^{1}$	315 MB
1,572,864	32λ		$1 \cdot 10^{-3}$	$9.74 \cdot 10^{1}$	$2.21 \cdot 10^{2}$	1,308 MB
6,291,456	64λ		$2 \cdot 10^{-3}$	$4.89 \cdot 10^{2}$	$1.05 \cdot 10^{3}$	5,396 MB

Table 5.1: Computing times T required by the IFGF accelerator for a radius-a sphere of increasing acoustic size $d = \kappa a$, with $(P_s, P_{\text{ang}}) = (3, 5)$, and for various numbers N of surface discretization points—at a fixed number of points-per-wavelength. The precomputation times T_{pre}, the resulting relative accuracy ε and the peak memory ("Memory") used are also displayed.

N	d	PPW	ε	T_{pre} (s)	T (s)	Memory
24,576	4λ	22.4	$1 \cdot 10^{-4}$	$1.30 \cdot 10^{-1}$	$1.44 \cdot 10^{0}$	17 MB
98,304	8λ		$2 \cdot 10^{-4}$	$1.15 \cdot 10^{0}$	$6.52 \cdot 10^{0}$	42 MB
393,216	16λ		$2 \cdot 10^{-4}$	$5.03 \cdot 10^{0}$	$2.87 \cdot 10^{1}$	158 MB
1,572,864	32λ		$3 \cdot 10^{-4}$	$2.63 \cdot 10^{1}$	$1.31 \cdot 10^{2}$	605 MB
6,291,456	64λ		$3 \cdot 10^{-4}$	$1.30 \cdot 10^{2}$	$5.72 \cdot 10^{2}$	2,273 MB
25,165,824	128λ		$4 \cdot 10^{-4}$	$6.27 \cdot 10^{2}$	$2.64 \cdot 10^{3}$	9,264 MB

Table 5.2: Same as Table 5.1, but for an oblate spheroid of equation $x^2 + y^2 + (z/0.1)^2 = a^2$ depicted in Figure 5.1.

N	d	PPW	ε	T_{pre} (s)	T (s)	Memory
393,216	16λ	22.4	$2 \cdot 10^{-3}$	$2.21 \cdot 10^{0}$	$2.19 \cdot 10^{1}$	98 MB
1,572,864	32λ		$6 \cdot 10^{-3}$	$1.16 \cdot 10^{1}$	$9.75 \cdot 10^{1}$	371 MB
6,291,456	64λ		$8 \cdot 10^{-3}$	$5.70 \cdot 10^{1}$	$4.24 \cdot 10^{2}$	1,316 MB
25,165,824	128λ		$1 \cdot 10^{-2}$	$2.72 \cdot 10^{2}$	$1.85 \cdot 10^{3}$	5,317 MB
25,165,824	256λ	11.2	$1 \cdot 10^{-2}$	$3.89 \cdot 10^{2}$	$2.05 \cdot 10^{3}$	5,470 MB
25,165,824	512λ	5.6	$2 \cdot 10^{-2}$	$1.01 \cdot 10^{3}$	$2.57 \cdot 10^{3}$	10,685 MB

Table 5.3: Same as Table 5.1, but for a prolate spheroid of equation $x^2+y^2+(z/10)^2 = a^2$ and a target accuracy of $\varepsilon = 10^{-2}$ (cf. Section 5.1.4 with regards to the selection of PPW in each case).

$O(N \log N)$, thus yielding the expected improvement over the $O(N^2)$ costs required by the straightforward non-accelerated algorithm. Note that, as indicated at the beginning of Chapter 5, the presented implementation does not use an adaptive octree structure. This can lead to large deviations in the number of surface points within boxes, in the number of relevant boxes and in the number of relevant cone segments.

Moreover, the complexity analysis in Section 3.7 assumes certain asymptotics as $N \to \infty$, which may not hold for finite N. These two observations may result in slight departures from the predicted $O(N \log N)$ costs in terms of memory requirements and computing time, especially for the smaller test cases.

Additionally, we note that the computational times and memory required for a given N, which are essentially proportional to the number of relevant cone segments used, depend on the character of the surface considered (since the number of relevant cone segments used is heavily dependent on the surface character), and they can therefore give rise to significant memory and computing-cost variations in some cases. For the oblate spheroid case, for example, the number of relevant cone segments in upward- and downward-facing cone directions is significantly smaller than the number for the regular sphere case, whereas the prolate spheroid requires even less relevant cone segments than the oblate spheroid, resulting in a highly efficient method for elongated geometries. Table 5.3 also shows the incredibly performance of the IFGF method for low-accuracy computations. This is a direct consequence of the $O(P^2)$ scaling of the computing time, where P denotes the number of interpolation points per cone segment.

Table 5.4 demonstrates the scaling of the IFGF method for a fixed number N of surface discretization points and increasing size $d = \kappa a$ for the sphere geometry with radius a. The table demonstrates that the memory requirements and the timings scale like $O(\kappa^2 \log \kappa)$, which is expected, since the interpolation to interpolation points (the *Propagation* function shown in Algorithm 5) used in the algorithm is independent of N and scales like $O(\kappa^2 \log \kappa)$. But the time required for the interpolation back to the surface (the *Interpolation* function shown in Algorithm 4) depends only on N and is therefore constant in this particular test—which explains the slight reductions in overall computing times for a given value of d over the ones displayed in Table 5.1 for the case in which N is scaled proportionally to κ^2.

Table 5.5 shows a similar sphere test but for a sphere of constant acoustic size d and with various numbers N of surface discretization points. As we found earlier, the computation times and memory requirements scale like $O(N \log N)$ (the main cost of which stems from the process of interpolation back to the surface discretization points—the *Interpolation* function; see Algorithm 4). Since the cost of the IFGF method (in terms of computation time and memory requirements) is usually dominated by the cost of the interpolation to interpolation points—the *Propagation* function (Algorithm 5)—which is only dependent on the wavenumber

N	d	PPW	ε	T_{pre} (s)	T (s)	Memory
393,216	16λ	22.4	$9 \cdot 10^{-4}$	$1.86 \cdot 10^{1}$	$4.55 \cdot 10^{1}$	315 MB
	32λ	11.2	$1 \cdot 10^{-3}$	$8.17 \cdot 10^{1}$	$1.33 \cdot 10^{2}$	1,032 MB
	64λ	5.6	$1 \cdot 10^{-3}$	$3.73 \cdot 10^{2}$	$5.63 \cdot 10^{2}$	3,927 MB

Table 5.4: Same as Table 5.1, but for a fixed number N of surface discretization points, demonstrating the scaling of the algorithm as the acoustic $d = \kappa a$ of the sphere is increased independently of the discretization size while maintaining the accelerator's accuracy.

κa, the scaling in N is better than $O(N \log N)$ until N is sufficiently large, so that the process of interpolation back to the surface discretization points requires a large enough portion of the share of the overall computing time—as observed in the fourth and fifth rows in Table 5.5.

N	d	PPW	ε	T_{pre} (s)	T (s)	Memory
24,576	16λ	5.6	$3 \cdot 10^{-4}$	$9.30 \cdot 10^{0}$	$1.66 \cdot 10^{1}$	228 MB
98,304		11.2	$6 \cdot 10^{-4}$	$1.13 \cdot 10^{1}$	$2.23 \cdot 10^{1}$	267 MB
393,216		22.4	$9 \cdot 10^{-4}$	$1.40 \cdot 10^{1}$	$4.14 \cdot 10^{1}$	320 MB
1,572,864		44.8	$1 \cdot 10^{-3}$	$2.34 \cdot 10^{1}$	$1.63 \cdot 10^{2}$	498 MB

Table 5.5: Same as Table 5.1, but for a fixed acoustic size $d = \kappa a$ of the sphere, demonstrating the scaling of the algorithm as N is increased independently of the acoustic size.

To conclude this section, Figure 5.2 presents results of an investigation regarding the linearithmic scaling of the parallel IFGF method for the prolate spheroid geometry on a fixed number of compute nodes, namely, all 30 nodes available in the computer cluster we use, and for a discretization size N ranging from 6,291,456 to 402,653,184, for corresponding diameters d ranging from 512λ to $4,096\lambda$. The data in this figure, which is also presented in tabular form in Table 5.6, was generated by pinning a single MPI rank to each compute node, each of which spawns 56 OpenMP threads, with parameters resulting in an IFGF error $\varepsilon \approx 1.5 \cdot 10^{-2}$ (cf. equation (5.3)). The results show that the linearithmic algorithmic complexity and memory requirements of the basic IFGF algorithm are maintained in the parallel setting. Indeed, the observed complexity even slightly outperforms the postulated $O(N \log N)$ within this range of values of N; cf. Table 5.6 which suggests convergence to exact linearithmic complexity as N grows. Note, in particular, the last

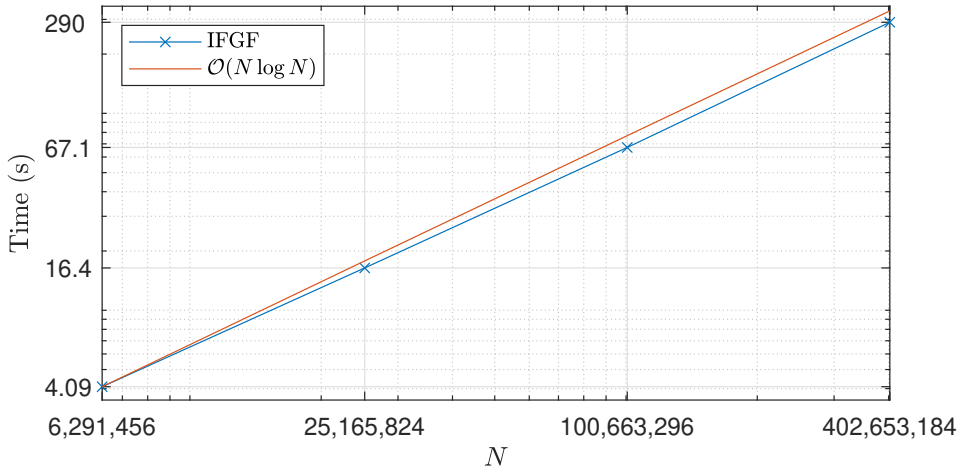


Figure 5.2: Illustration of the linearithmic complexity of the parallel IFGF method, for the prolate spheroid geometry, on 30 compute nodes, with error $\varepsilon \approx 1.5 \cdot 10^{-2}$. The acoustic diameter of the ellipsoid is kept proportional to $\sqrt{N}$, and it ranges from 512λ to $4,096\lambda$. Clearly, the parallel implementation preserves (and, in fact, slightly improves upon) the ideal linearithmic scaling. For this test one MPI rank per node and 56 OpenMP threads per MPI rank were used (resulting in 1680 cores). The peak IFGF memory used per MPI rank (excluding the memory required to store the initial geometry) as well as other additional data in tabular form are presented in Table 5.6.

column of Table 5.6 suggests rapid convergence to exact linearithmic complexity with a well defined proportionality constant, as N grows.

N	d	N_c	T (s)	Mem/rank	$T/(N \log N)$
$6,291,456$	512λ	$1,680$	$4.09 \cdot 10^{0}$	0.50 GB	$9.56 \cdot 10^{-8}$
$25,165,824$	$1,024\lambda$		$1.64 \cdot 10^{1}$	1.89 GB	$8.83 \cdot 10^{-8}$
$100,663,296$	$2,048\lambda$		$6.71 \cdot 10^{1}$	7.42 GB	$8.33 \cdot 10^{-8}$
$402,653,184$	$4,096\lambda$		$2.90 \cdot 10^{2}$	29.73 GB	$8.37 \cdot 10^{-8}$

Table 5.6: Preservation of the linearithmic IFGF scaling in the parallel context. One MPI rank per node and 56 OpenMP threads per MPI rank on 30 compute nodes, resulting in $N_c = 1,680$ cores, for a prolate spheroid geometry were used for this test. The peak memory per MPI rank used by the IFGF method (excluding the memory required to store the initial geometry) is listed in the next-to-last column. The value in the last column suggests convergence to exact $\mathcal{O}(N \log N)$ scaling. All tests were performed with an error of $\varepsilon = 1 \cdot 10^{-2}$

5.3 Higher order results

As indicated in Section 3.3, the accuracy of the IFGF method is determined by the Chebyshev interpolation procedure in each of the cone segments. This fact is in Table 5.7: It demonstrates the scaling of the IFGF method in terms of computing time, memory requirements and achievable accuracy in the number of interpolation points P per cone segment, again on the basis of the regular sphere geometry. For a number $P = P_s P_{\text{ang}}^2$ of interpolation points per cone segment, the computing time and memory required by the IFGF method are expected to scale like $O(P^2)$ and $O(P)$, respectively, while the relative accuracy increases exponentially fast. The predicted scaling can easily be observed by comparing the results from Table 5.7 to the results shown in Table 5.1 for $P = 3 \times 5 \times 5$. In particular, as discussed in the context of the prolate spheroid results shown in Table 5.3, lower accuracy IFGF computations can be produced at extremely low costs.

N	d	PPW	ε	(P_s, P_{ang})	T (s)	Memory
24,576	4λ	22.4	$7 \cdot 10^{-6}$	(5,7)	$8.95 \cdot 10^{0}$	65 MB
98,304	8λ		$1 \cdot 10^{-5}$		$5.05 \cdot 10^{1}$	269 MB
393,216	16λ		$2 \cdot 10^{-5}$		$2.46 \cdot 10^{2}$	1,039 MB
1,572,864	32λ		$2 \cdot 10^{-5}$		$1.20 \cdot 10^{3}$	4,308 MB
24,576	4λ	22.4	$4 \cdot 10^{-7}$	(7,9)	$3.36 \cdot 10^{1}$	134 MB
98,304	8λ		$6 \cdot 10^{-7}$		$1.88 \cdot 10^{2}$	584 MB
393,216	16λ		$8 \cdot 10^{-7}$		$9.83 \cdot 10^{2}$	2,320 MB
1,572,864	32λ		$1 \cdot 10^{-6}$		$4.90 \cdot 10^{3}$	9,633 MB

Table 5.7: Same as Table 5.1, but for two different sets of interpolation orders.

5.4 Laplace equation

In our final pure IFGF example, we consider an application of the IFGF method to a spherical geometry for the Laplace equation. The results are shown in Table 5.8. A perfect $O(N \log N)$ scaling is observed. Note that the *Propagation* portion of the algorithm (Algorithm 5), which requires a significant fraction of the computing time in the Helmholtz case, runs at a negligible cost in the Laplace case—for which a constant number of cone segments can be used throughout all levels, as discussed in Section 3.3.

N	ε	T(s)	Memory
24,576	$2 \cdot 10^{-5}$	$7.81 \cdot 10^{-1}$	25 MB
98,304	$1 \cdot 10^{-5}$	$3.62 \cdot 10^{0}$	69 MB
393,216	$1 \cdot 10^{-5}$	$1.69 \cdot 10^{1}$	246 MB
1,572,864	$1 \cdot 10^{-5}$	$7.45 \cdot 10^{1}$	962 MB
6,291,456	$2 \cdot 10^{-5}$	$3.29 \cdot 10^{2}$	3,676 MB

Table 5.8: Same as Table 5.1, but for the Laplace equation ($\kappa = 0$). The precomputation times (not shown) are negligible in this case, since the cost of the most cost-intensive part of the precomputation algorithm, namely, the determination of the relevant cone segments, is negligible in the present Laplace context. Per the IFGF Laplace algorithmic prescription, a fixed number of cone segments per box is used across all levels in the hierarchical data structure.

5.5 Full solver and sample engineering problems

In this section, we present the first complete acoustic scattering solver [82] based on the above introduced IFGF method for the acceleration of discrete integral operators of the form (3.1). The proposed accelerated solver utilizes the GMRES algorithm (see Section 2.1) to solve the linear system of equations where in each iteration it handles, in addition to the non-neighboring interactions covered by the IFGF method, the singular local integrations by means of a high-order Chebyshev-based singularity resolution methodology [52] (see following Section 5.5.4). In particular, it relies on the IFGF accelerator to evaluate the vast amount of non-local integration points, i.e., for each box $B^D_{\mathbf{k}}$ in the octree structure at all non-neighboring surface discretization points $x \in \Gamma_N \setminus \mathcal{N}B^D_{\mathbf{k}}$. In what follows, an overall parallel OpenMP implementation of the proposed solver is presented and numerical experiments confirm the overall $O(N \log N)$ computational cost as the frequency and discretization sizes are increased. A variety of numerical examples presented in this section demonstrate that the proposed solver enables the efficient solution of large problems over complex geometries on small parallel hardware infrastructures. Numerical examples include acoustic scattering by a sphere of up to 128 wavelengths, an 80-wavelength submarine, and a turbofan nacelle that is more than 80 wavelengths in size, requiring, on a single 28-core processor in the above described *Wavefield* cluster (see Section 5.1.2), computing times of the order of a few minutes per iteration and a few tens of iterations of the GMRES iterative solver (see Section 2.1).

5.5.1 Scattering boundary-value problem

We consider wave propagation in a homogeneous isotropic medium with density ρ, speed of sound c, and no damping [12]. Scattering obstacles are represented by a bounded set $\Omega \subset \mathbb{R}^3$ which is the open complement of an unbounded domain. For time-harmonic acoustic waves, the wave motion can be obtained from the velocity potential $U(x,t) = \Re\{u(x)e^{-\iota\omega t}\}$, where $\omega > 0$ is the angular frequency, and the spatially-dependent complex-valued part $u(x)$ satisfies the exterior problem for the Helmholtz equation shown in (1.1) with $\kappa = \omega/c$; the corresponding acoustic wavelength is given by $\lambda = 2\pi/\kappa$. Denoting the boundary of Ω by Γ, the *sound-soft* obstacle case that we consider requires that $u = 0$ on Γ. Writing the total field $u(x) = u^i(x) + u^s(x)$, where $u^i(x)$ is a given incident field which also satisfies Helmholtz equation, leads to an exterior Dirichlet boundary value problem for the scattered field $u^s(x)$

$$\begin{cases} \Delta u^s(x) + \kappa^2 u^s(x) = 0, & x \in \mathbb{R}^3 \setminus \bar{\Omega}, \\ u^s(x) = -u^i(x), & x \in \Gamma, \\ |x|\left(\frac{x}{|x|} \cdot \nabla u^s(x) - \iota\kappa u^s(x)\right) = 0, & |x| \to \infty. \end{cases} \tag{5.5}$$

5.5.2 Integral representations and integral equations

The solutions to the acoustic scattering problem can be obtained in terms of an integral equation posed on the obstacle boundary, as shown in Section 1.1. In what follows, we cover the specific integral equation relevant to the engineering-motivated problems presented in this section.

Recall from Section 1.1 that the fundamental solution to the Helmholtz equation with positive wavenumber κ is given by (1.8). Utilizing the single and double layer potentials, $\mathcal{S}_\kappa$ and $\mathcal{D}_\kappa$, as presented in the same section in equations 1.6 and 1.7, respectively, the solution to (5.5) can be expressed as a combined-layer potential

$$u^s(x) = \int_\Gamma \left\{ \frac{\partial G(x,y)}{\partial \nu(y)} - \iota\gamma G(x,y) \right\} \varphi(y)\, dS(y), \qquad x \in \mathbb{R}^3 \setminus \bar{\Omega}, \tag{5.6}$$

for a real *coupling parameter* $\gamma \neq 0$, where the density φ is a solution to the integral equation

$$\frac{1}{2}\varphi(x) + \mathcal{D}_\kappa[\varphi](x) - \iota\gamma \mathcal{S}_\kappa[\varphi](x) = f(x), \qquad x \in \Gamma, \tag{5.7}$$

with $f(x) = -u^i(x)$.

5.5.3 Surface representation

Following [52], we partition a scattering surface as the disjoint union of a set of non-overlapping parametrized component *patches*. We also refer to a surface patch as a *logical quadrilateral* (LQ) since it is assumed to be the image of a rectangular reference domain. Given a scattering surface Γ, we thus utilize a number Q of smooth parametrizations

$$y^q : R \to \mathbb{R}^3, \qquad (q = 1, \ldots, Q),$$

from a uv-plane reference domain $R \coloneqq (-1, 1)^2$ onto an LQ patch $\Gamma^q \subset \mathbb{R}^3$ such that

$$\Gamma^q = y^q(R) \quad \text{and} \quad \Gamma = \bigcup_{q=1}^{Q} \Gamma^q. \tag{5.8}$$

A general integral operator defined over Γ can then be evaluated component-wise over each patch Γ^q.

We discretize the patch Γ^q by means of a surface grid containing $N_u^q \times N_v^q$ points given by the image of the tensor-product discretization

$$\left\{u_i = s_i \mid i = 0, \ldots, N_u^q - 1\right\} \times \left\{v_j = s_j \mid j = 0, \ldots, N_v^q - 1\right\},$$

under the parametrization y^q, where the nodes s_j and associated integration weights w_j are given by Fejér's first quadrature rule:

$$s_j = \cos\left(\pi \frac{2j+1}{2J}\right), \tag{5.9}$$

$$w_j = \frac{2}{J}\left[1 - 2\sum_{\ell=1}^{\lfloor J/2 \rfloor} \frac{1}{4\ell^2 - 1} \cos\left(\ell\pi \frac{2j+1}{J}\right)\right], \tag{5.10}$$

for $j = 0, \ldots, J-1$ and J is either N_u^q or N_v^q. The set of all surface discretization points will be denoted by

$$\Gamma_N \coloneqq \bigcup_{q=1}^{Q} \Gamma^q_{N_u, N_v}, \tag{5.11}$$

where N denotes the total number of grid points over all patches.

5.5.4 Chebyshev-based rectangular-polar integral equation solver

This section presents a brief description of the high-order integral equation solver presented in [52]. In that approach, a general integral operator I^q with singular kernel K^q and density φ^q defined over a component patch $\Gamma^q \subset \Gamma$ of a surface Γ is expressed in the parametric form

$$(I^q\varphi)(x) = \int_R K^q(x,u,v)\varphi^q(u,v)J^q(u,v)\,dudv, \tag{5.12}$$

for $x \in \Gamma$, where $K^q(x,u,v) \coloneqq K(x, y^q(u,v))$ and $\varphi^q(u,v) \coloneqq \varphi(y^q(u,v))$, and where $J^q(u,v)dudv$ denotes the element of area.

To compute (5.12) accurately, we use two different high-order methods depending on whether the target point x is less than or greater than some "proximity distance" δ to the integration patch. In detail, letting

$$\operatorname{dist}(x,\Gamma^q) \coloneqq \inf\{\,|x-y| \;:\; y \in \Gamma^q\}, \tag{5.13}$$

denote the distance from a point x to a patch Γ^q (where $|\cdot|$, as before, denotes the Euclidean distance), the set of target points gives rise to "singular" and "nearly-singular" over Γ^q is defined by

$$\Omega_q^{s,\delta} \coloneqq \{x \in \Gamma \;:\; \operatorname{dist}(x,\Gamma^q) \le \delta\}. \tag{5.14}$$

In contrast, the set of regular (non-singular) target points is defined by

$$\Omega_q^{r,\delta} \coloneqq \{x \in \Gamma \;:\; \operatorname{dist}(x,\Gamma^q) > \delta\}. \tag{5.15}$$

We say that the interaction of an integration patch Γ^q with a target point is *singular* or *regular/non-singular*, according to whether the target point lies in $\Omega_q^{s,\delta}$ or $\Omega_q^{r,\delta}$, respectively.

5.5.5 Integration algorithm for singular interactions

To evaluate (5.12) at a singular or near-singular target point $x \in \Omega_p^{s,\delta}$, we proceed as follows. First, we form the Chebyshev expansion of the density φ^q over Γ^q:

$$\varphi^q(u,v) \approx \sum_{m=0}^{N_v^q-1}\sum_{n=0}^{N_u^q-1} a_{n,m}^q T_n(u)T_m(v), \tag{5.16}$$

where, in view of the discrete orthogonality property satisfied by Chebyshev polynomials at the Fejér nodes, we have

$$a^q_{n,m} = \frac{\alpha_n \alpha_m}{N^q_u N^q_v} \sum_{j=0}^{N^q_v-1} \sum_{i=0}^{N^q_u-1} \varphi^q(u_i, v_j) T_n(u_i) T_m(v_j), \qquad \alpha_n := \begin{cases} 1, & n = 0 \\ 2, & n \neq 0 \end{cases}. \tag{5.17}$$

Replacing the density φ^q by its Chebyshev expansion (5.16), in the proposed scheme the integral (5.12) is numerically approximated by

$$(I^q\varphi)(x) \approx \int_R K^q(x,u,v) \left(\sum_{m=0}^{N^q_v-1} \sum_{n=0}^{N^q_u-1} a^q_{n,m} T_n(u) T_m(v) \right) J^q(u,v)\, du dv \tag{5.18a}$$

$$= \sum_{m=0}^{N^q_v-1} \sum_{n=0}^{N^q_u-1} a^q_{n,m} \left(\int_R K^q(x,u,v) T_n(u) T_m(v) J^q(u,v)\, du dv \right). \tag{5.18b}$$

Note that the double integral in (5.18b) does not depend on the density; it depends only on the kernel, a product of Chebyshev polynomials, and the geometry. Once this integral has been computed to the desired accuracy, the proposed method stores its value and uses it as needed.

We write the value of I^q at all target points $x_\ell \in \Omega^{s,\delta}_p$ succinctly as

$$(I^q\varphi)(x_\ell) = \sum_{m=0}^{N^q_v-1} \sum_{n=0}^{N^q_u-1} a^q_{n,m}\, \beta^{q,\ell}_{n,m}, \tag{5.19}$$

where

$$\beta^{q,\ell}_{n,m} := \int_R K^q(x_\ell,u,v) T_n(u) T_m(v) J^q(u,v)\, du dv. \tag{5.20}$$

To compute (5.20) at an evaluation point x_ℓ, we first identify its corresponding integration patch node $(\bar{u}^q_\ell, \bar{v}^q_\ell)$. If the target point x_ℓ is itself a grid point of Γ^q, then finding its node is straightforward: $x_\ell = y^q(\bar{u}^q_\ell, \bar{v}^q_\ell)$ for some point $(\bar{u}^q_\ell, \bar{v}^q_\ell)$ in the uv-plane reference domain for Γ^q. On the other hand, if $x_\ell \in \Omega^{s,\delta}_p \setminus \Gamma^q$, then we search for a Γ^q node such that

$$(\bar{u}^q_\ell, \bar{v}^q_\ell) = \operatorname*{arg\,min}_{(u,v)\in[-1,1]^2} \left\| x_\ell - y^q(u,v) \right\|. \tag{5.21}$$

As in [52], for robustness and simplicity, we solve the minimization problem (5.21) by means of the golden section search algorithm.

Next, we apply a one-dimensional change of variables to each coordinate in the uv-parameter space to construct a clustered grid around each given target node. To

this end, we consider the following one-to-one, strictly monotonically increasing, and infinitely differentiable function $w : [0, 2\pi] \to [0, 2\pi]$, with parameter $d \geq 2$ (proposed in [11, Sec. 3.5]),

$$w(\tau; d) \coloneqq 2\pi \frac{[\nu(\tau)]^d}{[\nu(\tau)]^d + [\nu(2\pi - \tau)]^d}, \qquad 0 \leq \tau \leq 2\pi, \tag{5.22}$$

where

$$\nu(\tau; d) \coloneqq \left(\frac{1}{d} - \frac{1}{2}\right)\left(\frac{\pi - \tau}{\pi}\right)^3 + \frac{1}{d}\left(\frac{\tau - \pi}{\pi}\right) + \frac{1}{2}. \tag{5.23}$$

It can be shown that w has vanishing derivatives up to order $d - 1$ at the interval endpoints. Then, the following change of variables

$$\xi_\alpha(\tau; d) \coloneqq \begin{cases} \alpha + \left(\frac{\operatorname{sgn}(\tau) - \alpha}{\pi}\right) w(\pi|\tau|; d), & \text{for } \alpha \neq \pm 1, \\ \alpha - \left(\frac{1+\alpha}{\pi}\right) w(\pi \left|\frac{\tau-1}{2}\right|; d), & \text{for } \alpha = 1, \\ \alpha + \left(\frac{1-\alpha}{\pi}\right) w(\pi \left|\frac{\tau+1}{2}\right|; d), & \text{for } \alpha = -1, \end{cases} \tag{5.24}$$

has the effect of clustering points around α. Fejér's rule applied to the integral (5.20), transformed using the change of variables (5.24), yields the approximation

$$\beta_{n,m}^{q,\ell} \approx \sum_{j=0}^{N_\beta - 1} \sum_{i=0}^{N_\beta - 1} K^q(x_\ell, u_i^{q,\ell}, v_j^{q,\ell}) T_n(u_i^{q,\ell}) T_m(v_j^{q,\ell}) J^q(u_i^{q,\ell}, v_j^{q,\ell})\, w_i^{u,q,\ell} w_j^{v,q,\ell}, \tag{5.25}$$

where

$$u_i^{q,\ell} = \xi_{\bar{u}_\ell^q}(s_i; d), \quad w_i^{u,q,\ell} = \frac{d\xi_{\bar{u}_\ell^q}}{d\tau}(s_i; d)\, w_i, \tag{5.26}$$

$$v_j^{q,\ell} = \xi_{\bar{v}_\ell^q}(s_j; d), \quad w_j^{v,q,\ell} = \frac{d\xi_{\bar{v}_\ell^q}}{d\tau}(s_j; d)\, w_j, \tag{5.27}$$

for $i,\ j = 0, \ldots, N_\beta - 1$. To avoid division by zero, we set the kernel K^q to zero at integration points where the distance to the target point is less than some prescribed tolerance, usually on the order of 10^{-14}.

5.5.6 Integration algorithm for non-singular interactions

Together with the singular integration method discussed in the previous subsection, the (non-accelerated) high-order solver [52] evaluates the integral operator (5.12) at all regular target points $x_\ell \in \Omega_q^{r,\delta}$ simply by means of Fejér's first quadrature rule:

$$(I^q \varphi)(x_\ell) \approx \sum_{j=0}^{N_v^q - 1} \sum_{i=0}^{N_u^q - 1} K^q(x_\ell, u_i, v_j) \varphi^q(u_i, v_j) J^q(u_i, v_j)\, w_i w_j. \tag{5.28}$$

It is not difficult to show that, asymptotically, the regular interactions dominate the integral operator computation (see [52, Sec. 4.4]). Evaluating all non-singular interactions using (5.28) leads to an algorithm with complexity $O(N^2)$ operations. Clearly, for acoustically-large problems this quadratic computational complexity becomes prohibitively expensive. To deal with this difficulty, we use instead the above introduced IFGF acceleration method to accelerate the evaluation which is described in the following section. As indicated in Section 3.7 for simple discrete operators, and is confirmed for full scattering problems by the following numerical results, the IFGF method leads to an overall algorithm that runs at computing cost of $O(N \log N)$ operations.

5.5.7 IFGF method for the combined-layer formulation

The IFGF approach was described in Chapter 3 as an algorithm for accelerated evaluation of the sum (3.1) for a Green function such as (1.8). Various considerations are necessary to apply the IFGF method to discrete forms of the integral operators $\mathcal{S}_\kappa$ and $\mathcal{D}_\kappa$ on the left-hand side of (5.7), as discussed in what follows. On one hand, the singular interactions considered in Section 5.5.5 and certain sets of neighboring points in the $d = D$ IFGF level, which are non-singular for the rectangular-polar method but which may not be evaluated by means of the IFGF interpolation strategy are handled independently of the IFGF accelerator (cf. Section 3.6), by means of the algorithms described in Sections 5.5.5 and 5.5.6, respectively.

The non-neighboring IFGF interactions in the discretizations of these operators—i.e., the contributions that involve pairs of discretization points $x = x_\ell$ and $y = x_m$ that are sufficiently far from each other, as indicated in equation (5.15) and associated text—, on the other hand, lead to sums which can be treated by means of the IFGF accelerator. However, it must be noted that, while the non-neighboring contributions arising from the operator $\mathcal{S}_\kappa$ are precisely of the form (3.1), the corresponding non-neighboring contributions associated with $\mathcal{D}_\kappa$ are somewhat different in character—as evidenced, in particular, by their asymptotic $O(1/|x-y|^2)$ growth as $|x-y| \to 0$. This difference can be tackled in two different manners. In a first approach, two separate IFGF accelerators are used, one as described above for the single-layer potential and a separate one, based on the use of a different centered factor, namely $G(x, x^d_{\mathbf{k}})/|x - x^d_{\mathbf{k}}|$, for the double layer potential. A second approach, on the other hand, combines the kernels of the single and double layer potentials and uses the centered factor $G(x, x^d_{\mathbf{k}})$ for the combined kernel. While rigorously accounting

for the Green function singularity and maintaining accuracy for arbitrarily small values of the level-D box size H_D, the first approach doubles the acceleration cost. It is therefore valuable to consider the ranges of values of H_D for which the second approach remains unaffected by the somewhat unresolved double-layer Green function singularity. Our numerical experiments indicate that the double-layer singularity errors in the second approach are negligible for $H_D \geq 0.5\lambda$. Thus, for

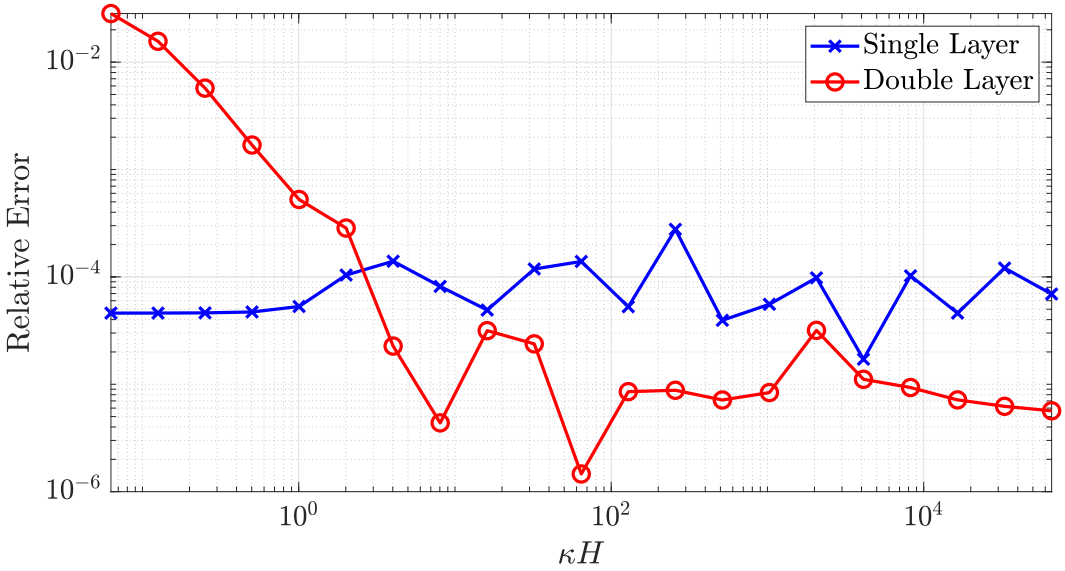


Figure 5.3: Same as Figure 3.4, but only showing the interpolation strategy in the s variable, although, for the single layer potential and the double layer potential.

the types of structures considered in this thesis, which do not contain significant sub-wavelength geometric features, the second approach is advantageous, as it enjoys the reduced computational cost while preserving accuracy. In a more general context, an adaptive approach would be used (cf. the second paragraph of Section 5.5.9) which incorporates the first approach for the portion of the IFGF octree containing boxes of size $H_d < 0.5\lambda$ and the second approach for the remainder the octree. Such an extension is beyond the scope of this thesis, and is left for future work.

After solving (5.7) for the density φ, the far-field pattern u^∞ can be obtained from

$$u^\infty(\hat{x}) = \frac{1}{4\pi} \int_\Gamma \left\{ \frac{\partial}{\partial \nu(y)} e^{-\imath\kappa\hat{x}\cdot y} - \imath\gamma e^{-\imath\kappa\hat{x}\cdot y} \right\} \varphi(y)\, dS(y), \quad \hat{x} \in \mathbb{S}^2, \tag{5.29}$$

where $\mathbb{S}^2$ denotes the unit sphere and Γ is the scatterer's boundary. The far-field is computed over a uniformly-spaced unit spherical grid

$$\mathbb{S}^2_N := \left\{ (\phi_m, \theta_n) \in [0,\pi] \times [0,2\pi] \ : \ 1 \leq m \leq N_\phi,\ 1 \leq n \leq N_\theta \right\}, \tag{5.30}$$

with $\phi_m = (m-1)\Delta\phi$, $\theta_n = (n-1)\Delta\theta$ and where the spacings are defined as $\Delta\phi = \pi/(N_\phi - 1)$ and $\Delta\theta = 2\pi/(N_\theta - 1)$, respectively; specific values of N_ϕ and N_θ are given in each example's subsection. Given the exact (or reference) far-field modulus $|u^\infty|$ and an approximate far-field modulus $|\tilde{u}^\infty|$, the maximum far-field relative error ε_{far} over $\mathbb{S}^2_N$ given by

$$\varepsilon_{far} = \max_{(m,n)\in\mathbb{S}^2_N} \left\{ \frac{||u^\infty_{m,n}| - |\tilde{u}^\infty_{m,n}||}{||u^\infty_{m,n}||} \right\} \tag{5.31}$$

is reported in each case.

Similarly, using the solution φ in the combined-layer representation (5.6), we evaluate and display the scattered field u^s over near-field planes that are parallel to the xy-, xz-, or yz-planes. For example, we evaluate fields (incident, scattered, and total) at every point of a uniformly-spaced two-dimensional xy-planar grid $\mathbb{P}^{xy}_N(z_0)$ at $z = z_0$ defined by

$$\begin{aligned} \mathbb{P}^{xy}_N(z_0) \coloneqq \big\{ (x_m, y_m, z) \in [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times \{z_0\} \ : \\ 1 \le m \le N_x,\ 1 \le n \le N_y \big\}, \end{aligned} \tag{5.32}$$

where the grid points are given by $x_m = (m-1)\Delta x$, $y_n = (n-1)\Delta y$ and the grid spacings are $\Delta x = (x_{max} - x_{min})/(N_x - 1)$ and $\Delta y = (y_{max} - y_{min})/(N_y - 1)$. Near-field planar grids parallel to the xz- and yz-plane are defined analogously. Denoting the exact (or reference) and approximate modulus of the total field at each point of $\mathbb{P}^{xy}_N(z_0)$ by $v_{m,n}$ $(= |u^s_{m,n} + u^i_{m,n}|)$ and $\tilde{v}_{m,n}$ $(= |\tilde{u}^s_{m,n} + u^i_{m,n}|)$, respectively, we compute the near-field (total magnitude) relative error ε_{near} over $\mathbb{P}^{xy}_N(z_0)$ as

$$\varepsilon_{near} = \max_{(m,n)\in\mathbb{P}^{xy}_N(z_0)} \left\{ \frac{|v_{m,n} - \tilde{v}_{m,n}|}{|v_{m,n}|} \right\}. \tag{5.33}$$

The numerical results presented in what follows were obtained using a single processor on one of the *Wavefield* compute nodes presented in Section 5.1.2, i.e., using 28 cores. Solutions to the complex-coefficient linear systems that arise from discretizations of the boundary integral equation (5.7) were obtained with a complex-arithmetic GMRES iterative solver (see Section 2.1). Following [37], we set the combined-layer equation (5.7) coupling parameter $\gamma = \max\{3, A/\lambda\}$, where A is the diameter of the scatterer; computational results indicate that, to reach a given residual tolerance, this value reduces the number of GMRES iterations by a factor of $5 - 10$ compared with $\gamma = \kappa$. Plots were generated using the visualization software VisIt [83].

5.5.8 Scattering by a sphere

We consider plane wave scattering by a sphere of various acoustical sizes. For a sound-soft acoustic sphere, the well-known closed-form far-field expression is used to compute relative errors [84]. Table 5.9 summarizes the accuracy and efficiency of the IFGF-accelerated solver and non-accelerated solver for a sphere of diameter ranging from 4 to 128 wavelengths. For each problem, the number of IFGF levels is selected so that the finest-level IFGF box side length is approximately 0.5λ. All computations are performed using a GMRES residual tolerance set to 10^{-4}. We report the total number of unknowns, the size of the sphere in wavelengths, the time required to compute one GMRES iteration as well as the total number of iterations required to achieve the prescribed residual, and the far-field relative error. far-field relative errors are computed over the spherical grid (5.30) with $(N_\phi, N_\theta) = (200, 200)$. Table 5.9 shows that the time per iteration required by the

N	d	IFGF levels	T (1 iter.)	Tot. iter.	ε_{far}
13,824	4λ	4	0.2 s	12	$1 \cdot 10^{-4}$
55,296	8λ	5	1.0 s	14	$1 \cdot 10^{-4}$
221,184	16λ	6	4.6 s	14	$6 \cdot 10^{-5}$
884,736	32λ	7	19.4 s	16	$3 \cdot 10^{-5}$
3,538,994	64λ	8	83.1 s	18	$6 \cdot 10^{-5}$
14,155,776	128λ	9	443.2 s	21	$4 \cdot 10^{-4}$

Table 5.9: IFGF-accelerated solver for acoustic scattering by a sphere of acoustical sizes ranging from 4 to 128 wavelengths. The table summarizes the total number of surface unknowns, sphere size in wavelengths, maximum number of IFGF levels, time required to compute one GMRES iteration, total number of iterations, and far-field relative error ε_{far}. In all cases the GMRES residual tolerance was set to 10^{-4}.

non-accelerated algorithm grows by a factor of around $14.8 - 15.7$ as the number of points per dimension in each surface patch is doubled (so that the overall number of unknowns is quadrupled), which is consistent with the expected quadratic complexity of the algorithm. For an N-point surface discretization, the IFGF-based solver, on the other hand, the computing costs scale like $O(N \log N)$, as shown in Figure 5.4 where we plot the accelerated solver compute time (in s) versus N. The reduced complexity of the IFGF-based algorithm has a significant impact on computing times. At 128 wavelengths, the non-accelerated solver takes more than 1000 times longer than the accelerated method for each GMRES iteration; for larger

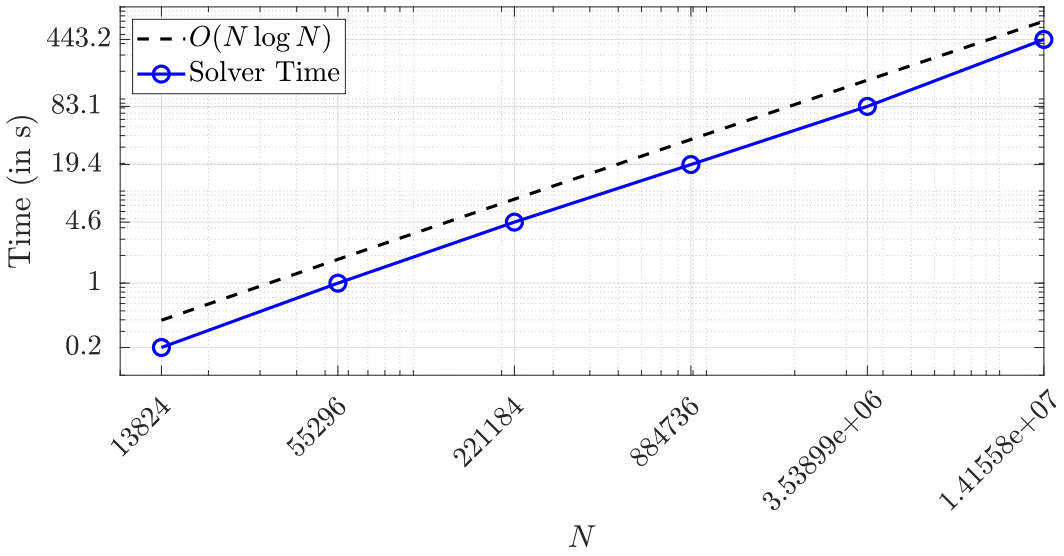


Figure 5.4: IFGF-accelerated solver compute time (in seconds) versus total number of discretization points N, plotted with line-circle markers, for acoustic scattering by a sphere. For reference, we also plot a graph of $CN \log N$, $C = 0.3 \cdot 10^{-5}$, using a dashed line.

problems, the difference in compute times grows as expected from the complexity estimates for the two methods. Note that the total number of GMRES iterations necessary to satisfy the residual tolerance is the same for both the non-accelerated and accelerated solvers. Additionally, the errors for both algorithms are comparable: the non-accelerated solver yields solutions for the 4, 8 and 16 wavelength problems with an average relative error of $1.1 \cdot 10^{-4}$, while errors obtained with the accelerated method average $1.3 \cdot 10^{-4}$ across the entire 4 to 128 wavelength range.

5.5.9 Scattering by a submarine geometry

In this section, we present acoustic scattering simulations for a realistic submarine configuration of up to 80 wavelengths in acoustical size. Due to its importance in detection and tracking applications, methods for efficient and accurate scattering simulations are the subject of ongoing research [85–91]. The submarine model used in subsequent simulations, which is comprised of the main hull, sail, diving planes, rudders, and a five-blade propeller, is depicted in Figure 5.5. The complete submarine geometry is contained in the bounding region $[-3.2, 3.2] \times [-1.9, 2.8] \times [-19.2, 10.9]$. Figures 5.5(b) and 5.5(c) show a surface mesh of $4,560$ patches, each of which is represented by 6×6 points.

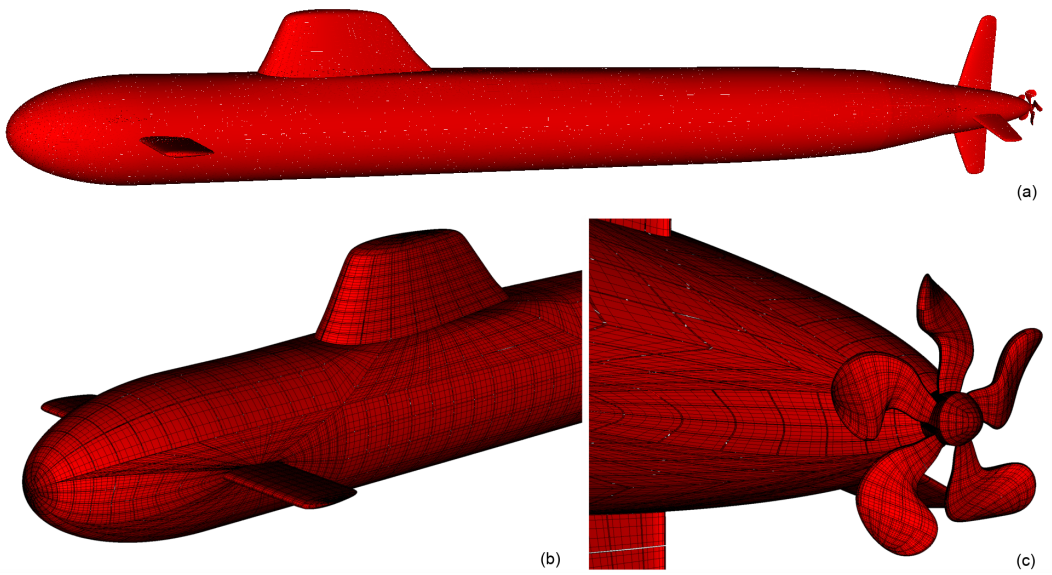


Figure 5.5: Submarine model and surface mesh with a total of 164, 160 points. The submarine hull is aligned with the z-axis and the sail is parallel to the $+y$-axis; the front of the vessel points in the $+z$-direction.

We consider plane wave scattering for two cases: a) head-on incidence and b) oblique incidence. The incident field is a plane wave u^i that travels along the wave direction $\hat{k}$ and is given by

$$u^i(x) = e^{\iota\kappa\hat{k}\cdot x}, \qquad \hat{k} = \begin{pmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \\ \cos\phi \end{pmatrix}, \tag{5.34}$$

where the position vector $x = (x_1, x_2, x_3)$, $\kappa > 0$ is a given wavenumber, and $(\theta, \phi) \in [0, 2\pi) \times [0, \pi]$. Since the bow of the submarine points in the $+z$-direction, "head-on" incidence corresponds to $(\theta, \phi) = (0, \pi)$ in (5.34). For the oblique incidence case, we set $(\theta, \phi) = (0, 5\pi/4)$.

To verify the accuracy of the IFGF-accelerated solver in the present case, we conducted convergence studies for the submarine structure at 10λ, 20λ, and 40λ in acoustical size (measured from the bow to the propeller cap). In all cases the number of IFGF levels was chosen so that the side length of the smallest, finest-level, boxes is around 0.8λ. The GMRES residual tolerance was set to 10^{-3} in all cases. We start with a 10λ vessel whose geometry is represented by $1,140$ surface patches, each of which has 6×6 points. As the size of the problem is doubled, the geometry is partitioned from the previous size so that every patch is split into four subpatches while keeping the same number of points per patch. Thus, for example, the 20λ

problem uses four times as many surface points as the 10λ case. This is admittedly a sub-optimal strategy, in this case, (as the smaller patches on the propeller, rudders and diving planes which already fully discretize the wavelength do not require additional partitioning), which, however, simplifies the code implementation. Additionally, this distribution of surface points makes sub-optimal use of the present version of the IFGF algorithm. As indicated in Section 3.6, the IFGF method can be extended to incorporate a box octree algorithm that adaptively partitions a geometry until each box contains a (small) prescribed number of points, thus eliminating this difficulty. While such an addition is left for future work, as demonstrated in Table 5.10, even the simple uniform-partition IFGF algorithm we use in this thesis is sufficient to simulate scattering by a realistic submarine geometry for up to 80 wavelengths in size with several digits of accuracy and using only modest computational resources. For example, the $656,640$ unknowns, 40λ run for head-on incidence, required a computing time of 313 seconds per iteration and a total of 78 iterations. The fully adaptive version of the IFGF algorithm, which, as mentioned above, is not pursued in this thesis, should yield for the submarine geometry computing times consistent with those shown in Tables (5.9), (5.11) and (5.12) for the sphere and nacelle geometries. Near-field relative errors for front (head-on) and

N	d	IFGF levels	Front ε_{near}	Oblique ε_{near}
41,040	10λ	5	$2 \cdot 10^{-4}$	$7 \cdot 10^{-4}$
164,160	20λ	6	$2 \cdot 10^{-4}$	$6 \cdot 10^{-4}$
656,640	40λ	7	$2 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
2,626,560	80λ	8	(est.) $2 \cdot 10^{-4}$	(est.) $5 \cdot 10^{-4}$

Table 5.10: Convergence study of IFGF-accelerated acoustic solver for the submarine geometry, with acoustical sizes ranging from 10 to 40 wavelengths, and the front and oblique wave incidence. In all cases, the residual tolerance was set to 10^{-3}.

oblique plane wave incidence are shown in Table 5.10. For each problem, we estimate ε_{near} over $\mathbb{P}_N^{xy}(z_0)$, where $[x_{min}, x_{max}] \times [y_{min}, y_{max}] = [-12, 12]^2$, $z_0 = -25$ and $N_x = N_y = 260$, using (5.33) with a reference solution obtained with the same number of surface patches as the target discretization but using 8×8 points per patch and a residual tolerance of 10^{-5}. (Thus, the reference solution uses nearly twice as many discretization points and it satisfies a more stringent convergence condition.) The numerical results indicate that the solution accuracy is consistent for both front and oblique incidence and for all acoustical sizes considered. For front and oblique

incidence, the relative errors for the 10λ, 20λ, and 40λ problems achieve an average accuracy of $2.1 \cdot 10^{-4}$ and $4.8 \cdot 10^{-4}$, respectively, and we use these values to estimate the expected relative errors in the 80-wavelength case.

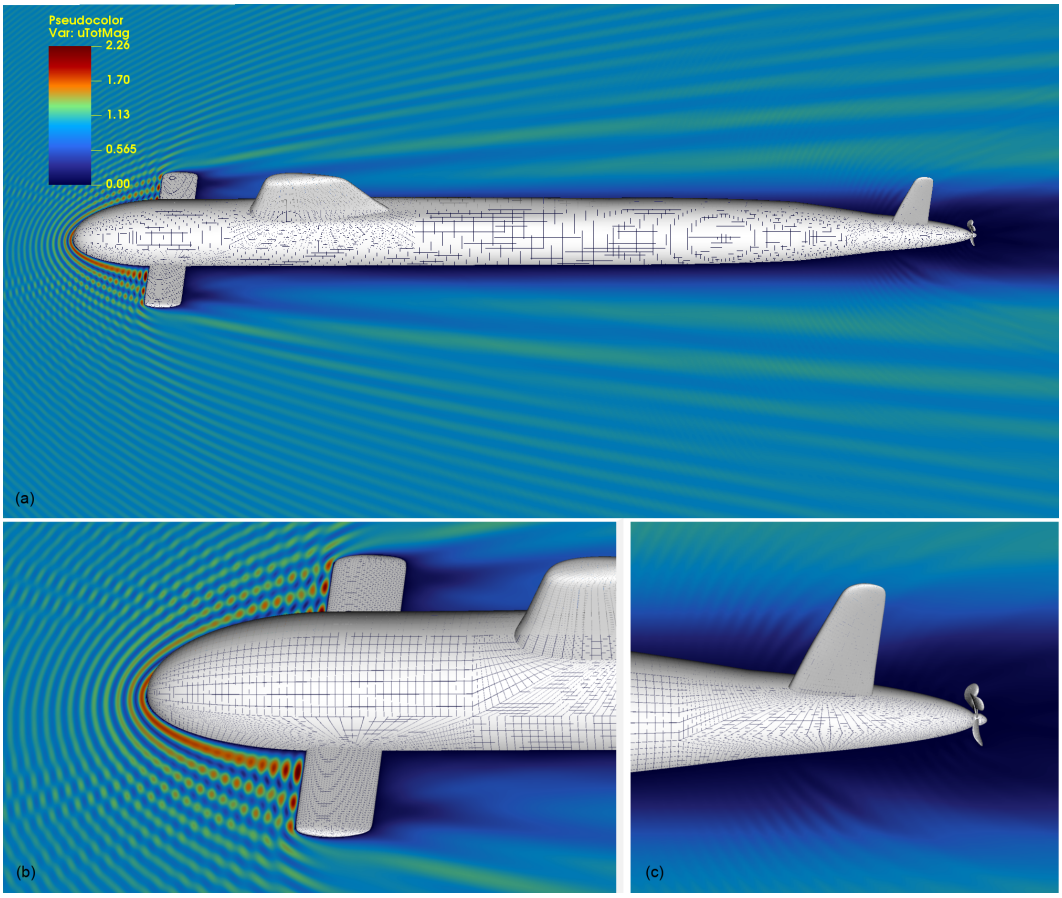


Figure 5.6: Total field magnitude $|u(x)| = |u^i(x) + u^s(x)|$ pseudocolor plots for an 80-wavelength submarine. The field is plotted over a uniform grid of 1040×1760 points for $(x, z) \in [-12, 12] \times [-25, 15]$. In this case, the incident plane wave impinges on the vessel head-on, which corresponds to the wave direction $\hat{k}$ in (5.34) with $(\theta, \phi) = (0, \pi)$.

In Figure 5.6, we present pseudocolor near-field plots of the total field magnitude $|u(x)| = |u^i(x) + u^s(x)|$ for front plane wave incidence for an 80-wavelength submarine. The field is plotted over a uniform 1040×1760 point planar grid $\mathbb{P}_N^{xz}(y_0)$ for $(x, z) \in [-12, 12] \times [-25, 15]$ and $y_0 = 0$. The incident plane wave impinges on the vessel head-on and we see in Figures 5.6(a) and 5.6(b) that the strongest interaction occurs around the bow and diving planes (also known as hydroplanes) of the ship. Shadow regions are visible immediately behind the hydroplanes as well as along the hull, particularly in the aft of the ship where the body tapers. Figure 5.6(c) shows

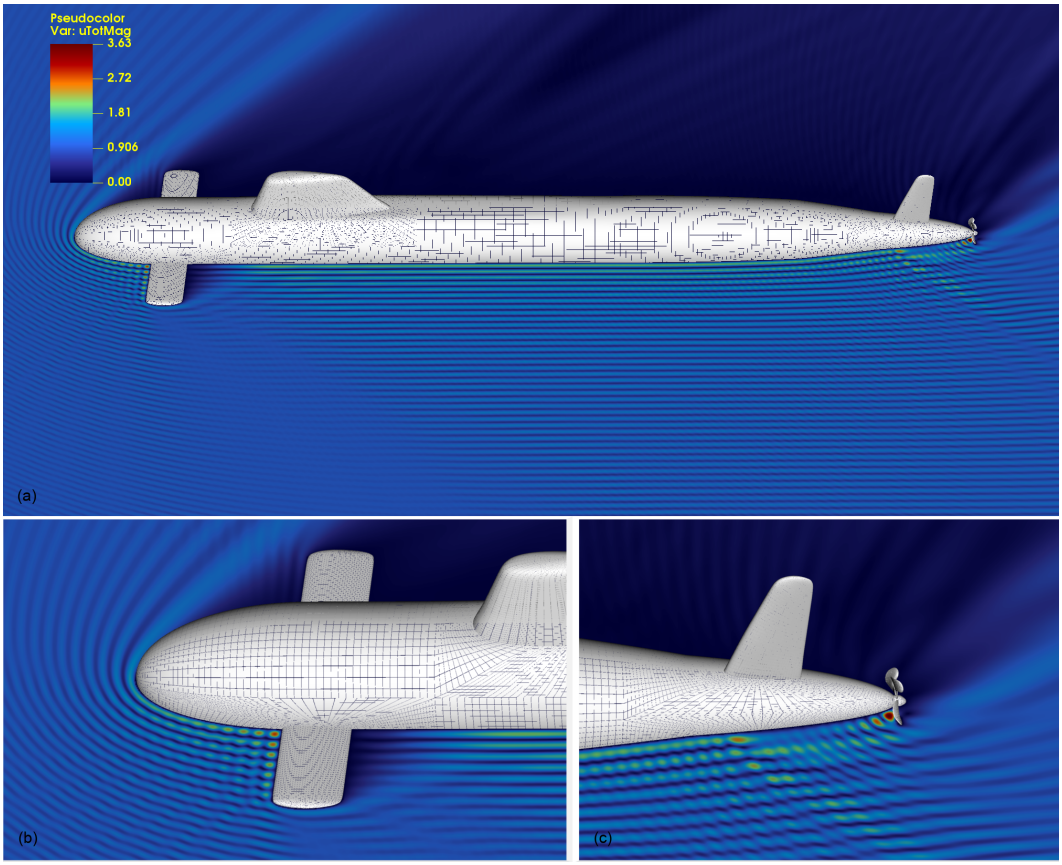


Figure 5.7: Total field magnitude $|u(x)| = |u^i(x) + u^s(x)|$ pseudo-color plots for an 80-wavelength submarine. The field is plotted over a uniform grid of 1040×1760 points for $(x, z) \in [-12, 12] \times [-25, 15]$. In this case, the incident plane wave impinges on the vessel at an oblique angle, which corresponds to the wave direction $\hat{k}$ in (5.34) with $(\theta, \phi) = (0, 5\pi/4)$.

that the wider sections of the ship obstruct the propeller from most incoming waves and, as a consequence, there is minimal interaction in this region.

Figure 5.7 shows near-field pseudocolor plots for the same 80-wavelength submarine but this time for oblique plane wave incidence. The total field magnitude is plotted over the uniform grid $\mathbb{P}_N^{xz}(y_0)$ described in the previous paragraph. In this case, the wave interaction is markedly different. We see the expected shadow region in the opposite side of the incoming wave but there is now clear evidence of wave interaction between the hull and diving planes as well as around the rudders and propeller. In addition to multiple scattering, the close-up views of Figures 5.7(b) and 5.7(c) show the formation of bright spots near the junction of the left hydroplane and hull and in the vicinity of the propeller.

5.5.10 Scattering by an aircraft nacelle

The simulation of aircraft engine noise has been the subject of intense research for the past several decades due to its importance in civil aviation applications [92–96]. In this section, we present simulations of sound propagation in and around the turbofan engine nacelle model shown in Figure 5.8. According to the nacelle wall liner case study [97], under typical operating conditions, engine nacelle noise occurs in the 125 – 5650 Hz frequency range. For a typical airliner engine that is around 5 m long, these frequencies correspond to acoustical sizes between 2 and 82 wavelengths. The engine nacelle geometry used in the simulations that follow is

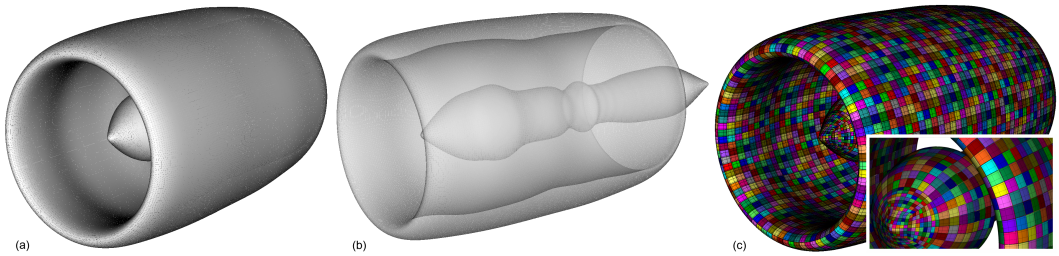


Figure 5.8: Left panel: Aircraft engine nacelle model. Center panel: translucent view of the geometry where the center shaft is visible. Right panel: 8,576-surface-patch discretization with 6×6 points per patch (for clarity, only every other mesh point is plotted).

depicted in Figures 5.8(a) and 5.8(b); it is comprised of an outer housing and a center shaft. The entire two-piece nacelle structure is contained inside the bounding region $[-1.5, 1.5] \times [-1.5, 1.5] \times [-3.27, 3.27]$. The center shaft is aligned with the z-axis, with the tip of the shaft pointing towards the positive direction. A discretization with 8,576 surface patches with 6×6 points per patch is shown in Figure 5.8(c); for future reference, note that the inset image shows that the mesh is not rotationally symmetric near the tip of the shaft.

Two types of incident fields are used in simulations: a) a plane wave that travels towards the $-z$-axis, so that it impinges on the nacelle head-on and b) a set of eight point sources placed inside the housing around the center shaft. As in the submarine example, the plane wave incident field is given by (5.34) with $(\theta, \phi) = (0, \pi)$. The incident field b), on the other hand, serves as a simple model for fan noise generation inside the nacelle and is given by

$$\begin{aligned} u^i(x) &= \sum_{j=1}^{8} \frac{e^{\iota\kappa|x-x^j|}}{|x-x^j|}, \quad \text{with point source locations} \\ x^j &= (x_1^j, x_2^j, x_3^j) = (\cos\alpha_j, \sin\alpha_j, 2), \end{aligned} \tag{5.35}$$

where $\alpha_j = (j-1)\Delta\alpha + \pi/8$, for $j = 1, \ldots, 8$, and $\Delta\alpha = \pi/4$.

N	d	IFGF levels	T (1 iter.)	Tot. iter.	ε_{near}
77,184	10.2λ	6	2.0 s	33	$2 \cdot 10^{-3}$
308,736	20.5λ	7	8.7 s	47	$2 \cdot 10^{-3}$
1,234,944	40.9λ	8	40.4 s	55	$7 \cdot 10^{-4}$
4,939,776	81.8λ	9	176.4 s	65	(est.) $2 \cdot 10^{-3}$

Table 5.11: Convergence study of IFGF-accelerated acoustic solver for a nacelle geometry of 10.2, 20.5 and 40.9 wavelengths for plane wave scattering. (The table also includes data for an 81.8λ nacelle, but in this case, the near-field relative error is estimated using the average relative errors of the three previous problems.) The table summarizes the total number of surface unknowns, nacelle size in wavelengths, maximum number of IFGF levels, time required to compute one GMRES iteration, total number of iterations, and near-field relative error ε_{near}. In all cases, the GMRES residual tolerance was set to 10^{-3}.

N	d	IFGF levels	T (1 iter.)	Tot. iter.	ε_{near}
77,184	10.2λ	6	2.0 s	39	$4 \cdot 10^{-3}$
308,736	20.5λ	7	8.7 s	59	$4 \cdot 10^{-3}$
1,234,944	40.9λ	8	40.4 s	115	$2 \cdot 10^{-3}$
4,939,776	81.8λ	9	176.4 s	219	(est.) $4 \cdot 10^{-3}$

Table 5.12: Same as Table 5.11, but for point source scattering.

Tables 5.11 and 5.12 tabulate the results of a convergence study for both plane wave and point source incidence for a nacelle of 10.2, 20.5 and 40.9 wavelengths in size, respectively. We also include results for an 81.8-wavelength nacelle. The number of IFGF levels is selected so that the finest-level IFGF box side length is approximately 0.6λ in all cases. All computations were performed with a GMRES residual tolerance equal to 10^{-3}. The total near-field magnitude relative error ε_{near} was estimated over a near-field planar grid $\mathbb{P}_N^{xy}(z_0)$, where $[x_{min}, x_{max}] \times [y_{min}, y_{max}] = [-4, 4]^2$, $z_0 = -5$ and $N_x = N_y = 400$, by computing (5.33) with a reference solution obtained with the same number of surface patches as the target discretization but using 8×8 points per patch and a residual tolerance of 10^{-5}. In addition to the near-field relative error, Tables 5.11 and 5.12 also include the total number of unknowns, the size of the nacelle in wavelengths, the time required to compute one GMRES iteration and the total number of GMRES iterations required to satisfy the 10^{-3} residual tolerance. Thus, as the problem size increases from

10.2λ to 20.5λ, 20.5λ to 40.9λ, and 40.9λ to 81.8λ, and the number of unknowns is quadrupled in each case, the computing cost per iteration increases by a factor of only 4.4, 4.6 and 4.4, respectively (which is consistent with an $O(N \log N)$ complexity), and not the 16-fold cost increase per wavelength doubling that would result from a non-accelerated algorithm with quadratic complexity. This scaling of the IFGF-accelerated combined-layer solver is consistent with the IFGF method computations presented throughout this Chapter, which did not include singular local interactions, and suggests that the partitioning and discretization of the geometry makes optimal use of the IFGF algorithm. The results also indicate that the discretization and 10^{-3} residual tolerance is sufficient to produce solutions for the 10.2, 20.5 and 40.9 wavelength cases with an average error of $1.5 \cdot 10^{-3}$ for plane wave scattering and $3.5 \cdot 10^{-3}$ for point source scattering. The average relative error values are used to estimate the accuracy of the 81.8λ simulation, which also converged to the same GMRES tolerance as the smaller problems. Note that, as reported in Tables 5.11 and 5.12, the number of iterations required for convergence increases by only $8 - 14$ iterations. The total near-field magnitude $|u(x)| = |u^i(x) + u^s(x)|$

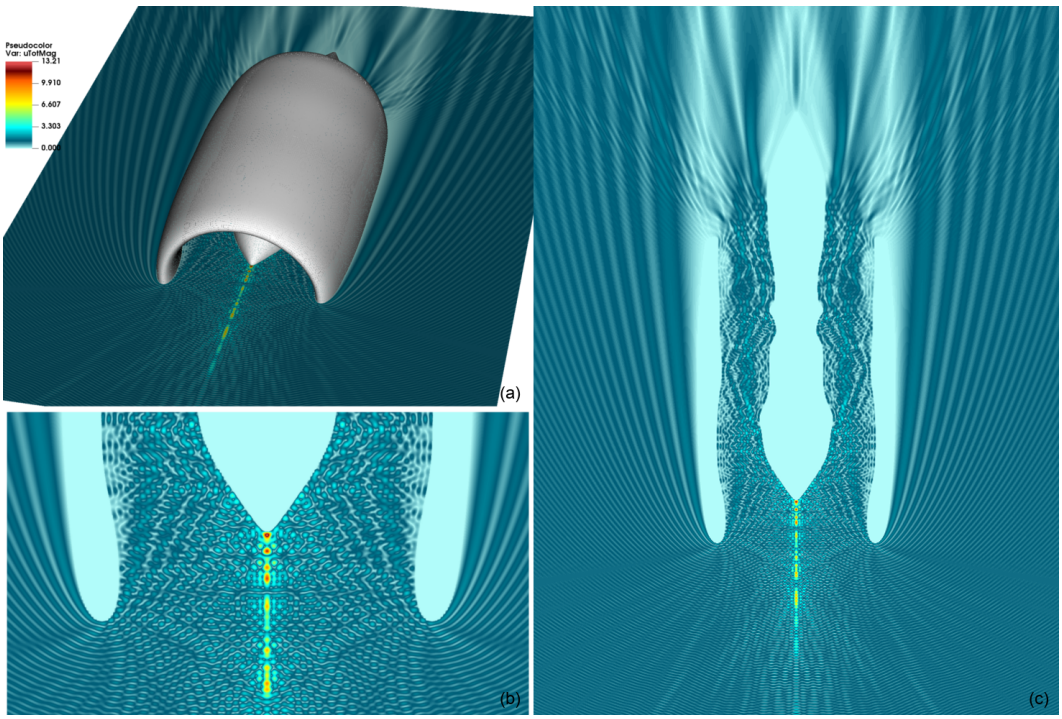


Figure 5.9: Total field magnitude $|u(x)| = |u^i(x) + u^s(x)|$ pseudo-color plots for an 82-wavelength aircraft nacelle. The field is plotted over a uniform grid of 2800×4000 points for $(x, z) \in [-4, 4] \times [-5, 6]$. The nacelle is aligned with the z-axis and the front points towards the $+z$-direction. The incident plane wave impinges on the geometry head-on and travels towards the negative z-axis.

for the 81.8-wavelength plane wave scattering case is displayed in Figure 5.9. The field magnitude is plotted over the xz-planar grid $\mathbb{P}_N^{xz}(y_0)$ (recall the planar grid definition (5.32)), where $[x_{min}, x_{max}] \times [z_{min}, z_{max}] = [-4, 4] \times [-5, 6]$, $y_0 = 0$, with $N_x = 2800$ and $N_z = 4000$. Along most of the exterior circumference of the nacelle housing, the total field forms a relatively uniform stratified pattern. In other regions, intricate multiple-scattering patterns develop, particularly in the region around the intake and throughout the inside of the nacelle. For a closer examination, Figures 5.9(b) and 5.9(c) display top views of the field but with the scattering surfaces removed. It is evident that the strongest reflection occurs directly in front of the tip of the nacelle shaft. Note the symmetry in the detail of the near-field shown in Figure 5.9(b), which results in spite of the lack of symmetry in the geometry discretization illustrated in the inset in Figure 5.8(c).

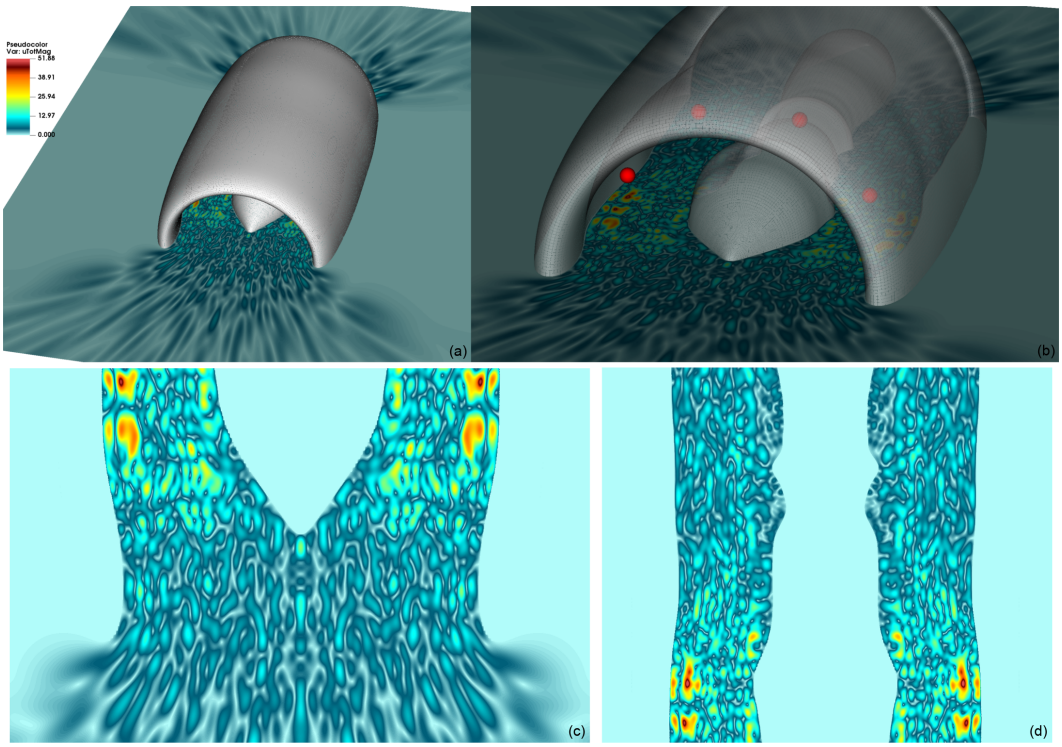


Figure 5.10: Total field magnitude $|u(x)| = |u^i(x) + u^s(x)|$ pseudocolor plots for an 82-wavelength aircraft nacelle. The field is plotted over a uniform grid of 2800×4000 points for $(x, z) \in [-4, 4] \times [-5, 6]$. The nacelle is aligned with the z-axis and the front points towards the $+z$-direction. The incident field is given by the sum (5.35) of eight point sources within the nacelle around the center shaft, four of which are shown as small red spheres in panel (b).

Near-fields for the eight-point source, 81.8-wavelength, incident field are displayed in Figure 5.10. The total field magnitude is plotted over the same xz-planar grid

$\mathbb{P}_N^{xz}(y_0)$ used for the plane wave scattering case. In Figure 5.10(a), the point-source generated fields can be seen to scatter and exit the front inlet and rear exhaust. The close-up view in Figure 5.10(b) highlights the location of four of the eight point sources, drawn as red spheres for emphasis; the remaining four sources are obstructed from view by the near-field plane. In Figures 5.10(c) and 5.10(d), the geometry is removed so we can examine the field interaction within the scatterer in greater detail. Both images exhibit complex multiple scattering and a high degree of symmetry throughout the interior of the structure and in the regions outside that surround the nacelle assembly. In contrast to the plane wave scattering case, where the incident wave travels mostly parallel to the housing and shaft, placing sources between the shaft and nacelle walls guarantees that most waves scatter multiple times before exiting the geometry.

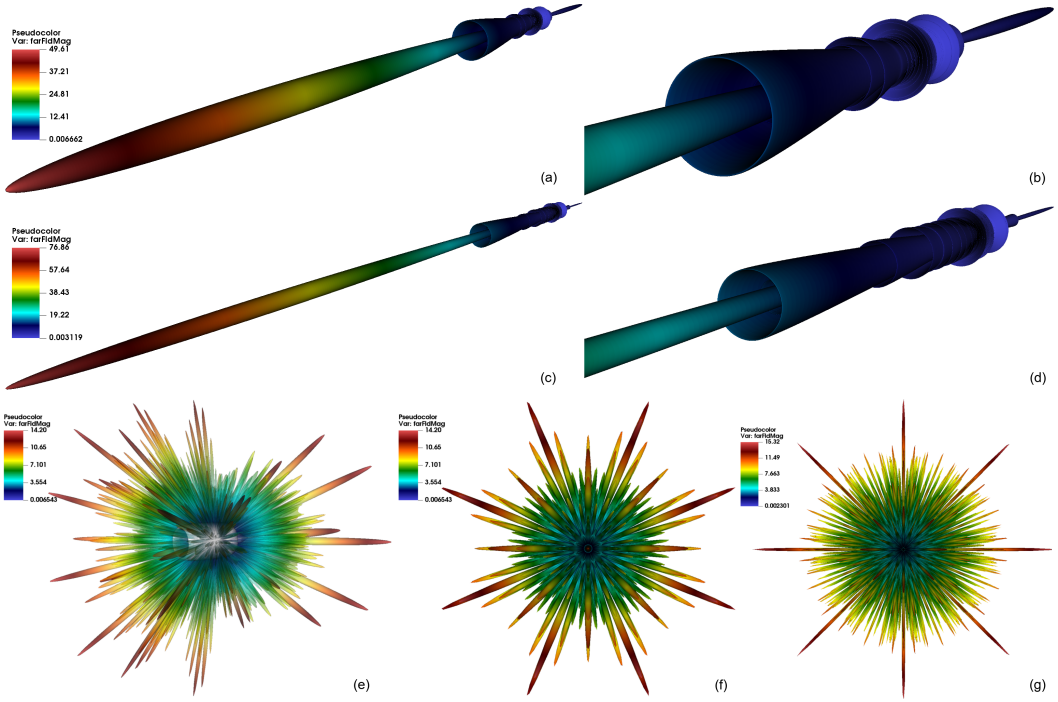


Figure 5.11: Far-field magnitudes for the nacelle geometry under plane wave and eight point-source incident fields. Panels (a) and (b) present the far-field for a 40.9λ plane wave and panels (c) and (d) present the far-field for an 81.8-wavelength plane wave. Panels (e)-(g) display far-fields for the eight-point source incident field defined in (5.35), for 40.9λ in panels (e) and (f) and for 81.8λ in panel (g).

The far-field magnitudes are shown in Figure 5.11 for both plane wave and point source incident fields. Figures 5.11(a) and 5.11(b) present the far-field for a 40.9λ plane wave, while Figures 5.11(c) and 5.11(d) present the far-field for an 81.8-wavelength plane wave. Using (5.29), the far-field $\tilde{u}^\infty$ is computed over $\mathbb{S}_N^2$ (recall

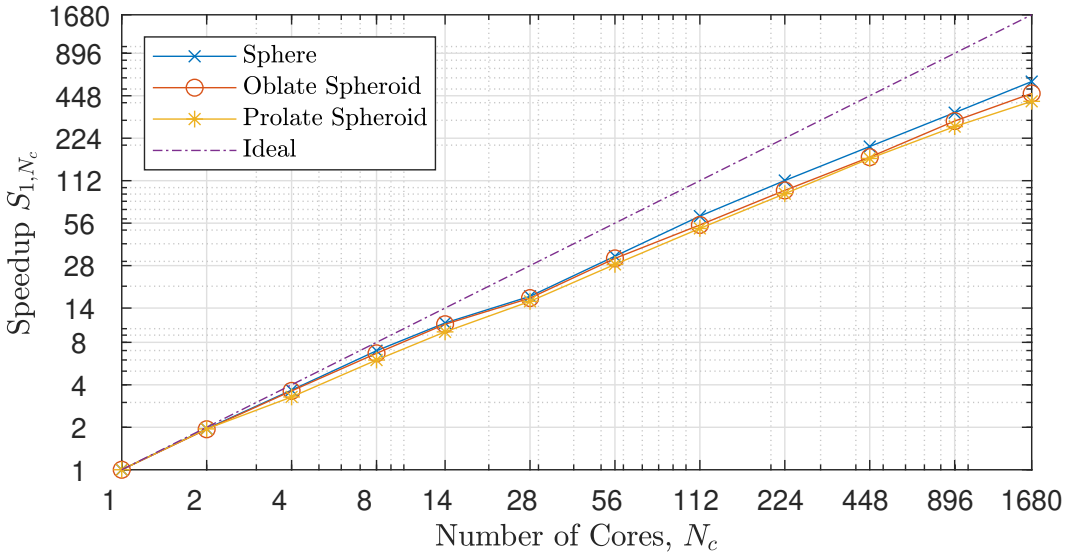


Figure 5.12: Measured speedup S_{1,N_c} (vertical axis) versus number of cores N_c (horizontal axis) in a strong scaling test transitioning from 1 core to 1,680 cores (= 30 compute nodes) for three geometries: a sphere of size 128 wavelengths (blue), an oblate spheroid of size 128 wavelengths (red), and prolate spheroid of size 256 wavelengths (yellow). The dash-dotted purple line indicates the theoretical perfect speedup.

the spherical grid definition (5.30)) with $(N_\phi, N_\theta) = (2000, 600)$ in the 40.9λ case and $(N_\phi, N_\theta) = (3200, 800)$ for the 81.8 wavelength plane wave. More points are used at higher frequencies to resolve the far field lobes that are visible in Figures 5.11(b) and 5.11(d). The far-field plots for both 40.9λ and 81.8λ plane wave scattering once again show that most of the wave reflection occurs in the region directly in front of the nacelle intake; note that this reflection intensifies as the wavelength decreases. The maximum magnitude of the far-field increases by a factor of approximately 1.5 for the 81.8λ wave compared with the 40.9λ case. Figures 5.11(e-g) show the far-field magnitude for the eight-point source incident field defined in (5.35) at 40.9 and 81.8 wavelengths. Figure 5.11(e) displays a side view of the 40.9λ far-field magnitude $|\tilde{u}^\infty|$ including the nacelle geometry, for reference, with the intake pointing left. Figures 5.11(f) and 5.11(g), where the geometry is not included, present the far-field, with the positive z direction pointing out of the page, for the 40.9λ and 81.8λ cases, respectively.

5.6 Strong parallel scaling

The observed speedups under strong scaling tests are displayed in Figure 5.12. This figure presents speedup tests for three test cases: a sphere of diameter $d = 128\lambda$

(where, as before, $\lambda = \frac{2\pi}{\kappa}$ denotes the wavelength and d is given in (5.1)), and oblate and prolate spheroids (Figure 5.1) of large diameters $d = 128\lambda$ and $d = 256\lambda$, respectively. The curves in Figure 5.12 display, in each case, the observed speedup S_{1,N_c} for $1 \leq N_c \leq 1,680$ (see Section 5.1.6). In view of the requirements of the strong-scaling setup, test problems were selected that can be run in a reasonable time on a single core and with the memory available in the corresponding compute node. Clearly, such test problems tend to be too small to admit a perfect distribution onto large numbers of cores. As illustrated in Figure 5.12, however, in spite of this constraint, excellent scaling is observed in the complete range going from 1 core to 1,680 cores (30 nodes). As in the weak-scaling tests, further, there is no hard limitation on scaling, even for such small problems, (once again, in line with the discussion presented in Chapter 4), and it is reasonable to expect that, unlike other approaches (for which either hard limits arise [44] as described in the first paragraph of Section 4.1, or which rely on memory duplication [46, 47]), the observed speedup continues to scale with the number of cores, as suggested by Figure 5.12, up to very large numbers of cores. The computing speedups achieved by the proposed parallel strategy outperform those achieved by other MPI-parallel implementations of FMM and other numerical methods [34, 48, 98], and can be best appreciated by noting that, instead of the, e.g., approximately 40 minutes ($2.54 \cdot 10^3$ secs., see first line in Table 5.13) required by a single-core IFGF run, a total of 4.5 secs. ($4.5 = 2.54 \cdot 10^3 / S_{1,1680}$ secs., where, per Figure 5.12, $S_{1,1680} = 565$) suffices for the corresponding 1,680-core IFGF run. It is interesting to note that an approximately 1.51 second 1,680-core run would have resulted under perfect scaling.

Tables 5.13- 5.15, 5.16- 5.18, and 5.19- 5.21, in turn, present the strong parallel efficiencies achieved by the proposed parallel IFGF method under OpenMP, shared-memory MPI, and distributed-memory MPI parallelization strategies, respectively, for each of the three test geometries considered in this section. In detail, these tables display the main two strong parallel performance quantifiers, namely the observed strong parallel efficiency $E^s_{N_c^0,N_c}$ and speedup $S_{N_c^0,N_c}$, along with the computing times T, the obtained accuracy ε, and details concerning the geometry and the discretizations. The tables clearly show that, in all cases, the IFGF parallel efficiencies are essentially independent of the geometry type. With reference to Section 5.1.3 above, the ranges of the parameter N_c considered in these tables span all of the available cores in each one of the relevant hardware units used: 14 cores in a single NUMA node, 56 cores (4 NUMA nodes) in a single compute node, and 16 nodes in

the complete cluster (the largest number of nodes which equals a power of 2 in the cluster used).

The largest efficiency deficit observed as a result of a hardware-doubling transition is the decrease by a full 23% (from 100% to 77%) shown in Tables 5.16- 5.18, which results from the transition from one to two MPI ranks (that is, from one to two 14-core NUMA nodes). We argue that this deficit, which takes place precisely as an MPI communication between NUMA nodes is first introduced, is not a sole reflection of the character of the algorithm in presence of the MPI interface, since such large deficits are not observed in any other MPI related hardware-doubling transitions reported in the various tables. As potential additional contributing elements to this deficit we mention notably, MPI overhead (which would only be incurred in the first doubling transition but not in subsequent doubling transitions, in view of the decreasing number of pairwise communications incurred by the algorithm under a doubling transition in a strong scaling test, as indicated by the theoretical discussion in Section 4.3), and the Intel Turbo Boost Technology inherent in the processors used—which achieve maximum turbo frequencies when running under lower loads, and which, when concurrently using larger numbers of cores in a single node, cease to operate.

A variety of other data is presented in these tables. Tables 5.13- 5.15 and 5.16- 5.18, which demonstrate the strong scaling within a single NUMA node, and among all four NUMA nodes within a compute node, are included for completeness, but as discussed below, we attach far greater significance to Tables 5.19- 5.21, which demonstrate the scaling of the method under the one hardware element that can truly be increased without bounds, namely, the number of compute nodes. In these tables, geometries twice as large than those used for the previous tables are considered (to reasonably increase the minimum computing times), and the hardware is scaled from one compute node to sixteen compute nodes. Per the description in Section 5.1.3, each node is assigned four MPI ranks, each one of which is pinned to one of the four NUMA nodes present in the compute node. Overall, a strong scaling efficiency of over 60% can be observed in all cases, with the results of the sphere test case even above 70% owing to the symmetry of the geometry and the resulting increased load-balance and minimized communication between ranks. The loss of efficiency can be attributed the load-imbalance induced by our data partitioning strategy, the communication between ranks, and the parallelization overhead introduced by MPI and OpenMP.

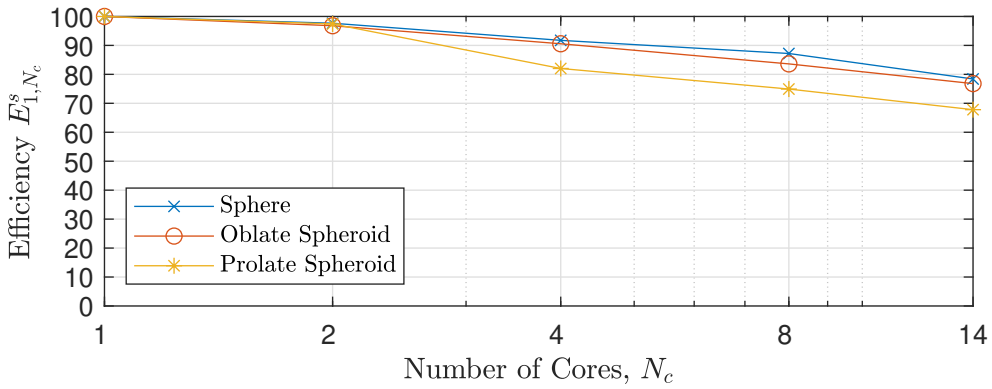


Figure 5.13: Visualization of the strong parallel efficiency E^s_{1,N_c} of the OpenMP parallelization, scaling from 1 to 14 cores for all three test geometries shown in Table 5.13- 5.15.

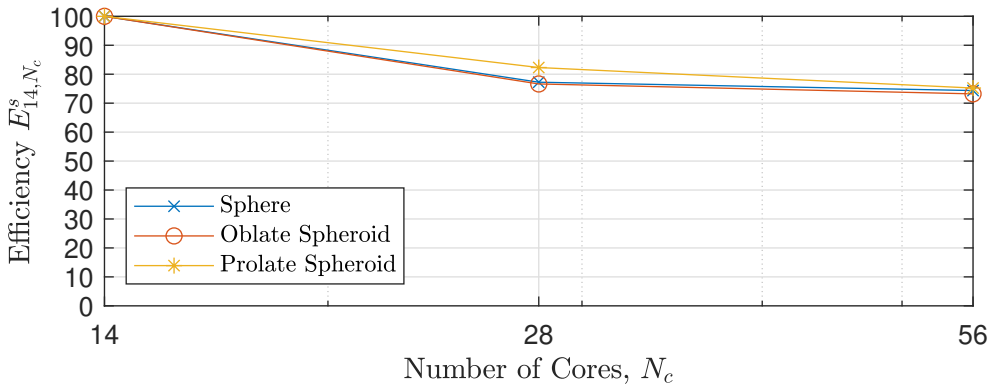


Figure 5.14: Visualization of the strong parallel efficiency E^s_{14,N_c} of the shared-memory MPI parallelization, scaling from 14 to 56 cores for all three test geometries shown in Table 5.16- 5.18.

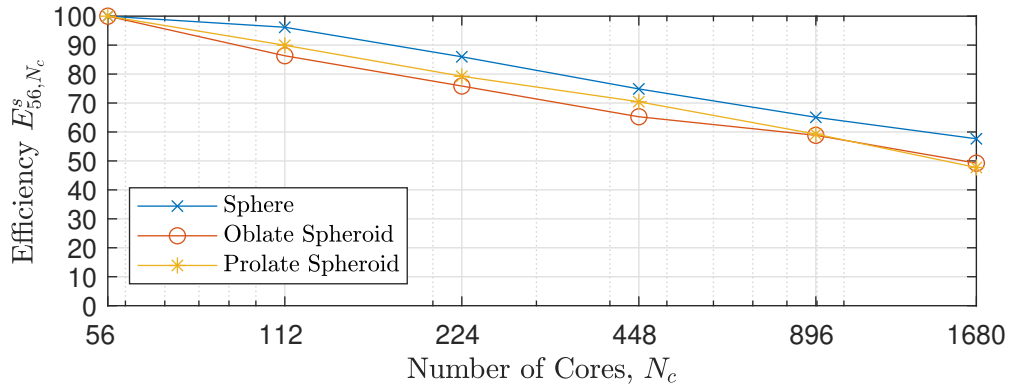


Figure 5.15: Visualization of the strong parallel efficiency E^s_{56,N_c} of the distributed-memory MPI parallelization, scaling from 56 to 1,680 cores for all three test geometries shown in Table 5.19- 5.21.

N	d	N_c	ε	T (s)	E^s_{1,N_c}	S_{1,N_c}
$1,572,864$	128λ	1	$2 \cdot 10^{-3}$	$2.54 \cdot 10^3$	100%	1.00
		2		$1.29 \cdot 10^3$	98%	1.95
		4		$6.91 \cdot 10^2$	92%	3.67
		8		$3.63 \cdot 10^2$	87%	6.98
		14		$2.31 \cdot 10^2$	78%	10.98

Table 5.13: Strong parallel scaling test of the OpenMP IFGF implementation from $N_c = 1$ to $N_c = 14$ cores in a single node for the sphere geometry.

N	d	N_c	ε	T (s)	E^s_{1,N_c}	S_{1,N_c}
$1,572,864$	128λ	1	$5 \cdot 10^{-4}$	$9.42 \cdot 10^2$	100%	1.00
		2		$4.86 \cdot 10^2$	97%	1.94
		4		$2.60 \cdot 10^2$	91%	3.62
		8		$1.40 \cdot 10^2$	84%	6.69
		14		$8.76 \cdot 10^1$	77%	10.75

Table 5.14: Same as Table 5.13 for the oblate spheroid geometry.

N	d	N_c	ε	T (s)	E^s_{1,N_c}	S_{1,N_c}
$6,291,456$	256λ	1	$6 \cdot 10^{-4}$	$1.42 \cdot 10^3$	100%	1.00
		2		$7.29 \cdot 10^2$	97%	1.95
		4		$4.33 \cdot 10^2$	82%	3.28
		8		$2.37 \cdot 10^2$	75%	5.99
		14		$1.49 \cdot 10^2$	68%	9.49

Table 5.15: Same as Table 5.13 for the prolate spheroid geometry.

The most important quality illustrated in these tables is the IFGF's efficiency performance under strong-scaling hardware-doubling transitions demonstrated in the last column of Tables 5.19- 5.18. This performance, which mirrors the corresponding weak-scaling performance presented in the last columns of Tables 5.22- 5.24, shows that, as in the weak scaling case, under the assumption that the displayed trend is maintained for large numbers of nodes (below the obvious limit imposed by the fixed problem size), the parallel IFGF method for a fixed problem can be efficiently run in large numbers of computing cores—with an efficiency factor no worse than a constant $\approx 80\%$ as the hardware sizes are doubled from a given point of reference.

The character of the IFGF algorithm under weak- and strong-scaling hardware-doubling tests, as discussed above in this section and in Section 4.3, would ensure

N	d	N_r	N_c	ε	T (s)	E^s_{14,N_c}	S_{14,N_c}
$1,572,864$	128λ	1	14	$2 \cdot 10^{-3}$	$2.31 \cdot 10^2$	100%	1.00
		2	28	$2 \cdot 10^{-3}$	$1.49 \cdot 10^2$	77%	1.54
		4	56	$2 \cdot 10^{-3}$	$7.77 \cdot 10^1$	74%	2.97

Table 5.16: Strong parallel scaling test of the shared-memory MPI implementation on a single node, transitioning from $N_c = 14$ cores to $N_c = 56$ (all cores available in one compute node) by increasing the number N_r of MPI ranks from 1 to 4, for the sphere geometry.

N	d	N_r	N_c	ε	T (s)	E^s_{14,N_c}	S_{14,N_c}
$1,572,864$	128λ	1	14	$5 \cdot 10^{-4}$	$8.76 \cdot 10^1$	100%	1.00
		2	28	$6 \cdot 10^{-4}$	$5.71 \cdot 10^1$	77%	1.53
		4	56	$6 \cdot 10^{-4}$	$2.99 \cdot 10^1$	73%	2.93

Table 5.17: Same as Table 5.16 for the oblate spheroid geometry.

N	d	N_r	N_c	ε	T (s)	E^s_{14,N_c}	S_{14,N_c}
$6,291,456$	256λ	1	14	$6 \cdot 10^{-4}$	$1.49 \cdot 10^2$	100%	1.00
		2	28	$6 \cdot 10^{-4}$	$9.10 \cdot 10^1$	82%	1.65
		4	56	$5 \cdot 10^{-4}$	$4.97 \cdot 10^1$	75%	3.01

Table 5.18: Same as Table 5.16 for the prolate spheroid geometry.

that, provided the demonstrated trends are maintained (as is expected in view of the discussion in the first paragraph of Section 4.2), the method can be executed successfully in very large hardware infrastructures.

5.7 Weak parallel scaling

Tables 5.22 through 5.24 demonstrate the weak IFGF parallel efficiency, for all three geometries considered, from a single compute node ($N_c^0 = 56$) to 4 and 16 compute nodes ($N_c = 224$ and 896, respectively). We find that the efficiency relative to the base $N_c^0 = 56$ case steadily decreases, but importantly, the weak relative efficiency $E^w_{\frac{N_c}{4},N_c}$ remains essentially constant as N_c increases. Thus, under the assumption that this trend is maintained for arbitrarily large numbers of nodes (as is expected in view of the discussion in the first paragraph of Section 4.2 concerning absence of hard limitations on achievable parallelism), the parallel IFGF method is applicable to arbitrarily large problems—provided correspondingly large hardware is used—

N	d	N_c	ε	T (s)	E^s_{56,N_c}	S_{56,N_c}	$E^s_{\frac{N_c}{2},N_c}$
$6,291,456$	256λ	56	$2 \cdot 10^{-3}$	$3.55 \cdot 10^2$	100%	1.00	-
		112	$2 \cdot 10^{-3}$	$1.80 \cdot 10^2$	99%	1.97	99%
		224	$2 \cdot 10^{-3}$	$9.78 \cdot 10^1$	91%	3.64	92%
		448	$2 \cdot 10^{-3}$	$5.52 \cdot 10^1$	81%	6.44	89%
		896	$2 \cdot 10^{-3}$	$3.12 \cdot 10^1$	71%	11.40	89%

Table 5.19: Strong parallel scaling test of the distributed-memory MPI implementation from $N_c = 56$ to $N_c = 896$ cores (1 to 16 compute nodes) with 4 MPI ranks per node for the sphere geometry.

N	d	N_c	ε	T (s)	E^s_{56,N_c}	S_{56,N_c}	$E^s_{\frac{N_c}{2},N_c}$
$6,291,456$	256λ	56	$6 \cdot 10^{-4}$	$1.34 \cdot 10^2$	100%	1.00	-
		112	$6 \cdot 10^{-4}$	$7.41 \cdot 10^1$	91%	1.81	91%
		224	$6 \cdot 10^{-4}$	$4.17 \cdot 10^1$	80%	3.22	89%
		448	$6 \cdot 10^{-4}$	$2.38 \cdot 10^1$	70%	5.64	88%
		896	$6 \cdot 10^{-4}$	$1.40 \cdot 10^1$	60%	9.56	85%

Table 5.20: Same as Table 5.19 for the oblate spheroid geometry.

N	d	N_c	ε	T (s)	E^s_{56,N_c}	S_{56,N_c}	$E^s_{\frac{N_c}{2},N_c}$
$25,165,824$	512λ	56	$4 \cdot 10^{-4}$	$2.23 \cdot 10^2$	100%	1.00	-
		112	$5 \cdot 10^{-4}$	$1.22 \cdot 10^2$	92%	1.83	92%
		224	$6 \cdot 10^{-4}$	$6.83 \cdot 10^1$	82%	3.28	89%
		448	$6 \cdot 10^{-4}$	$3.74 \cdot 10^1$	75%	5.97	91%
		896	$6 \cdot 10^{-4}$	$2.23 \cdot 10^1$	63%	10.01	84%

Table 5.21: Same as Table 5.19 for the prolate spheroid geometry.

with a constant $\approx 80\%$ efficiency factor as the problem and hardware sizes are both quadrupled from a given point of reference. Section 5.6 demonstrated a similar quality of the proposed algorithm under strong-scaling tests.

N	d	N_c	ε	T (s)	E^w_{56,N_c}	$E^w_{\frac{N_c}{4},N_c}$
1,572,864	128λ	56	$2 \cdot 10^{-3}$	$7.77 \cdot 10^1$	100%	-
6,291,456	256λ	224	$2 \cdot 10^{-3}$	$9.78 \cdot 10^1$	87%	87%
25,165,824	512λ	896	$2 \cdot 10^{-3}$	$1.34 \cdot 10^2$	69%	79%

Table 5.22: Weak scaling test transitioning from 1 to 4 nodes, and then from 4 to 16 nodes, for the sphere geometry. The number of nodes, each one containing $N_c = 56$ cores, is kept proportional to the number of surface discretization points, as required by the weak-scaling paradigm.

N	d	N_c	ε	T (s)	E^w_{56,N_c}	$E^w_{\frac{N_c}{4},N_c}$
1,572,864	128λ	56	$7 \cdot 10^{-4}$	$2.99 \cdot 10^1$	100%	-
6,291,456	256λ	224	$6 \cdot 10^{-4}$	$4.17 \cdot 10^1$	79%	79%
25,165,824	512λ	896	$8 \cdot 10^{-4}$	$5.74 \cdot 10^1$	62%	79%

Table 5.23: Same as Table 5.22 for the oblate spheroid geometry.

N	d	N_c	ε	T (s)	E^w_{56,N_c}	$E^w_{\frac{N_c}{4},N_c}$
6,291,456	256λ	56	$5 \cdot 10^{-4}$	$4.97 \cdot 10^1$	100%	-
25,165,824	512λ	224	$6 \cdot 10^{-4}$	$6.83 \cdot 10^1$	79%	79%
100,663,296	$1,024\lambda$	896	$7 \cdot 10^{-4}$	$9.29 \cdot 10^1$	63%	79%

Table 5.24: Same as Table 5.22 for the prolate spheroid geometry.

5.8 Large sphere tests

Table 5.25 illustrates the performance of the IFGF method in terms of computing time and memory requirements for several large-sphere configurations, all of them run in our full 30 node, 1,680 core cluster. In particular, Table 5.25 shows that, as mentioned in Section 1.2, on the basis of less than 1.5 TB of memory, a sphere 1,389 λ with 1.94 billion DOF was run in a computing time of 1,010 seconds—a computing time that is just a factor of approximately 20 times larger than the time reported in [58], for a similar number of DOFs and the same sphere size, on a computer 78 times larger, containing 131,072 cores, and on the basis of an unspecified amount of memory.

The sphere of acoustic size $1,389\lambda$ in this table coincides with largest sphere test case considered in [58], cited in Table 2 in that reference as a sphere of two-meters in diameter illuminated at the frequency of $f = 238.086$ KHz with 343m/s speed of

N	d	N_c	ε	T (s)	Mem/rank
$1,610,612,736$	$1,389\lambda$		$4 \cdot 10^{-3}$	$3.59 \cdot 10^3$	125.96 GB
$1,610,612,736$	$2,048\lambda$		$7 \cdot 10^{-2}$	$2.84 \cdot 10^3$	105.91 GB
$1,944,000,000$	$1,389\lambda$	$1,680$	$1 \cdot 10^{-1}$	$1.01 \cdot 10^3$	42.45 GB
$1,944,000,000$	$1,389\lambda$		$5 \cdot 10^{-3}$	$2.34 \cdot 10^3$	133.44 GB
$2,120,640,000$	$1,389\lambda$		$5 \cdot 10^{-3}$	$2.38 \cdot 10^3$	134.63 GB

Table 5.25: Large sphere test cases run on thirty 56-core compute nodes (for a total of $1,680$ cores), utilizing thirty MPI ranks. The sphere of acoustic size $1,389\lambda$ in this table coincides with largest sphere test case considered in [58].

sound. The largest discretization presented in the present table for this sphere test-case ($2,120,640,000$ discretization points, a limit induced by the largest number representable by a signed 32-bit integer assumed in our geometry-generation code, which will be avoided in subsequent code implementations by switching to 64-bit integers), is slightly smaller than the $2,300,067,840$ discretization considered in [58] under a 131,072 core run.

Other test cases listed in Table 5.25 include an example for a much larger sphere, $2,048\lambda$ in diameter, as well as other $1,389\lambda$ test cases for various accuracies and discretization sizes—and, in all cases, on the basis of memory consumptions ranging between $\approx$ 1.2TB and $\approx$ 4TB.

Chapter 6

CONCLUDING REMARKS

6.1 Conclusions

This thesis introduced the efficient, novel and extremely simple IFGF approach for the fast evaluation of discrete integral operators of scattering theory in linearithmic time. The theoretical background was thoroughly discussed and the correctness of the approach was demonstrated through mathematical proofs and several numerical examples. Further, a parallelization strategy for the IFGF acceleration method was developed that shows excellent parallel scaling to large core numbers while simultaneously preserving the linearithmic complexity of the sequential algorithm. The proposed parallelization approach exploits the box-cone octree structure inherent in the IFGF method, resulting in a strategy that, per the theoretical discussion in Section 4.1 and in the first paragraph of Section 4.2, is applicable to arbitrarily large number of processing cores, and it thereby does not suffer from bottlenecks or hard limits inherent in approaches that orchestrate the parallelization on the basis of octree-box partitioning only. Finally, a full IFGF-accelerated acoustic scattering solver was presented and applied to several engineering motivated problems. In particular, it was shown in Section 5.5 that the linearithmic scaling and the accuracy of the IFGF method can be preserved in the context of the full solver and that real world engineering problems can be solved with an IFGF-accelerated solver in minimal computing time on reasonable hardware.

6.2 Future work

As indicated throughout this thesis, a number of important improvements to the IFGF method have been left for future work. On one hand, the current implementation of the IFGF method utilizes a Chebyshev interpolation procedure of order P, which is utilized in the *Propagation* function—which requires a total of $O(P^2)$ operations as the accuracy order P of the interpolants is increased. The investigation of higher order methods that reduce this scaling to a linear or linearithmic scaling in P is highly desirable and necessary to achieve a competitive high-order method.

Another important improvement currently under investigation concerns the adaptivity in the box-partitioning method (so as to eliminate large deviations of surface

discretization points per box which impact negatively on the overall efficiency of the algorithm) which partitions boxes more closely aligned with the position and the number of surface discretization points, instead of using a fixed D-leveled box-octree. Clearly, this is not a novel technique and has been applied in the context of other acceleration methods, as shown in, e.g., [20]. Such an adaptive octree structure may result in a lower parallel efficiency and its viability in the context of the IFGF method is subject to further investigation.

Further, except for Section 5.5, only the single-layer potentials for the Helmholtz and Laplace Green functions were considered here, but the proposed methodology is applicable, with minimal modifications (as indicated in Section 5.5), in a wide range of contexts, possibly including elements such as double-layer potentials, mixed formulations, electromagnetic and elastic scattering problems, dielectric problems and Stokes flows, as well as volumetric distribution of sources, etc. studies of the potential advantages offered by the IFGF strategies in these areas are left for future work.

Moreover, the feasibility of implementations on heterogeneous architectures such as, e.g., computer systems that incorporate general purpose graphical processing units (GPUs), is currently under study. In particular, the use of GPUs to accelerate the interpolation processes, which represent the most time consuming part of the IFGF method, appears as highly promising avenue of inquiry.

Finally, minor modifications to the data-decomposition strategy presented in Section 4.2.1 could be introduced to not only (approximately) equipartition the surface discretization points and cone segments among MPI ranks, but to also incorporate the number of actual computations and the amount of data required from other MPI ranks in the partitioning scheme. Such an improved data-decomposition design could indeed be obtained by relying on minor adjustments to the cone and box intervals introduced in Section 4.2.1 leading to improved load-balancing, and, thus, improved parallel efficiency.

BIBLIOGRAPHY

[1] Constantine Sideris, Emmanuel Garza, and Oscar P. Bruno. Ultrafast simulation and optimization of nanophotonic devices with integral equation methods. In: *ACS Photonics* 6.12 (2019), pp. 3233–3240. DOI: 10.1021/acsphotonics.9b01137.

[2] Constantine Sideris et al. Foundry-fabricated grating coupler demultiplexer inverse-designed via fast integral methods. In: *Communications Physics* 5.1 (2022). DOI: 10.1038/s42005-022-00839-w.

[3] William Pinello, Andreas C. Cangellaris, and Albert Ruehli. Hybrid electromagnetic modeling of noise interactions in packaged electronics based on the partial-element equivalent-circuit formulation. In: *IEEE Transactions on Microwave Theory and Techniques* 45.10 (1997), pp. 1889–1896. DOI: 10.1109/22.641787.

[4] Merrill I. Skolnik. Introduction to Radar Systems. Electrical engineering series. McGraw-Hill, 2001. ISBN: 9780071181891.

[5] Parham P. Khial, Alexander D. White, and Ali Hajimiri. Nanophotonic optical gyroscope with reciprocal sensitivity enhancement. In: *Nature Photonics* 12.11 (2018), pp. 671–675. DOI: 10.1038/s41566-018-0266-5.

[6] Lionel Kimerling et al. Design guidelines for optical resonator biochemical sensors. In: *J. Opt. Soc. Am. B* 26.5 (May 2009). DOI: 10.1364/JOSAB.26.001032.

[7] Christopher V. Poulton et al. Coherent solid-state LIDAR with silicon photonic optical phased arrays. In: *Optics Letters* 42.20 (Oct. 2017), pp. 4091–4094. DOI: 10.1364/OL.42.004091.

[8] Firooz Aflatouni et al. Nanophotonic projection system. In: *Optics Express* 23.16 (Aug. 2015), pp. 21012–21022. DOI: 10.1364/OE.23.021012.

[9] Reza Fatemi et al. High sensitivity active flat optics optical phased array receiver with a two-dimensional aperture. In: *Optics Express* 26.23 (Nov. 2018), pp. 29983–29999. DOI: 10.1364/OE.26.029983.

[10] Hooman Abediasl and Hossein Hashemi. Monolithic optical phased-array transceiver in a standard SOI CMOS process. In: *Optics Express* 23.5 (Mar. 2015), pp. 6509–6519. DOI: 10.1364/OE.23.006509.

[11] David Colton and Rainer Kress. Inverse Acoustic and Electromagnetic Scattering Theory. 3rd ed. Applied mathematical sciences. Springer, 2013. ISBN: 9781461449423.

[12] David Colton and Rainer Kress. Integral equation methods in scattering theory. SIAM, 2013. ISBN: 9781611973150.

[13] Rainer Kress. Linear Integral Equations. 3rd ed. Springer, 2014. ISBN: 9781461495932.

[14] Nail A. Gumerov and Ramani Duraiswami. Fast Multipole Methods for the Helmholtz Equation in Three Dimensions. Elsevier Science, 2004. ISBN: 9780080443713.

[15] Weng C. Chew. Waves and Fields in Inhomogenous Media. IEEE Press Series on Electromagnetic Wave Theory. Wiley, 1999. ISBN: 9780780347496.

[16] Sergej Rjasanow and Olaf Steinbach. The Fast Solution of Boundary Integral Equations. 1st ed. Mathematical and Analytical Techniques with Applications to Engineering. Springer, 2007. ISBN: 9780387340418.

[17] James C. Maxwell. A dynamical theory of the electromagnetic field. In: *Philosophical Transactions of the Royal Society of London* 155 (1865), pp. 459–512.

[18] Andreas Kirsch and Frank Hettlich. The Mathematical Theory of Time-Harmonic Maxwell's Equations: Expansion-, Integral-, and Variational Methods. Applied Mathematical Sciences. Springer International Publishing, 2014. ISBN: 9783319110868.

[19] Julius A. Stratton. Electromagnetic Theory. IEEE Press Series on Electromagnetic Wave Theory. Wiley, 2007. ISBN: 9780470131534.

[20] Wolfgang Hackbusch. Hierarchical matrices: Algorithms and analysis. Vol. 49. Springer, 2015. ISBN: 9783662473245.

[21] George Green. An Essay on the Application of Mathematical Analysis to the Theories of Electricity and Magnetism. (Göteborg: Wezäta-Melins 1958). 1828. URL: https://books.google.at/books?id=GwYXAAAAYAAJ.

[22] John C. Strikwerda. Finite Difference Schemes and Partial Differential Equations. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2007. ISBN: 9780898716399.

[23] Junuthula .N. Reddy. An Introduction to the Finite Element Method. McGraw-Hill series in mechanical engineering. McGraw-Hill, 2006. ISBN: 9780071244732.

[24] Allen Taflove and Susan C. Hagness. Computational Electrodynamics: The Finite-difference Time-domain Method. 3rd ed. Artech House antennas and propagation library. Artech House, 2005. ISBN: 9781580538329.

[25] Ivo M. Babuška and Stefan A. Sauter. Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers? In: *SIAM Journal on Numerical Analysis* 34.6 (1997), pp. 2392–2423. DOI: 10.1137/S0036142994269186.

[26] Arnaud Deraemaeker, Ivo Babuška, and Philippe Bouillard. Dispersion and pollution of the FEM solution for the Helmholtz equation in one, two and three dimensions. In: *International Journal for Numerical Methods in Engineering* 46.4 (1999), pp. 471–499. DOI: https://doi.org/10.1002/(SICI)1097-0207(19991010)46:4<471::AID-NME684>3.0.CO;2-6.

[27] Roland Bulirsch, Josef Stoer, and J Stoer. Introduction to numerical analysis. 3rd ed. Springer, 2002. ISBN: 9780387954523.

[28] Peter Deuflhard and Andreas Hohmann. Numerische Mathematik 1: Eine algorithmisch orientierte Einführung. De Gruyter Studium. De Gruyter, 2018. ISBN: 9783110614350.

[29] Peter Deuflhard and Andreas Hohmann. Numerical Analysis in Modern Scientific Computing: An Introduction. 2nd ed. Texts in Applied Mathematics. Springer New York, 2003. ISBN: 9780387215846.

[30] William H. Press et al. Numerical Recipes 3rd Edition: The Art of Scientific Computing. 3rd ed. USA: Cambridge University Press, 2007. ISBN: 0521880688.

[31] Hongwei Cheng et al. A wideband fast multipole method for the Helmholtz equation in three dimensions. In: *Journal of Computational Physics* 216 (2006), pp. 300–325. DOI: https://doi.org/10.1016/j.jcp.2005.12.001.

[32] Björn Engquist and Lexing Ying. Fast directional multilevel algorithms for oscillatory kernels. In: *SIAM Journal on Scientific Computing* 29.4 (2007), pp. 1710–1737. DOI: https://doi.org/10.1137/07068583X.

[33] Lexing Ying et al. A new parallel kernel-independent fast multipole method. In: *Proceedings of the ACM/IEEE SC2003 Conference on Supercomputing (SC'03)* (Nov. 2003). DOI: 10.1109/SC.2003.10013.

[34] Luis Landesa et al. Solution of very large integral-equation problems with single-level FMM. In: *Microwave and Optical Technology Letters* 51 (Oct. 2009), pp. 2451–2453. DOI: 10.1002/mop.24651.

[35] Matthias Messner, Martin Schanz, and Eric Darve. Fast directional multilevel summation for oscillatory kernels based on Chebyshev interpolation. In: *Journal of Computational Physics* 231 (2012), pp. 1175–1196. DOI: doi:10.1016/j.jcp.2011.09.027.

[36] Joel R. Phillips and Jacob K. White. A Precorrected-FFT Method for Electrostatic Analysis of Complicated 3-D Structures. In: *IEEE Transactions on computer-aided design of integrated circuits and systems* 16.10 (1997), pp. 1059–1072. DOI: 10.1109/43.662670.

[37] Oscar P. Bruno and Leonid A. Kunyansky. A fast, high-order algorithm for the solution of surface scattering problems: Basic implementation, tests, and applications. In: *Journal of Computational Physics* 169 (2001), pp. 80–110. DOI: doi:10.1006/jcph.2001.6714.

[38] Elizabeth Bleszynski, Maria Bleszynski, and Thomas Jaroszewicz. AIM: Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems. In: *Radio Science* 31.5 (1996), pp. 1225–1251. DOI: 10.1029/96RS02504.

[39] Emmanuel J. Candès, Laurent Demanet, and Lexing Ying. A fast butterfly algorithm for the computation of fourier integral operators. In: *Multiscale Modeling and Simulation* 7 (2009), pp. 1727–1750. DOI: https://doi.org/10.1137/080734339.

[40] Eric Michielssen and Amir Boag. A multilevel matrix decomposition algorithm for analyzing scattering from large structures. In: *IEEE Transactions on Antennas and Propagation* 44.8 (1996), pp. 1086–1093. DOI: 10.1109/8.511816.

[41] Jack Poulson et al. A parallel butterfly algorithm. In: *SIAM Journal on Scientific Computing* 36.1 (2014), pp. C49–C65. DOI: https://doi.org/10.1137/130921544.

[42] Steffen Börm. Directional H2-matrix compression for high-frequency problems. In: *Numerical Linear Algebra with Applications* 24 (July 2017). DOI: 10.1002/nla.2112.

[43] Oscar Bruno et al. Electromagnetic integral equations requiring small numbers of Krylov-subspace iterations. In: *Journal of Computational Physics* 228.17 (2009), pp. 6169–6183. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2009.05.020.

[44] Austin R. Benson et al. A parallel directional fast multipole method. In: *SIAM Journal on Scientific Computing* 36.4 (2014), pp. C335–C352. DOI: https://doi.org/10.1137/130945569.

[45] Aparna Chandramowlishwaran et al. Optimizing and tuning the fast multipole method for state-of-the-art multicore architectures. In: *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*. 2010, pp. 1–12. DOI: 10.1109/IPDPS.2010.5470415.

[46] Rui-Qing Liu et al. Massively parallel discontinuous galerkin surface integral equation method for solving large-scale electromagnetic scattering problems. In: *IEEE Transactions on Antennas and Propagation* 69.9 (2021), pp. 6122–6127. DOI: 10.1109/TAP.2021.3078558.

[47] Ming-Lin Yang, Yu-Lin Du, and Xin-Qing Sheng. Solving electromagnetic scattering problems with over 10 billion unknowns with the parallel MLFMA. In: *2019 Photonics Electromagnetics Research Symposium - Fall (PIERS*

- *Fall)*. 2019, pp. 355–360. DOI: 10.1109/PIERS-Fall48861.2019.9021504.

[48] Özgür Ergül and Levent Gurel. A hierarchical partitioning strategy for an efficient parallelization of the multilevel fast multipole algorithm. In: *IEEE Transactions on Antennas and Propagation* 57.6 (2009), pp. 1740–1750. DOI: 10.1109/TAP.2009.2019913.

[49] Caleb Waltz et al. Massively parallel fast multipole method solutions of large electromagnetic scattering problems. In: *IEEE Transactions on Antennas and Propagation* 55.6 (2007), pp. 1810–1816. DOI: 10.1109/TAP.2007.898511.

[50] Fangzhou Wei and Ali E. Yilmaz. A more scalable and efficient parallelization of the adaptive integral method—Part I: Algorithm. In: *IEEE Transactions on Antennas and Propagation* 62.2 (2014), pp. 714–726. DOI: 10.1109/TAP.2013.2291559.

[51] Fangzhou Wei and Ali E. Yilmaz. A more scalable and efficient parallelization of the adaptive integral method—Part II: BIOEM application. In: *IEEE Transactions on Antennas and Propagation* 62.2 (2014), pp. 727–738. DOI: 10.1109/TAP.2013.2291564.

[52] Oscar P. Bruno and Emmanuel Garza. A Chebyshev-based rectangular-polar integral solver for scattering by geometries described by non-overlapping patches. In: *Journal of Computational Physics* 421 (2020), p. 109740. DOI: https://doi.org/10.1016/j.jcp.2020.109740.

[53] Vladimir Rokhlin. Diagonal forms of translation operators for the Helmholtz equation in three dimensions. In: *Applied and Computational Harmonic Analysis* 1.1 (1993), pp. 82–93. ISSN: 1063-5203. DOI: https://doi.org/10.1006/acha.1993.1006.

[54] Miloš Nikolić et al. An analysis of FFTW and FFTE performance. In: *High-Performance Computing Infrastructure for South East Europe's Research Communities: Results of the HP-SEE User Forum 2012*. Springer International Publishing, 2014, pp. 163–170. ISBN: 9783319015200. DOI: 10.1007/978-3-319-01520-0_20.

[55] Steffen Börm and Jens Melenk. Approximation of the high-frequency Helmholtz kernel by nested directional interpolation. In: *Numerische Mathematik* 137.1 (Oct. 2017), pp. 1–37. DOI: 10.1007/s00211-017-0873-y.

[56] Mario Bebendorf and Sergej Rjasanow. Adaptive low-rank approximation of collocation matrices. In: *Computing* 70 (Feb. 2003), pp. 1–24. DOI: 10.1007/s00607-002-1469-6.

[57] Oscar P. Bruno and Stéphane K. Lintner. A high-order integral solver for scalar problems of diffraction by screens and apertures in three-dimensional space. In: *Journal of Computational Physics* 252 (2013), pp. 250–274. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2013.06.022.

[58] Mustafa Abduljabbar et al. Extreme scale FMM-accelerated boundary integral equation solver for wave scattering. In: *SIAM Journal on Scientific Computing* 41.3 (2019), pp. C245–C268. DOI: 10.1137/18M1173599.

[59] Mathias Winkel et al. A massively parallel, multi-disciplinary Barnes–Hut tree code for extreme-scale N-body simulations. In: *Computer physics communications* 183.4 (2012), pp. 880–889. DOI: https://doi.org/10.1016/j.cpc.2011.12.013.

[60] Dhairya Malhotra and George Biros. Algorithm 967: A distributed-memory fast multipole method for volume potentials. In: *ACM Transactions on Mathematical Software (TOMS)* 43.2 (2016), pp. 1–27. DOI: https://doi.org/10.1145/2898349.

[61] Michael S. Warren. 2HOT: An improved parallel hashed oct-tree N-Body algorithm for cosmological simulation. In: *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 2013, pp. 1–12. DOI: 10.1145/2503210.2503220.

[62] Youcef Saad and Martin H Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. In: *SIAM Journal on scientific and statistical computing* 7.3 (1986), pp. 856–869. DOI: https://doi.org/10.1137/0907058.

[63] John P Boyd. Chebyshev and Fourier spectral methods. 2nd ed. Courier Corporation, 2001. ISBN: 9780486141923.

[64] Leslie Fox and Ian B. Parker. Chebyshev polynomials in numerical analysis. Oxford mathematical handbooks. Oxford U.P., 1968. ISBN: 9780198596141.

[65] Thomas Rauber and Gudula Rünger. Parallel programming. 2nd ed. Springer, Berlin, Heidelberg, 2013. ISBN: 9783642378010.

[66] Thomas Sterling, Maciej Brodowicz, and Matthew Anderson. High performance computing: Modern systems and practices. Elsevier Science, 2017. ISBN: 9780124202153.

[67] Christopher Dahnken et al. Optimizing HPC applications with Intel cluster tools. Oct. 2014. ISBN: 978-1-4302-6496-5. DOI: 10.1007/978-1-4302-6497-2.

[68] Christoph Bauinger and Oscar P. Bruno. "Interpolated Factored Green Function" method for accelerated solution of scattering problems. In: *Journal of Computational Physics* 430 (Jan. 2021). DOI: 10.1016/j.jcp.2020.110095.

[69] Faà di Bruno. Note sur une nouvelle formule de calcul differentiel. In: *Quarterly Journal of Pure and Applied Mathematics* 1 (1857), pp. 359–360.

[70] Thomas H. Cormen et al. Introduction to algorithms. MIT Press, 2009. ISBN: 9780262533058.

[71] Joseph O'Rourke. Finding minimal enclosing boxes. In: *International Journal of Computational and Information Sciences* 14 (1985), pp. 183–199. DOI: https://doi.org/10.1007/BF00991005.

[72] Christoph Bauinger and Oscar P. Bruno. Massively parallelized Interpolated Factored Green Function method. In: *arXiv:2112.15198* (2021).

[73] Mark Bull et al. Performance evaluation of mixed-mode OpenMP/MPI implementations. In: *International Journal of Parallel Programming* 38 (Oct. 2010), pp. 396–417. DOI: 10.1007/s10766-010-0137-2.

[74] Nikolaos Drosinos and Nectarios Koziris. Performance comparison of pure MPI vs hybrid MPI-OpenMP parallelization models on SMP clusters. In: *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* 2004. DOI: 10.1109/IPDPS.2004.1302919.

[75] Dana Akhmetova et al. Performance study of multithreaded MPI and OpenMP tasking in a large scientific code. In: *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 2017, pp. 756–765. DOI: 10.1109/IPDPSW.2017.128.

[76] Michel F. Dekking et al. A modern introduction to probability and statistics: Understanding why and how. Springer Texts in Statistics. Springer, 2005. ISBN: 9781852338961.

[77] James Reinders et al. Data parallel C++: Mastering DPC++ for programming of heterogeneous systems using C++ and SYCL. Springer Nature, 2021. ISBN: 9781484255742.

[78] Peter S. Pacheco. An introduction to parallel programming. Morgan Kaufmann, 2011. ISBN: 9780123742605.

[79] Michael S. Warren and John K. Salmon. A parallel hashed oct-tree N-body algorithm. In: *Supercomputing '93:Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*. 1993, pp. 12–21. DOI: https://doi.org/10.1145/169627.169640.

[80] Amanda K Sharp. Memory layout transformations. https://www.intel.com/content/www/us/en/developer/articles/technical/memory-layout-transformations.html. Accessed: 2021-12-20.

[81] Hanan Samet. The design and analysis of spatial data structures. USA: Addison-Wesley Longman Publishing Co., Inc., 1990. ISBN: 0201502550.

[82] Edwin Jimenez, Christoph Bauinger, and Oscar P. Bruno. IFGF-accelerated integral equation solvers for acoustic scattering. In: *arXiv:2112.06316* (2021).

[83] Hank Childs et al. VisIt: An end-user tool for visualizing and analyzing very large data. In: *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*. Oct. 2012, pp. 357–372.

[84] John J. Bowman, Thomas B. A. Senior, and Piergiorgio L. E. Uslenghi. Electromagnetic and acoustic scattering by simple shapes. 1st ed. CRC Press, 1988. ISBN: 9780891168850.

[85] Christopher W. Nell and Layton E. Gilroy. An improved BASIS model for the BeTSSi submarine. In: *DRDC Atlantic TR* 199 (2003).

[86] Hans G. Schneider et al. Acoustic scattering by a submarine: Results from a benchmark target strength simulation workshop. In: *ICSV10* (2003), pp. 2475–2482.

[87] Sascha Merz, Roger Kinns, and Nicole Kessissoglou. Structural and acoustic responses of a submarine hull due to propeller forces. In: *Journal of sound and vibration* 325.1-2 (2009), pp. 266–286. DOI: https://doi.org/10.1016/j.jsv.2009.03.011.

[88] Ilkka Karasalo. Modelling of acoustic scattering from a submarine. In: *Proceedings of Meetings on Acoustics ECUA2012* 17.1 (2012). DOI: 10.1121/1.4767963.

[89] Yingsan Wei et al. Scattering effect of submarine hull on propeller non-cavitation noise. In: *Journal of sound and vibration* 370 (2016), pp. 319–335. DOI: https://doi.org/10.1016/j.jsv.2016.01.027.

[90] Claudio Testa and Luca Greco. Prediction of submarine scattered noise by the acoustic analogy. In: *Journal of Sound and Vibration* 426 (2018), pp. 186–218. DOI: https://doi.org/10.1016/j.jsv.2018.04.011.

[91] Jon Vegard Venås and Trond Kvamsdal. Isogeometric boundary element method for acoustic scattering by a submarine. In: *Computer Methods in Applied Mechanics and Engineering* 359 (2020). DOI: https://doi.org/10.1016/j.cma.2019.112670.

[92] Ali H. Nayfeh, John E. Kaiser, and Demetri P. Telionis. Acoustics of aircraft engine-duct systems. In: *AIAA Journal* 13.2 (1975), pp. 130–153. DOI: https://doi.org/10.2514/3.49654.

[93] Walter Eversman. Theoretical models for duct acoustic propagation and radiation. In: *NASA. Langley Research Center, Aeroacoustics of Flight Vehicles: Theory and Practice. Volume 2: Noise Control* (1991).

[94] Damiano Casalino et al. Aircraft noise reduction technologies: A bibliographic review. In: *Aerospace Science and Technology* 12.1 (2008), pp. 1–17. DOI: https://doi.org/10.1016/j.ast.2007.10.004.

[95] Matthew Fergus Kewin. Acoustic liner optimisation and noise propagation through turbofan engine intake ducts. PhD thesis. University of Southampton, 2013.

[96] K. R. Pyatunin, N. V. Arkharova, and A. E. Remizov. Noise simulation of aircraft engine fans by the boundary element method. In: *Acoustical Physics* 62.4 (2016), pp. 495–504. DOI: 10.1134/S1063771016040151.

[97] MSC Software. Customer Case Studies 2021-04-27; Case Study: Airbus: Simulation Helps Airbus Optimize Acoustic Liners and Reduce Aircraft Noise. https://files.mscsoftware.com/cdn/farfuture/-WeJrL0uOWa_LL2SdOen2NMogjF1cpQl5PQF07shqyg/mtime:1401395701/sites/default/files/cs_airbus_ltr_w_4.pdf, last checked on 2021-10-14. 2021.

[98] Miguel Ruiz-Cabello N. et al. Performance of parallel FDTD method for shared- and distributed-memory architectures: Application to bioelectromagnetics. In: *PLOS ONE* 15.9 (Sept. 2020), pp. 1–16. DOI: 10.1371/journal.pone.0238115.

INDEX

Symbols

D

E

F

S

T

U

W

Z